**Ministry of Transport and Communication of Ukraine**
**Ukraine State Committee of Communications**
**Odessa National Academy of Telecommunications Named After A.S. Popov**

**Sub-faculty of information technologies**

# COMPUTER SCIENCE

**Module 1**

***The main data on a personal computer and about the organization of computing processes***

**Part 1**

***Lecture notes***

Odessa 2008

УДК 004.43 План УМИ 2008 г.

Composer - **Y. V. Prokop**

These lecture notes contain brief information on personal computers, Windows XP operating system, Microsoft Word text processor, and theoretical data and examples of projects in C ++ Builder for calculation of complex mathematical formulas and solving problems with branching. These lecture notes will be useful for students of the Academy of Telecommunications who are studying in English, fixing theoretical material, preparation for laboratory works, and practical occupations in the discipline of Computer science in the first module.

It is intended for the acquisition of skills for operation on a personal computer and programming by students of the academy studying in English, with the purpose of further usage of these skills in daily present and future professional work. Also, it will be further useful for users of personal computers, wishing to learn programming in the C ++ Builder environment.

The lecture notes have been approved by the sub-faculty IT meeting

Minutes № from

The head of the chair Leonov Y. G.

The lecture notes are considered and approved by the faculty of Informational Networks meeting

Minutes № from

The dean of the faculty Strelkovskaya I. V. B.

# Contents

# Introduction

## Structure of the module

The discipline of Computer science is studied in I - II semesters and is intended for the training of students for the professional use of personal computers.

The purpose of the Computer science course is to form knowledge and skills in areas such as:

- architecture of a personal computer
- operation in Windows operating system
- algorithmization of computing processes
- compilation of programs on C ++ programming language
- acquaintance to object oriented programming on an example of solution of the elementary tasks in C ++ Builder programming environment.

The course program consists of four modules:

- module 1 – "The main data on a personal computer and about the organization of computing processes";
- module 2 – "Programming of tasks with loops and arrays";
- module 3 – "Programming of tasks with the structured data";
- module 4 – "Programming of tasks with lists and files. Bases of object oriented programming".

According to the curriculum, the structure of the module is as following:

| **Type of activity** | **Hours** |
|---|---|
| Lectures | 8 |
| Exercises | 16 |
| Laboratory Training | 16 |
| Total Classwork Hours | 40 |
| Individual Work and Self-study | 27 |
| TOTAL | 67 |

## The subject schedule of lectures

**Lecture 1**. Introduction. Personal computer. Microsoft Windows. Microsoft Word.

**Lecture 2**. Introduction to programming. C++ Builder IDE.

**Lecture 3**. Arithmetic expressions in C++. Programs with linear structure.

**Lecture 4**. Conditions. If and switch conditional statements.

## The list of exercises in module 1

1. Binary, octal and hexadecimal number systems. Windows operating system.
2. Mathematical expressions. Programs with linear structure.
3. Console. Conditions.
4. Programs with if and switch statements.

## The list of laboratory trainings in module 1

1. Operation with files and folders in Microsoft Windows. Text document creation.
2. Creation and formatting the text in Microsoft Word processor.
3. Tables' creation and processing in Microsoft Word. Graphic images and formula editor in Word.
4. Introduction to Borland C++ Builder IDE: project, form, components properties and events.
5. Mathematical expressions. Programs with linear structure.
6. Console application.
7. Programs with branching.
8. Programs with the operator of variants.

## The list of knowledge and skills with which the student should start learning a contents of the given module

The study of computer science is based on the school course of computer science (students should own knowledge in size of school course of computer science according to the program of the Ministry of Education) and is based on school courses of mathematics and the discipline of Higher mathematics (functions, formulas of conversion, a factorial, numbers, an integral, a matrix, operations with matrixes, etc.).

## Literature

1. Буката Л. Н., Кузнецов В.Д. Информатика. Модуль 1. – ОНАС, 2008.
2. Архангельский А.Я., Тагин М.А. Программирование в C++ Builder 6 и 2006. – М.: «Бином», 2007. – 1184 с.
3. Леонов Ю.Г., Угрік Л.М., Швайко І.Г. Збірник задач з програмування. – Одесса: УДАЗ, 1997.
4. Архангельский А.Я. Программирование в C++ Builder 5.– М.:«Бином», 2000.– 1152с.
5. Березин Б.Н., Березин С.Б. Начальный курс С и С++. – М.: «Диалог-МИФИ», 2000. – 288 с.
6. Бьерн Страуструп Язык программирования С++. – СПб.– М.: Бином, 1999. – 991 с.
7. The cplusplus.com tutorial – http://www.cplusplus.com

## Guidelines to self-study of the student

For self-study and execution of individual tasks 27 hours are given to the student. It is recommended to arrange this time as follows:

| Type of activity | Hours |
|---|---|
| Studying of lectures | 4 |
| Learning of an additional material to lectures | 4 |
| Preparation for exercises | 4 |
| Preparation for laboratory works | 9 |
| Execution of the complex individual task on a theme: “Compilation of algorithms and programs with linear and branching structure” | 6 |
| Total: | 27 |

# Lecture 1. Introduction. Personal computer. Microsoft Windows. Microsoft Word

## *Personal computer*

Computers consist of **hardware** and **software** components:

**Hardware** is the physical equipment: integrated circuit boards, disk drives, terminals (the screen and keyboard together), tape drives, printers, and plotters.

**Software** is the programs that instruct the hardware to do something.

Every computer has the following general structure (fig. 1.1):

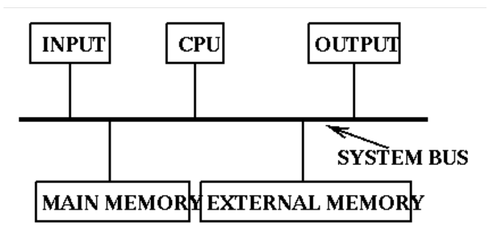


Fig. 1.1 General structure of computer

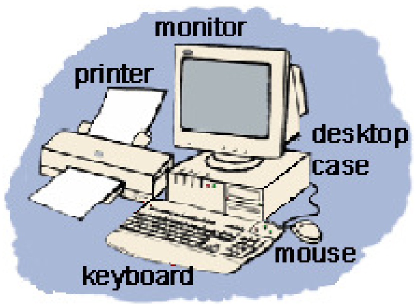


Fig. 1.2 Personal computer with desktop case

### Hardware

Hardware is all of the electronic equipment that a computer includes. If you can touch it, pick it up, or move it around, it is hardware.

A computer consists of a system unit, and external devices such as a monitor, keyboard, mouse, printer, etc. (fig. 1.2).

There are 8 basic parts to a computer:

1. **Case** houses all of the important computer components. If it stands up tall, it is a tower case (fig. 1.3). If it sits flat, it is a desktop case (fig. 1.2).
2. **Power Supply** is a case that holds a transformer, voltage control, and a cooling fan, and supplies power to the rest of the computer.
3. **Mainboard** (motherboard) is a large Printed Circuit Board (PCB) with fixed components and sockets that accept smaller PCBs and other components, hosting:

– The microprocessor or **Central Processing Unit** (CPU), with its own heat sink and cooling fan.

– Small PCBs with fast **Random Access Memory** (RAM).

– **Interface boards** (PCBs) for Hard Disk (HD) Drives, Floppy Disk Drives (FDD), Compact Disk (CD) Drives and external peripherals (the connectors on the back of the base unit are integral parts of the interface boards for external peripherals).

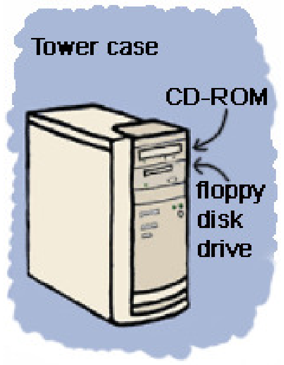


Fig. 1.3 Tower case

4. CPU (**central processing unit**) is the "brain" of the computer. Every instruction given to the computer passes through the electronic circuits of the CPU. When you program a computer to add two numbers together, the arithmetic is performed in the CPU. When you want something sorted, the CPU controls the task from start to finish. The type of CPU in a computer determines how fast the computer

can operate. A CPU generates a lot of heat, so there is usually a small fan nearby to cool it down.

5. **Memory**. Each program you run is loaded into RAM (random access memory). However, when the computer is switched off, all of this information disappears. RAM is also referred to as thinking memory. If computer has more RAM, it works (thinks) faster.

6. **Video Card** (video display controller) produces the output for the computer display. This will either be built into the motherboard or attached in its own separate slot (PCI, PCI-E or AGP), in the form of a Graphics Card.

7. **Hard Disk** (HDD) stores all the information in a computer even when it is switched off.

8. **CD** or **DVD** Drive. A CD-ROM can only read information from the disk. Many computers have a CD-RW (RW stands for ReWrite) instead of a CD-ROM. CD-RW allows you to write information to the disk as well as read from it. Also, new computers may have a DVD (Digital Video Disk) drive instead of a CD-ROM or CD-RW. A DVD holds much more information.

In fig. 1.4 you can see how components are placed inside a system unit.

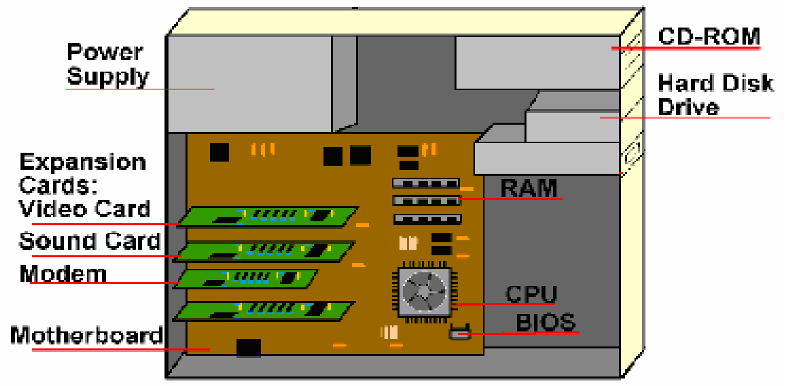


Fig.1.4 System unit: Components inside a computer

The computer can also contain:

**Sound cards** enable computers to output sound to audio devices, as well as accept input from a microphone. Most modern computers have sound cards built-in to the motherboard, though it is common for a user to install a separate sound card as an upgrade.

**Networking** connects the computer to the internet and/or other computers:

- **modem** – for dial-up connections. The modem is an expansion card that allows computers to communicate with each other. A modem plugs the computer in to a phone or cable line so that information can be transferred between computers.
- **network card** – for DSL/Cable internet, and/or connecting to other computers.

## Input devices

There are several ways to get new information or **input** into a computer. The two most common ways are the **keyboard** and the **mouse**. The keyboard has keys for **characters** (letters, numbers and punctuation marks) and special commands. Pressing the keys tells the computer what to do or what to write. The mouse allows you to move the **cursor** around on screen. By clicking on the buttons on the mouse, you give the computer directions on what to do. There are other devices similar to a mouse that can be used in its place. A **trackball** has the ball on top and you move it with your finger. A **touchpad** allows you to move your finger across a pressure sensitive pad and press to click.

Other types of input devices allow you to put images into the computer. A **scanner** copies a picture or document into the computer. There are several types of scanners and some look very different, but most look like a flat tray with a glass pane and a lid to cover it. You can input photographs into a computer with a **digital camera**.

## Output Devices

Output devices display information in a way that you can understand. The most common output device is a **monitor**. It houses the computer screen. The monitor allows you to "see" what you and the computer are doing together.

**Speakers** are output devices that allow you to hear sound from your computer. Computer speakers are like stereo speakers.

A **printer** is another common part of a computer system. It takes what you see on the computer screen and prints it on paper. There are three types of printers. An **inkjet printer** uses inks to print. It is the most common printer used with home computers and it can print in either black and white and/or color. **Laser printers** run much faster because they use lasers to print. Laser printers are mostly used in businesses. Black and white laser printers are the most common, but some print in color, too. Also, there is an old type of printer called the **matrix printer**. Basically they are used in cash registers for printing receipts, tickets, etc.

## Ports

**Ports** are on the outside of the computer case where you plug in hardware. On the inside of the case, they are connected to expansion cards. The keyboard, mouse, monitor, and printer all plug into ports. There are also extra ports to plug in extra hardware like joysticks, gamepads, scanners, digital cameras and the like. The ports are controlled by their expansion cards which are plugged into the motherboard and are connected to other components by **cables** which are long, flat bands that contain electrical wiring.

## System bus

All communication between the individual major components is via the **system bus**. The bus is merely a cable which is capable of carrying signals representing data from one place to another.

When data must be sent from memory to a printer then it will be sent via the system bus. The control signals that are necessary to access memory and to activate the printer are also sent by the CPU via the system bus.

A crucial component of a computer is the **BIOS** (Basic Input Output System) chip. In very simple terms, the BIOS chip wakes up the computer when you turn it on and reminds it what parts it has and what they do.

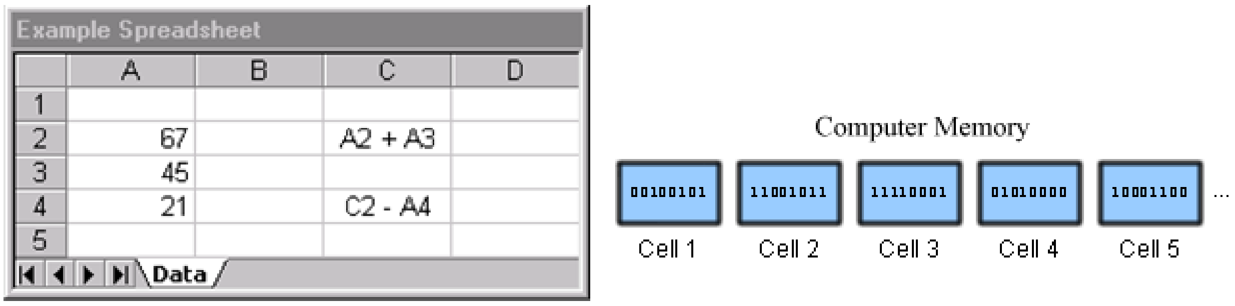
## *Memory*

The fundamental unit of data storage in a computer is called a bit or **bi**nary digi**t**. A bit is similar to a two-way switch. Just like a switch has two states (*off* or *on*), a bit also has two states (0 or 1). Often these two states represent the values *false* or *true* and are implemented inside a computer by using a *low voltage* value or a *high voltage* value. Since bits provide the foundation for all data storage, it is not surprising that the binary number system is very important to computers.

By themselves, bits are not very interesting or useful. In order to store more complex forms of data, bits are joined together into larger groups known as **bytes**. Every byte is made up of eight bits and can store one character symbol. Integer numbers can be stored in 2 or 4 bytes. The word "computer" occupies 8 bytes.

We can assign particular patterns of bits to represent common symbols such as letters, punctuation marks, and numerals. One very common representation of these symbols is **ASCII**, the American Standard Code for Information Interchange.

The main memory of a computer (RAM) is composed of millions of storage cells similar to the one illustrated in fig. 1.5. The size of the storage cells is known as the word size for the computer. In some computers, the word size is one byte while in other computers the word size is two, four, or even eight bytes. Each storage cell in the main memory has a particular address which the computer can use for storing or retrieving data. This arrangement of cells is somewhat similar to a computer spreadsheet where each box of the spreadsheet can hold various data. Just like the boxes of the spreadsheet are identified by a row and column combination (e.g., A2, C4, etc.), the cells of a computer's main memory are identified by a particular address (e.g., Cell 1, Cell 2, etc.). The addresses begin at 0 and increases by 1 until the end of the main memory is reached.



Example Spreadsheet

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | | | |
| 2 | 67 | | A2 + A3 | |
| 3 | 45 | | | |
| 4 | 21 | | C2 - A4 | |
| 5 | | | | |

Fig. 1.5 Model of computer memory

Because computers have such large amounts of RAM, the size of the main memory is usually measured in megabytes (MB) rather than just bytes. One megabyte is equal to $2^{20}$ bytes or 1,048,578 bytes. Some other common measures for quantities of bytes are listed in the table 1.1.

**Table 1.1 Measures for quantities of bytes**

| | | | |
|---|---|---|---|
| Kilobyte (KB) | 1024 or $2^{10}$ bytes | 1,024 bytes | Thousands of bytes |
| Megabyte (MB) | $1024^2$ or $2^{20}$ bytes | 1,048,578 bytes | Millions of bytes |
| Gigabyte (GB) | $1024^3$ or $2^{30}$ bytes | 1,073,741,824 bytes | Billions of bytes |
| Terabyte (TB) | $1024^4$ or $2^{40}$ bytes | 1,099,511,627,776 bytes | Trillions of bytes |

## ***Number systems***

With digital devices it is necessary to deal with various types of information. It is mostly the binary information, such as whether the device is switched on or off, the device is serviceable or not. The information can be presented in the form of a text. It is necessary to encode characters of the alphabet by means of binary levels of a signal. Often the information can represent numbers. Numbers can be presented in various number systems. Number systems may be **positional** or **not positional**.

**Not positional** systems are such number systems in which each character saves the value irrespective its position among other characters representing a number. An example of a not positional number system is the Roman system.

The number system is named **positional** if the same digit has various values, determined as a position of digit in the sequence of digits representing a number. An example of a positional number system is the decimal system used in a daily life.

The quantity ***p*** of the various digits used in the positional system defines the name of a number system and is called a **base number** – *"**p**"*.

In the decimal system, ten digits are used: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. This system has the basis number ten.

Any number ***N*** in a positional number system with the basis ***p*** can be presented in the form of a polynomial from:

$$N = a_n p^n + a_{n-1} p^{n-1} + \dots + a_1 p + a_0 + a_{-1} p^{-1} + a_{-2} p^{-2} + \dots$$

Here $\boldsymbol{N}$ = number, $\boldsymbol{a_j}$ = coefficients (digits of number), $\boldsymbol{p}$ = a base number ($p > 1$). It is accepted to represent numbers in the form of a sequence of digits:

$$N = a_n a_{n-1} \dots a_1 a_{0\dots}$$

In the COMPUTER positional number systems without decimal basis are applied: binary, octal, and hexadecimal (table 1.2).

**Table 1.2 The Most important number systems**

| **Binary (base number 2)** | **Octal (base number 8)** | | **Decimal (base number 10)** | **Hexadecimal (base number 16)** | |
|---|---|---|---|---|---|
| 0 | 0 | 000 | 0 | 0 | 0000 |
| 1 | 1 | 001 | 1 | 1 | 0001 |
| | 2 | 010 | 2 | 2 | 0010 |
| | 3 | 011 | 3 | 3 | 0011 |
| | 4 | 100 | 4 | 4 | 0100 |
| | 5 | 101 | 5 | 5 | 0101 |
| | 6 | 110 | 6 | 6 | 0110 |
| | 7 | 111 | 7 | 7 | 0111 |
| | | | 8 | 8 | 1000 |
| | | | 9 | 9 | 1001 |
| | | | | A | 1010 |
| | | | | B | 1011 |
| | | | | C | 1100 |
| | | | | D | 1101 |
| | | | | E | 1110 |
| | | | | F | 1111 |

The **binary number system** uses two digits: 0 and 1. In binary system any number can be presented in the form of:

$$N = b_n b_{n-1} \ldots b_1 b_0$$

where $\boldsymbol{b_i}$ are 0 or 1.

The **octal number system** uses eight digits: 0, 1, 2, 3, 4, 5, 6, 7. For representation of one digit in the octal system three binary bits (triad) are used (Table 1.2).

The **hexadecimal number system** uses 16 digits. The first ten digits of this system are designated by digits from 0 up to 9 and six uppercase Latin characters: A, B, C, D, E, F. For the representation of one digit in the hexadecimal number system (table 1.2) four binary bits (tetrad) are used.

## The translation of numbers from one number system to another

**The translation of numbers to the decimal system** is carried out by a compilation of a power series with the basis of that system from which the number is translated. Then the value of the sum is added up.

**Examples 1**

1. Translate $10101101.101_2 \rightarrow$"10" n. s.

$10101101.101_2 = 1\cdot2^7 + 0\cdot2^6 + 1\cdot2^5 + 0\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} = 173.625_{10}$

2. Translate $703.04_8 \rightarrow$"10" n. s.

$703.04_8 = 7\cdot8^2 + 0\cdot8^1 + 3\ 8^0 + 0\cdot8^{-1} + 4\cdot8^{-2} = 451.0625_{10}$

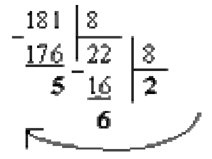
3. Translate $B2E.4_{16} \rightarrow$"10" n. s.

$B2E.4_{16} = 11\cdot16^2 + 2\cdot16^1 + 14\cdot16^0 + 4\cdot16^{-1} = 2862.25_{10}$

**The translation of the integer decimal numbers to the not decimal number system** is carried out by a sequential division of a decimal number by the basis of that system in which it is translated until it will turn out to be the quotient smaller than basis. The number in the new system is written in the form of the rests of division, beginning from the last.

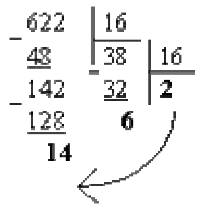
**Examples 2**

1. Translate $181_{10} \rightarrow$"8 " n. s.



Result: $181_{10} = 265_8$

2. To translate $622_{10} \rightarrow$"16" n. s.



Result: $622_{10} = 26E_{16}$

For the translation of **octal** or a **hex number to the binary form**, it is enough to substitute each digit of this number with an appropriate three-bit binary number (triad) (Tab. 1.2) or four-digit binary number (tetrad) (Tab. 1.2), thus discard unnecessary zero in seniors and low-order digits.

**Example 3**

Translate $305.4_8$→" 2 " n. s.

$$\underbrace{3}_{011}\;\underbrace{0}_{000}\;\underbrace{5}_{101}\;.\;\underbrace{4}_{100}{}_{8} = 11000101.1_2$$

1. Translate $7B2.E_{16}$ →"2" n. s.

$$\underbrace{7}_{0111}\;\underbrace{B}_{1011}\;\underbrace{2}_{0010}\;.\;\underbrace{E}_{1110}{}_{16} = 11110110010.111_2$$

In order to transition **from the binary to the octal (hexadecimal) system** do the following: move from a point to the left and to the right, divide binary number into groups on three (four) bits, supplementing, if necessary, in the zero extreme left and right groups. Then a triad (tetrad) will be substituted in the corresponding octal (hexadecimal) digit.

**Example 4**

Translate $1101111001.1101_2$→" 8 " n. s.

$$\underbrace{001}_{1}\underbrace{101}_{5}\underbrace{111}_{7}\underbrace{001}_{1}.\underbrace{110}_{6}\underbrace{100}_{4} = 1571.64_8$$

To translate$11111111011.100111_2$→" 16 " n. s.

$$\underbrace{0111}_{7}\underbrace{1111}_{F}\underbrace{1011}_{B}.\underbrace{1001}_{9}\underbrace{1100}_{C} = 7FB.9C_{16}$$

Translation **from** the **octal in hexadecimal system** is also carried out through the binary system by means of triads and tetrads.

**Example 5**

Translate $175.24_8$→" 16 " n. s.

$$\underbrace{1}_{001}\;\underbrace{7}_{111}\;\underbrace{5}_{101}\;.\;\underbrace{2}_{010}\;\underbrace{4}_{100}{}_{8} = 1111101.0101_2 = \underbrace{0111}_{7}\underbrace{1101}_{D}.\underbrace{0101}_{5}{}_{2} = 7D.5_{16}$$

Result: $175.24_8 = 7D.5_{16}$.

## Binary arithmetic

Rules of arithmetic operations above binary numbers are set by table 1.3 binary additions, subtraction and multiplying.

**Table 1.3 Binary arithmetic**

| The table of binary addition | The Table of binary subtraction | The Table of binary multiplying |
|---|---|---|
| 0+0=0 | 0-0=0 | 0×0=0 |
| 0+1=1 | 1-0=1 | 0×1=0 |
| 1+0=1 | 1-1=0 | 1×0=0 |
| 1+1=10 | 10-1=1 | 1×1=1 |

At addition of binary numbers each pair of digits is added. Thus, it is necessary to consider that 1+1 results in zero in the given bit and 1 is carried in previous bit.

**Example 6**

Add binary numbers:

X=1101, Y=101;

```
         1  1  <-units of carry
   X=  +1101
   Y=    101
       ------
 X+Y=  10010
```

Result 1101+101=10010.

X=1101, Y=101, Z=111;

```
          1        <-units of carry
          111
     X=  +1101
     Y=  + 101
     Z=    111
         ------
 X+Y+Z=  11001
```

Result 1101+101+111=11001.

The subtraction of binary numbers in the given bit 1 of high bit is engaged if necessary. This borrowed unit is equal to two units in a given bit.

**Example 7**

Binary numbers X=10010 and Y=101 are set. Calculate X-Y.

$$\begin{array}{r} \dot{1}00\dot{1}0 \\ -101 \\ \hline 01101 \end{array}$$

Result 10010 - 101=1101.

## *Microsoft Windows*

**Software** is the program that controls everything the computer does for you. The software makes the computer run and determines whether the computer acts as a text processor, graphic artist, office manager, or performs any of the hundreds of tasks that a computer can do for you.

There are two types of software:

**Application software** – programs designed to perform specific tasks that are transparent to the user;

**System software** – programs that support the execution and development of other programs. Two major types of system software are **operating systems** and **translation systems**.

**An operating system** oversees and directs the processing of all programs (fig. 1.6). It provides basic functions for computer such as:

- Controls hardware (printer, monitor, etc.)
- Runs programs

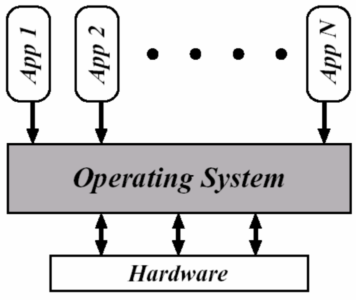


Fig. 1.6 Operating system

- Organizes information.

The most famous operating system is **Microsoft Windows**. Other operating systems are Unix, Linux (for network), OS/2, Macintosh etc.

**A translation system** is a set of programs used to develop software. A key component of a translation system is a translator. Some types of translators are:

- **Compiler** – converts from one language to another
- **Linker** – combines resources

Examples: Microsoft Visual C++®, CBuilder®, g++, Code Warrior®. They perform compilation, linking, and other activities.

When you turn a computer on, it will go through a process called booting. That is short for bootstrapping. When you press the “On” switch, a small program called a bootstrap loader runs. This program, in turn, loads the operating system which controls your PC’s functions. This process is, in effect, the computer “pulling itself up by its own bootstraps”.

During bootup, you will normally see a black screen with text summaries as the system checks itself over. After about a half of a minute, you will see the Windows startup screen displayed and then Windows itself will start. Then you will be presented with the Windows Desktop, your working and playing environment.

Microsoft Windows has Graphical User Interface (GUI). We use a mouse to operate it.

## Mouse

The mouse allows you to choose commands on menus and toolbars as well as select objects on the screen. As you move the mouse, a cursor moves on the screen. The cursor can take different shapes and is sometimes called an arrow of pointer. When you use a mouse, you first point the cursor to some object, then use the mouse button. Some mouse terms are:

- **Click**: press and release the left mouse button
- **Click and Drag**: press the left mouse button and hold it down while you move it over a text or an object. If an instruction says to drag, it assumes that you hold down the left mouse button while you are dragging.
- **Double-click**: quickly press and release the left mouse button twice
- **Right-click**: press and release the right mouse button
- **Waiting cursor**: when you see an hour-glass, it means you should wait while your computer completes a task.

Sometimes you will see an instruction like this: **Ctrl + click**. That means to hold down the **Ctrl** key while you click with the mouse. You can see a similar instruction with the **Shift** and **Alt** keys.

If you see a command that instructs to choose or select an item, it means that you should point and click.

## Introducing the Desktop

The Windows Desktop is a metaphor for your real-world desk, although it is more like an office-top than a desktop. The Desktop is highly customizable, so no two Windows Desktops look exactly the same. For example, you can see something very similar to the Desktop displayed in Fig. 1.7.

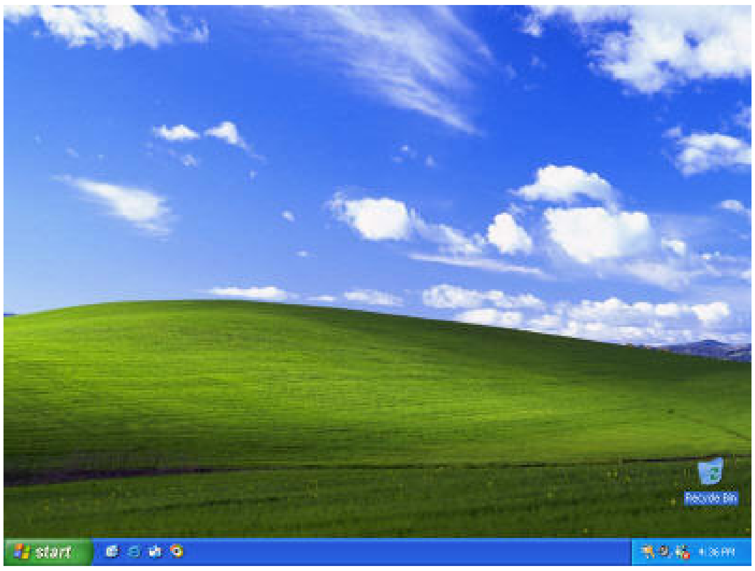


Fig. 1.7 Microsoft Windows XP Desktop

Figure 1.8 shows a Windows XP Desktop with the Start Menu open (you open it by clicking the "Start" button at the bottom left corner). Windows XP is the latest in the line of Microsoft operating systems which includes Windows 95, Windows 98, Windows 98 Second Edition, Windows Millennium and Windows 2000. If you are using any of Windows XP's predecessors your Desktop will look a little different but all the key features and functions will be present. Your desktop will have a **Taskbar** (access to all opened programs and files), **Start Button** (for opening files and running programs), **My Computer** (allows you to view folders and files), and a **Recycle Bin** (stores deleted files and allows you to retrieve them).

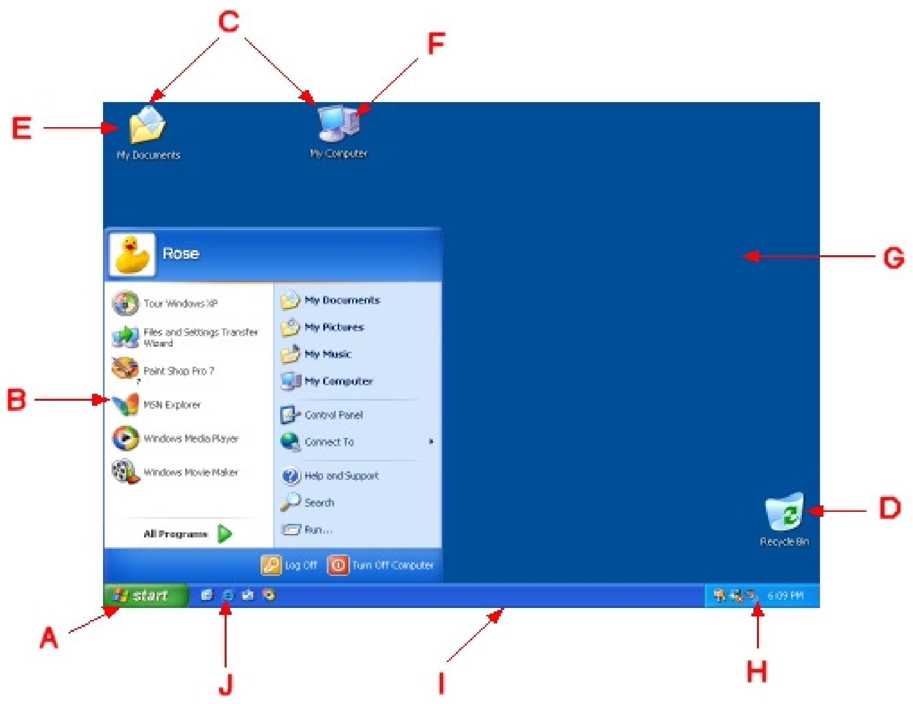


Fig. 1.8 Desktop with Start Menu open:
**A** – Start Button, **B** – The Start Menu, **C** – Icons, **D** – Recycle Bin,
**E** – My Documents, **F** – My Computer, **G** – Desktop,
**H** – Notification Area, **I** – The Taskbar, **J** – The Quick Launch Bar.

If you open several programs or windows, the buttons with their names appear on the Taskbar. You can click the mouse to toggle between them or press **Alt+Tab** keys.

## Shutting down

It is important you turn your computer off correctly. Simply hitting the power switch without shutting down properly is a sure way to lose documents you have been working on or even to damage data or programs on your system.

Here is how to exit from Windows:

1. **Close** any open programs or documents. Do so by clicking the little **X** in the top right-hand corner of each open window. If there appear, to be two Xs, one above the other in a window, click the topmost **X**.
2. Click the “**Start**” button and select **Turn Off Computer**. A Turn Off Computer dialog box will appear. Click “Turn Off”. A colored screen will appear indicating that Windows is shutting down. After a little while, your computer should shut off automatically.

## Files

The information in a computer is stored on a hard disk in **files**. To understand, what a file is, imagine a book. Books are kept on shelves or in a case. In a computer, files (books) are placed in folders (shelves or cases). Small folders can be inside larger folders. Thus, a tree of folders is formed. The most external folder is the **root**.

A **file name** consists of two parts: **name** and **extension**. The name consists of 250 or less characters: Latin, Russian, or Ukrainian letters, digits and some other symbols. It cannot contain symbols: «/», «\», «*», «,» «:» «?», « “», «<», «>», «|».

The extension shows what we can do with a particular file. For example:

Diary.txt – diary is the name, **txt** is the extension. Txt-files we can read, create (write) and edit. These files may only contain the simple text (non-formatted).

Letter.doc – has extension **doc**. This file can be opened with a special text editor – Microsoft Word or another other editor. This file we also can read, create and edit. Doc-files contain formatted text (colored text, bold etc.), pictures, tables, and other objects.

Game.exe has the extension **exe**. This file can be run. We cannot create or edit this file without special knowledge.

Windows divides all files on **executed** files, files with **data** and **system** files.

- An executed file is file which can be run.
- Data files contain text, tables, pictures, music, etc.
- System files contain special information and instructions for a system.

## Disks, paths and filenames

Disks are identified by a drive letter. On most computers the parts of a hard disk are **C:** and **D:**, the CD-ROM is E:. If your computer has additional drives these identifiers may slightly differ.

A folder path is a list of folders and sub-folders you must look through to find a file. The folder path of a file you place in the Correspondence sub-folder within the My Documents folder is:

\My Documents\Correspondence\

The backslashes are used to separate each element in the path.

(Actually, in Windows XP, the My Documents folder is a sub-folder within other folders, so the path is more likely to look something like this:

\Documents and Settings\username\My Documents\Correspondence

where username is your Windows log on name.)

A full filename takes the form:

driveletter:\folderpath\filename

For example:

C:\My Documents\My Pictures\Jumbo.jpg

Despite initial appearances, the two filenames are unique. If you make a copy of Jumbo.jpg on a floppy, its unique filename will be A:\Jumbo.jpg.

Notice that the Desktop, which you think of as your screen, is really just another sub-folder, stored within the Windows folder.

You can use "**My computer**" or "**Explorer**" to work with **files** and **folders**.

The "My computer" icon is on the Desktop. In the My computer window you can move from one folder to another by double-clicking.

To run "Explorer" you can right click on the "My computer" icon and choose "Explore". Explorer window consists of two parts. The left part is a tree of folders. If you choose a folder in the left part, the right part will show you consistence of this folder (files and folders).

You can change **views** of the right part. You have to option to view only file names (Tiles, Icons, List) or full information about files (Details): time and date of the last saving, size, type. You can **arrange** files by name, type, size, or a creation date.

## The basic operations with files and folders in Windows

**Running a file:**

- Choose a file to run (with mouse).
- Double click on it or press the "Enter" key.

**Making a Folder:**

- Find an open area (no icons below).
- **Right Click** and go to **New** – **Folder**.
- Once the folder is created, you will see that the New Folder's name is "New Folder" and it is selected or highlighted in Blue which means it is ready for you to type the name of your choice.
- You must type the folder's name and press "Enter".

**Creating a file:**

**Right-click** and go to **New**, and choose a type of new file. Then, type the name of the file. The file will be empty. To write something in it: double-click on the file (it will open). Type what you need and save changes (**File** – **Save**). If you do not want to save changes, close the file and press "No" – not to save.

**Copying Files:**

There are multiple ways to copy a file from one folder to another.

<u>Right-click method:</u>

1. **Right click** on the file.
2. Choose **Copy**. The file will be copied into the computer's memory.

3. Enter the folder where you want to paste the file.
4. Find an open area and right-click.
5. Go to **Paste**.

Edit-menus method:

1. Click on the file.
2. Go to **Edit** and choose **Copy**. The file will be copied into the computer's memory.
3. Enter the folder, where you want to paste the file.
4. Go to "Edit" and "Paste".

With keyboard:

1. Click on the file.
2. Press **Ctrl+C**. The file will be copied into the computer's memory.
3. Enter the folder where you want to paste the file.
4. Press **Ctrl+V**.

If you want to move a file instead of coping it, you must choose **Cut** instead of **Copy** or press **Ctrl+X** instead of **Ctrl+C**.

To **delete** a file or folder you must click on it. Then press "Delete" key or right-click on the file and choose **Delete**.

To **rename** a file or folder you must **right-click** on the file and choose **Rename**. Then type in a new name.

To undo an action you can press **Ctrl+Z** or go to **Edit** menu and choose **Undo**.

**Searching files**

1. Click "**Start**" button. Then **Search** to open the Search Companion.
2. Choose the appropriate type of search (if you are not sure what type of file you are trying to locate, select "All Files and Folders").
3. Type the search criteria into the boxes provided. The search criteria changes depending on the type of search. For example, if you are searching for multimedia files, you can specify whether you are looking for an audio, video or graphics. If you are searching for a document, you can specify the time frame in which it was created and enter any part of the filename which may help identify it.
4. Click **Search** to start the search.
5. When the search has completed, you can either refine the search or click "Yes, Finished Searching" to close the search panel and display a Task Pane.

**Selecting multiple files**

Often you will want to copy, move, delete, or open more than one file at a time. To do this, you need to select multiple files simultaneously. Here is how:

- Hold down the **Ctrl** key while you click each file you wish to select (this is called Ctrl-clicking).
- To deselect an already selected file Ctrl-click it.
- To select a whole list of files or folders, click the top file in the list, hold down the **Shift** key, and then click the last file in the list.

**Selecting files by corralling them**

For an even quicker method of selecting multiple files, you can click-and-drag a box around the files. Click in a vacant spot to the left of the first file icon you want to select and drag the mouse slightly downwards and to the right. Notice how, as you do so, a dotted outline of a box appears? This is called a selection rectangle.

You can use this same corralling technique in Windows Explorer, and even in Open and Save As dialog boxes within applications.

**Shortcut**

**Shortcut** is a special type of file. It contains the address of a file (the whole path and name of the executed file). When the shortcut is run, Windows passes the file its address and runs it. It is convenient to have shortcuts for files that often used. To create a shortcut on the Desktop you must **right-click** on the file and choose **Send to Desktop**. To create a shortcut in a current folder, you must **right-click** on the file and choose **Create Shortcut**.

## *Microsoft Word Text Processor*

To start up Word click the "**Start**" button, **Programs** and then **Microsoft Word**.

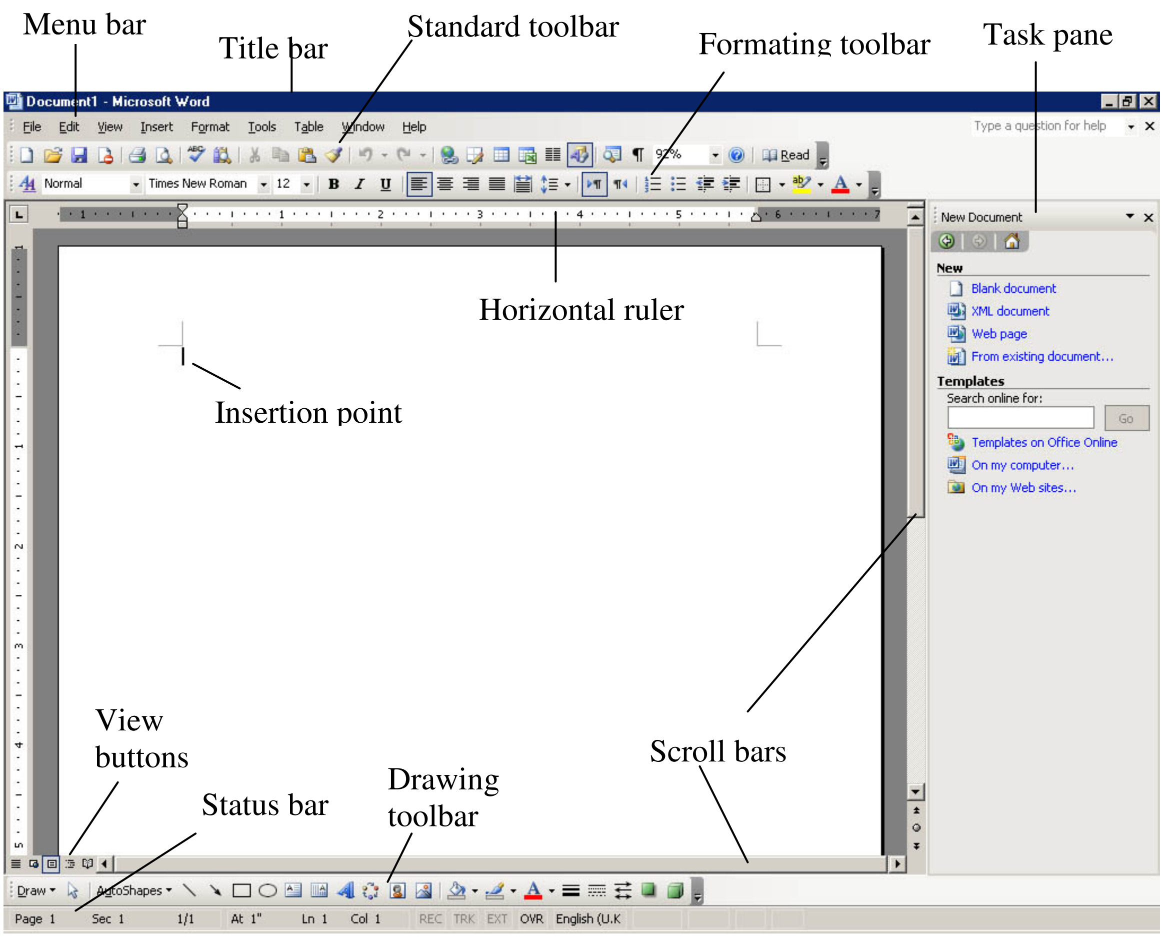


Fig. 1.9 Screen with Microsoft Word window

Once started up, Word automatically loads a new document – Document1 as identified in the title bar. This is a temporary area in which you enter your text. Once you have entered data that you do not want to lose, you can **save** it as a file. Your screen should look the same as the one shown in fig. 1.9, i.e. the Standard and

Formatting toolbars and the task pane are displayed. The ruler, status bar, and the scroll bars are all visible.

Commands in Word are accessed through the task pane, menus, toolbars and keyboard shortcuts.

Commands may be accessed by choosing an option from the menu bar at the top of the Word screen. By clicking on an option, the user is often faced with a second selection from either a sub-menu or a dialog box before the command can be carried out.

Toolbars provide a shortcut to many commands (fig. 1.10). Using the mouse, point and click on the required button. Different toolbars can be displayed and hidden at different times. In Application 1 you can see the description of the Standard and Formatting toolbars.

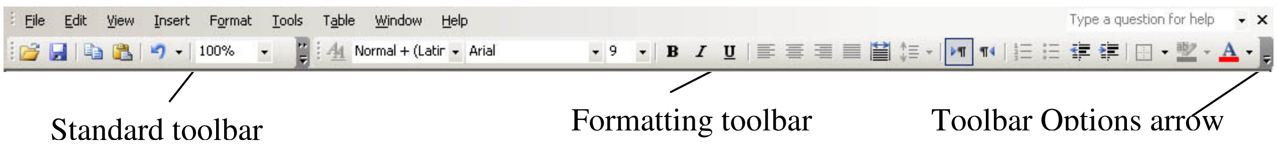


Fig. 1.10 Microsoft Word toolbars

The **insertion point** flashes in the document area of the screen which indicates where the text you type will appear. In Word 2003 you can move the insertion point to anywhere in your document by double-clicking at the point where you would like it to go.

Word, like most word processing packages, incorporates word-wrapping. This means that you should not press the “Enter” key ↵ at the end of each line. Only press “Enter” when you wish to end a paragraph or leave a blank line.

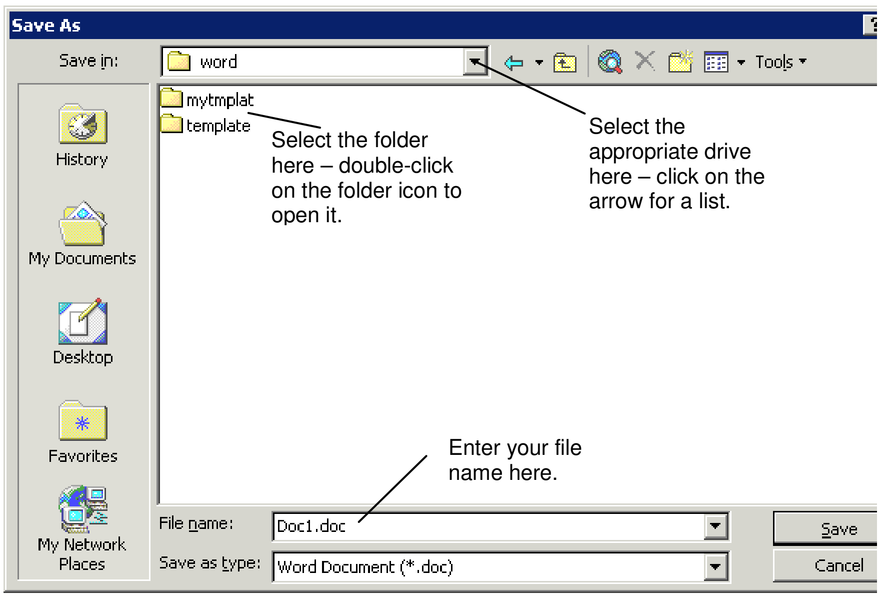
To correct simple typing mistakes either press the “Backspace” key ← to delete to the left of the insertion point or the “Delete” key to delete to the right of the insertion point.

As you type, your work is held in the computer’s temporary memory. If the computer is switched off or develops a fault, you may lose work that is not saved. Always save your work regularly.

When saving a document for the first time, Word displays the “Save As” dialog box (fig. 1.11). When prompted to name the file, name it, and select a folder and drive to place the file. To save the file:

1. In the **File** menu, click **Save**, or press **Ctrl+S**. The Save As dialog box appears (fig. 1.11).
2. Choose the appropriate drive and folder, using the drop-down arrow in the “Save in” box. When you save a file for the first time, check where it is stored.
3. Name the file. Word automatically adds the file extension ***.doc***. It is important to retain this default as, by international convention, any file with the extension ***.doc*** will be recognised by Windows as a Word document and associated with Word.
4. Click **Save**.

Once the document has been saved, the file name will appear in the title bar at the top of the screen. To re-save a document quickly with its existing name, location and file format, click “Save” on the Standard toolbar, or press **Ctrl+S**.



Select the folder here – double-click on the folder icon to open it.

Select the appropriate drive here – click on the arrow for a list.

Enter your file name here.

File name: Doc1.doc

Save as type: Word Document (*.doc)

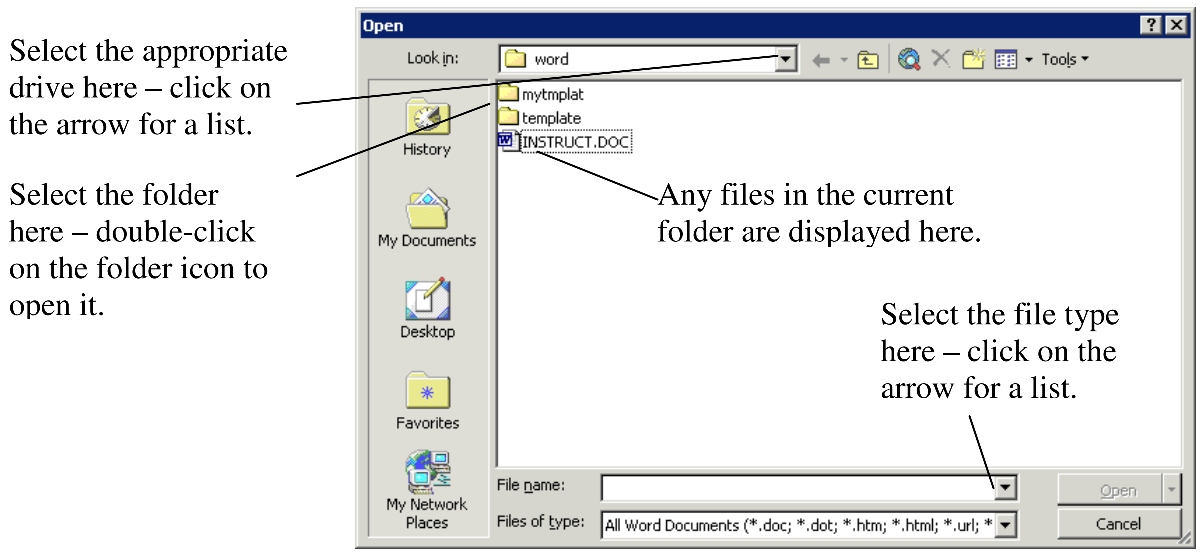
Fig. 1.11 Save As dialog box

The "Save As" command can be used when you do not wish to overwrite the old file, but wish to save your work in a new file with a different name, in a different location, or in a different format:

1. From the **File** menu click **Save As**. The Save As dialog box appears (fig. 1.11).
2. If necessary, select a different drive and folder or change the type of file.
3. Change the file name.
4. Click on **Save**.

## Opening a document

1. In the **File** menu click **Open**, press **Ctrl+O**, or click **Open** icon on the toolbar: . You will see the Open dialog box (fig. 1.12).



Select the appropriate drive here – click on the arrow for a list.

Select the folder here – double-click on the folder icon to open it.

Any files in the current folder are displayed here.

Select the file type here – click on the arrow for a list.

File name:

Files of type: All Word Documents (*.doc; *.dot; *.htm; *.html; *.url; *

Fig. 1.12 Open file dialog box

2. Choose the appropriate drive and folder.
3. Select a file from the files listed and click "OK".

The default for Word is to open files with a ***.doc*** extension. If your document has any other extension you will need to alter the **Files of type** option.

Before you can move, edit, delete, format or otherwise change any text or graphic in the document you must select (or highlight) the item first. To select text with mouse:

1. Position the insertion point at the start of the text.
2. Left-click and drag over the area to be selected.
3. Release the mouse button when you have selected the desired text.

These shortcuts will help you work faster in any Word document:

| **To highlight** | **Method** |
|---|---|
| A single word | Double-click on the word |
| A line of text | Click in the selection bar to the left of the line |
| Multiple lines of text | Drag in the selection bar to the left of the lines |
| A sentence | Hold down Ctrl and click anywhere in the sentence |
| A paragraph | Double-click in the selection bar to the left of the paragraph or triple-click anywhere in the paragraph |
| An entire document | Triple-click in the selection bar (or Ctrl+A) |
| Individual characters | Shift+← or Shift+→ |
| Non-adjacent text | Select the first piece of text then hold down the Ctrl key while you select the next bit of text, and so on (New in Word 2003) |

## Copying a text

- Select the text to be copied.
- In the **Edit** menu, click **Copy**, press **Ctrl+C**, or click on the **Copy** icon:
- Move the insertion point to the new location.
- In the **Edit** menu, click **Paste**, press **Ctrl+V**, or click on the **Paste** icon:

## Moving a text

- Select the text to be moved.
- In the **Edit** menu, click **Cut**, press **Ctrl+X**, or click on the **Cut** icon:
- Move the insertion point to the new location.
- In the **Edit** menu, click **Paste**, press **Ctrl+V**, or click on the **Paste** icon:

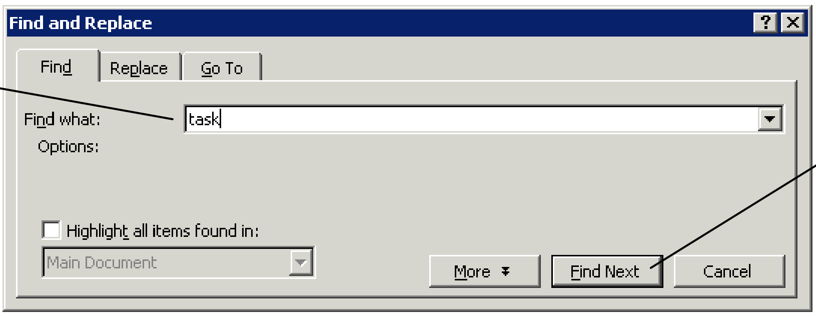
## Searching a word

- Position the insertion point at the beginning of the text. In the **Edit** menu, click **Find** or press **Ctrl+F**. The **Find and Replace** dialog box appears with the Find tab on top (fig. 1.13).
- Enter the word you wish to find in the **Find what** box.
- Click on the "Find Next" button to start the search.

Microsoft Word begins its search based on where your cursor is positioned in the document.

Word also lets you conduct expanded searches for such things as matching case, whole words, wildcards, sounds like, all word forms, character formats, and special punctuation. To access these features, click the More button in the Find and Replace window.

Enter the word you wish to find.



Click on “Find Next” to start the search.

Fig. 1.13 Find and Replace dialog box

**Replacing a word**

- Position the insertion point at the start of the text.
- In the **Edit** menu, click **Replace**, or press **Ctrl+H**.
- Enter the text you wish to replace in the **Find what** box.
- Enter the replacement text in the **Replace with** text box.
- Click the **Find Next** button to find the next instance of the word in the text.
- Click the **Replace** button to replace this instance of the word found in the text.
- Click the **Replace All** button to replace all instances of the word in the text.

Be careful when using Replace All. You might change things you don't mean to change. (For example, if you didn't choose the "Find whole words only" option, and changed "Smith" to "Jones," you would find that "Smithers" had changed to "Jonesers" - probably not what you intended.)

**Formatting a text**

Writing in Microsoft Word is more than just typing. Using MS Word also consists of formatting your document. There are several methods to edit and format your document. You can Copy, Paste, Bold, Italic, and Underline words or sentences. You can Align text to the right, center, or left. Among others, you may also add Bullets and Numbers to the content.

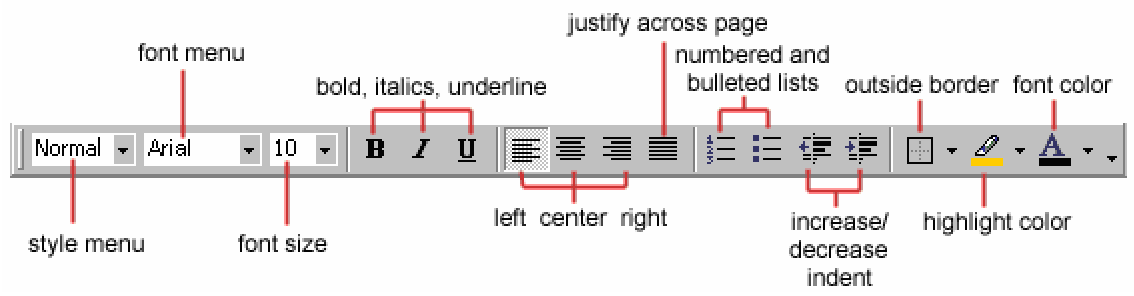


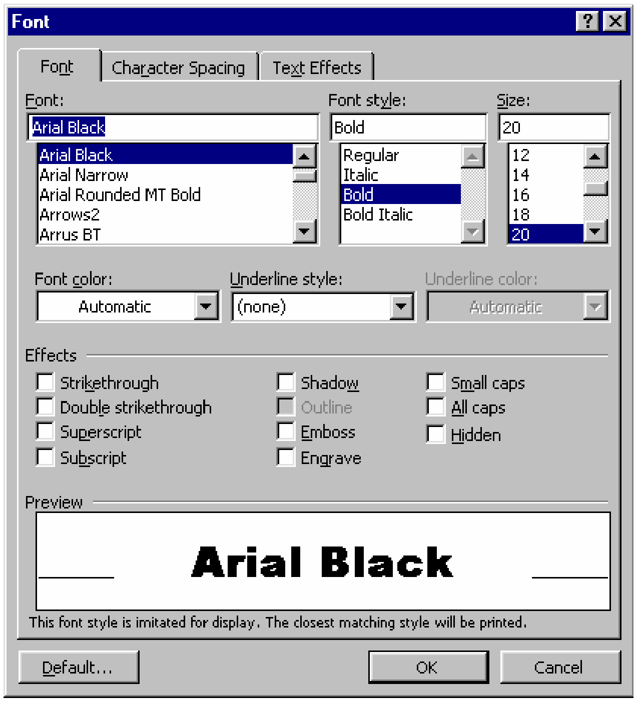
Fig. 1.14 Format toolbar

When you begin typing a new document, the text appears in the font (typeface) and font size (measured in points) that are predefined for Word. To achieve a different look, you can choose other formats for Word to use. You can apply formats by clicking appropriate buttons on the Formatting toolbar (fig. 1.14), by pressing shortcut keys or by clicking **Font** in the **Format** menu.

The Formatting toolbar contains the most commonly applied formats. However, there are more options available in the Font dialog box.

To access the Font dialog box:

- Select the text to format.
- In the **Format** menu select **Font**. The Font dialog box will appear (fig. 1.15).



Font

Font | Character Spacing | Text Effects

Font: Arial Black
Font style: Bold
Size: 20

Arial Black, Arial Narrow, Arial Rounded MT Bold, Arrows2, Arrus BT
Regular, Italic, Bold, Bold Italic
12, 14, 16, 18, 20

Font color: Automatic
Underline style: (none)
Underline color: Automatic

Effects

☐ Strikethrough ☐ Shadow ☐ Small caps
☐ Double strikethrough ☐ Outline ☐ All caps
☐ Superscript ☐ Emboss ☐ Hidden
☐ Subscript ☐ Engrave

Preview

Arial Black

This font style is imitated for display. The closest matching style will be printed.

Default... OK Cancel

Fig. 1.15 Font dialog box

### Align a text

You may want to center a title, put your address on the right side of the screen, keep your text aligned on the left, or justify your type across the entire line.

**Align Right** means the text will be flush with the documents right border.

**Align Left** means the text will be flush with the documents left border.

**Align Center** means the text will be centered across the document.

**Justify** means the characters will spread out across the line to fill the space.

- Select the text content you want to align.
- Click the icon on the Formatting Toolbar (fig. 1.16) to modify the text alignment.

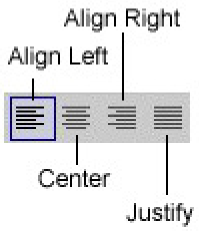
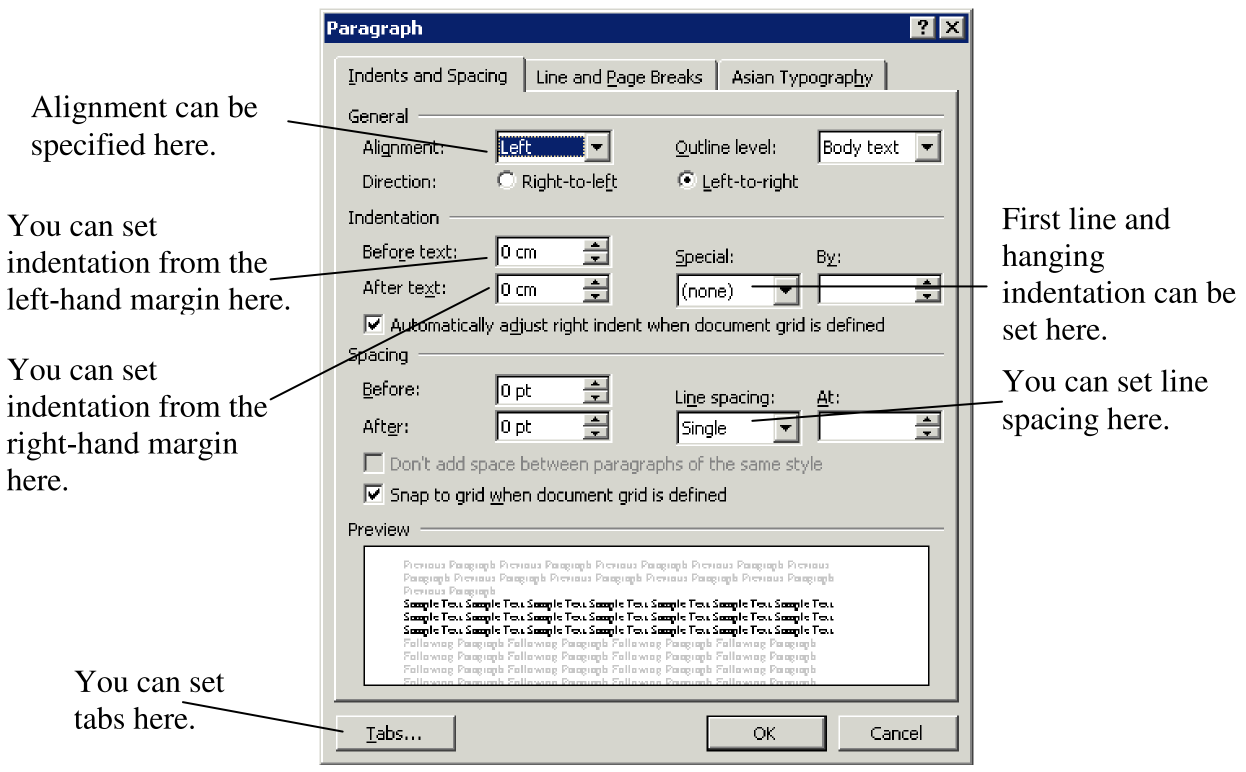


Fig. 1.16 Align text icons

### Formatting paragraphs

Some formatting options within Word apply to paragraphs, such as, alignment, line spacing, borders and shading, and indentation and tabs. A paragraph may consist of several lines of a text, a single word, or no text at all. A paragraph is created every time the “Enter” key is pressed. The end of each paragraph is defined by a paragraph mark: ¶.

Paragraph formats can be applied in a number of different ways: in the **Format** menu, the **Formatting** toolbar, the ruler, shortcut keys, or the **Reveal Formatting** task pane. To make several different types of changes to the paragraph(s) at the same time, you can click the **Paragraph** option in the **Format** menu. You can determine the indentation, alignment, line spacing, and tabs in the Paragraph dialog box as shown in fig. 1.17.



Alignment can be specified here.

You can set indentation from the left-hand margin here.

You can set indentation from the right-hand margin here.

You can set tabs here.

First line and hanging indentation can be set here.

You can set line spacing here.

Paragraph

Indents and Spacing | Line and Page Breaks | Asian Typography

General

Alignment: Left

Outline level: Body text

Direction: ( ) Right-to-left [✓ Left-to-right]

Indentation

Before text: 0 cm

After text: 0 cm

Special: (none)

By:

☑ Automatically adjust right indent when document grid is defined

Spacing

Before: 0 pt

After: 0 pt

Line spacing: Single

At:

☐ Don't add space between paragraphs of the same style

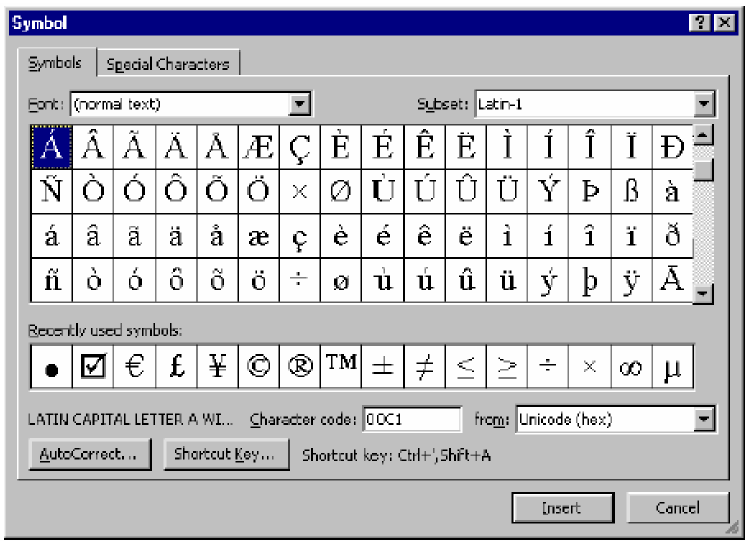
☑ Snap to grid when document grid is defined

Preview

Fig. 1.17 Paragraph dialog box

## Inserting non-English characters one-by-one

To insert non-English characters in your document, in the **Insert** menu, select **Symbol** (fig. 1.18).



Symbol

Symbols | Special Characters

Font: (normal text)

Subset: Latin-1

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï | Ð |
| Ñ | Ò | Ó | Ô | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß | à |
| á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï | ð |
| ñ | ò | ó | ô | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ | Ā |

Recently used symbols:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ● | ☑ | € | £ | ¥ | © | ® | ™ | ± | ≠ | ≤ | ≥ | ÷ | × | ∞ | µ |

LATIN CAPITAL LETTER A WI... Character code: 00C1 from: Unicode (hex)

Shortcut key: Ctrl+',Shift+A

Fig. 1.18 Symbol dialog box

In **Font** box, selecting **Normal** text will give you access to a range of characters in the same font that you are currently using.

Scroll slowly down to find the character you are looking for. The full range of characters for Western European languages will always be available. Depending on which font you are working with, you may also find other character sets - for example Greek, Hebrew, and Arabic.

If the characters you want are not available in your normal text font, select Arial Unicode. This has an extensive choice available from the Subset list. It includes character sets from all the main languages (for example Chinese, Arabic, Japanese).

## Using Column Layout

You will not see columns in their proper position unless View is set to Print Layout. In the Toolbar, click **Columns** (fig. 1.19), and then click on the number of columns you need.

To start a new column at a fixed point in the **Insert** menu, select **Break** and then **Column Break**.

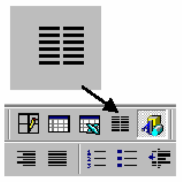
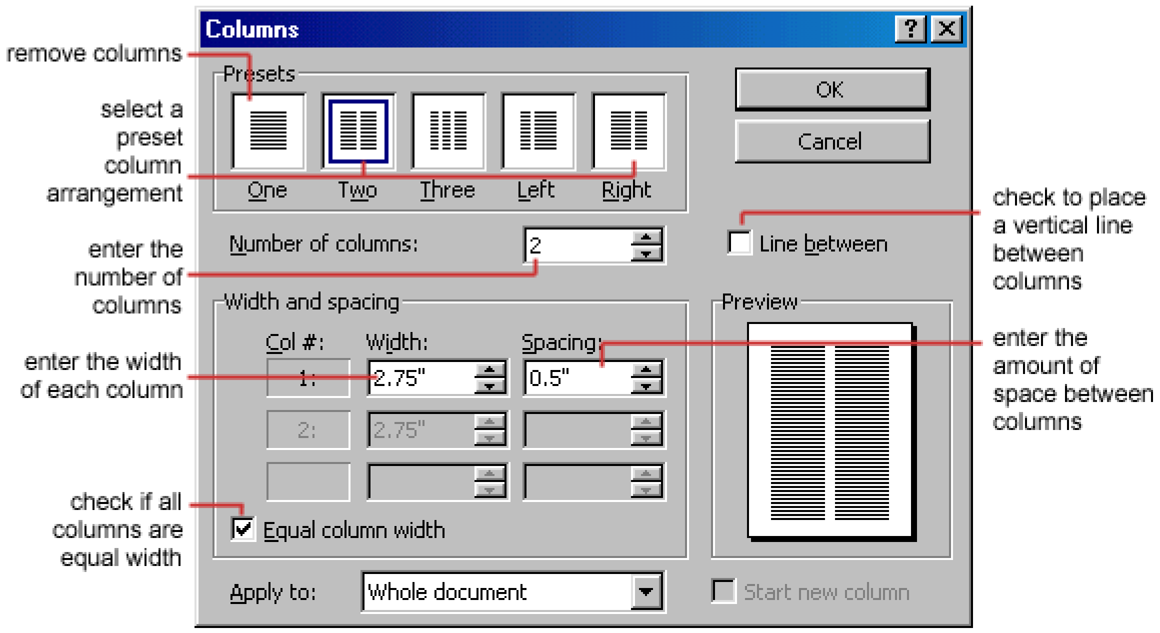


Fig. 1.19
Columns button

For a more sophisticated column formatting select **Columns** in the **Format** menu (fig. 1.20). If you want uneven column widths: switch off the "Equal Column Width" checkbox, then set "Width and Spacing". Notice the preset column format options that are available. Alter "Apply To" if necessary. A section break will be added automatically if you set it as "From this Point".



remove columns
select a preset column arrangement
enter the number of columns
enter the width of each column
check if all columns are equal width
check to place a vertical line between columns
enter the amount of space between columns
Columns
Presets
One
Two
Three
Left
Right
OK
Cancel
Preview

Number of columns: 2
☐ Line between
Width and spacing

| Col #: | Width: | Spacing: |
|---|---|---|
| 1: | 2.75" | 0.5" |
| 2: | 2.75" | |
| | | |

☑ Equal column width
Apply to: Whole document
☐ Start new column

Fig. 1.20 Columns dialog box

## Drop Caps

A drop cap is a large letter that begins a paragraph and drops through several lines of text as shown in fig. 1.21.

To add a drop cap to a paragraph follow these steps:

- Place the cursor within the paragraph whose first letter will be dropped.
- Select **Format|Drop Cap** from the menu bar.
- The **Drop Cap** dialog box allows you to select the position of the drop cap, the font, the number of lines to drop, and the distance from the body text.
- Click **OK** when all selections have been made.
- To modify a drop cap, select **Format|Drop Cap** again to change the attributes, or click on the letter and use the handles to move and resize the letter.

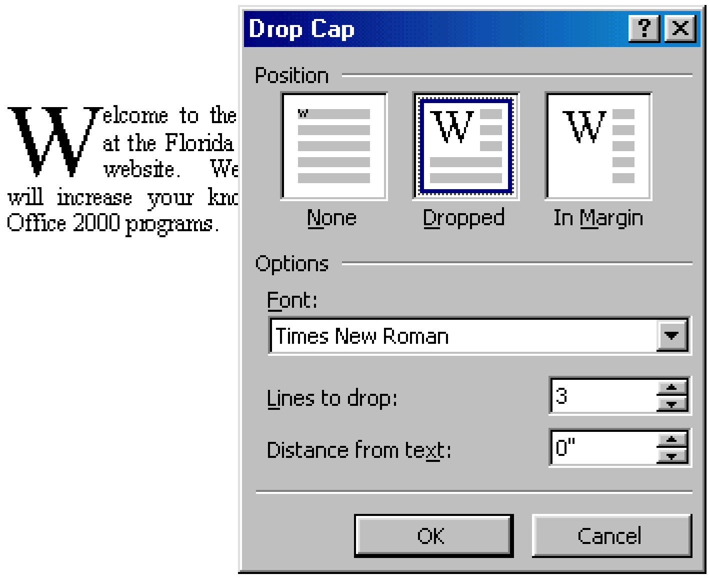


Fig. 1.21 Drop Cap

## Bullets and Numbering

Lists provide a great way to present a lot of information in an easy-to-understand format. Two types of lists are commonly used: **bulleted** and **numbered**.

There are two ways of inserting bullets or numbering. You can click the icon (fig. 1.22) from the **Formatting Toolbar** first and then type the text content in the document window. Every time you press "Enter", the bullets or numbering setting will be applied to that sentence. Or you can type the text content in the document window first, highlight the content, and then click the icon from the Formatting Toolbar.

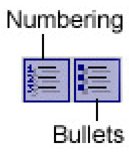


Fig. 1.22 Numbering and bullets icons

You can apply different styles of Bullets and Numbering by going to **Format – Bullets and Numbering**. When the Bullets and Numbering window opens (fig. 1.23), choose a style to apply in your document.

On the Numbering tab, you can choose to restart numbering if this is a new list, or continue numbering if numbers in your list are separated by text or graphics.

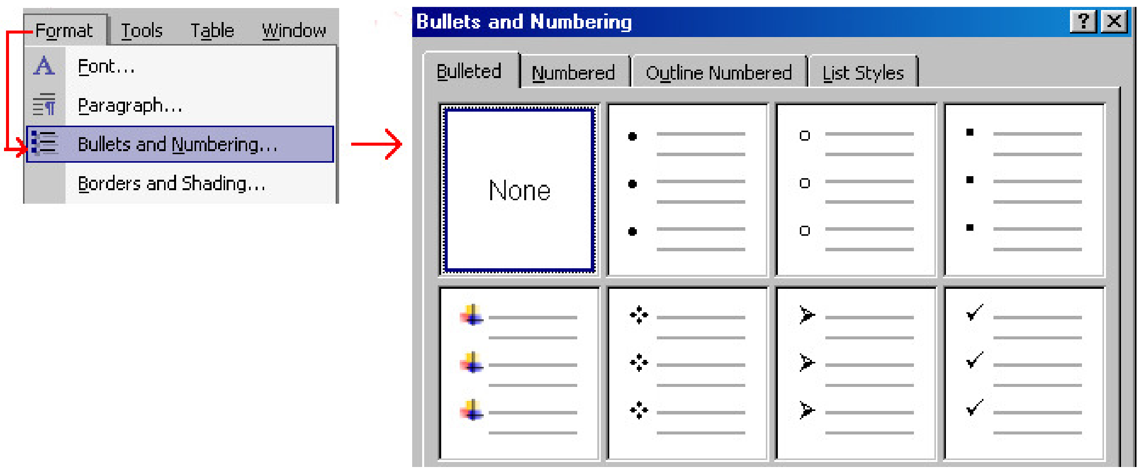
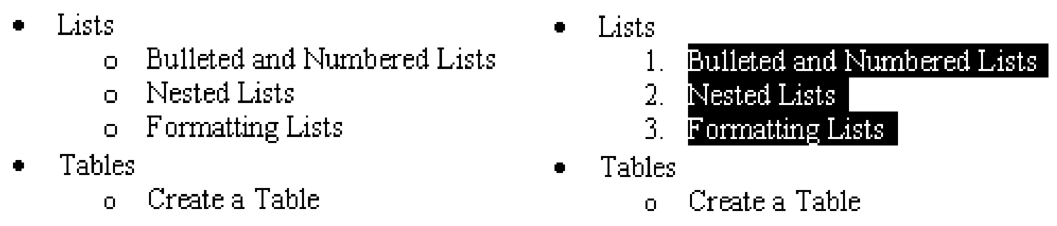


Fig. 1.23 Bullets and Numbering dialog box

## Nested Lists

To create a nested list, such as a numbered list inside of a bulleted list, follow these steps:

▪ Type the list and increase the indentation of the items that will make up the nested list by clicking the **Increase Indent** button for each item.



- Lists
  - o Bulleted and Numbered Lists
  - o Nested Lists
  - o Formatting Lists
- Tables
  - o Create a Table

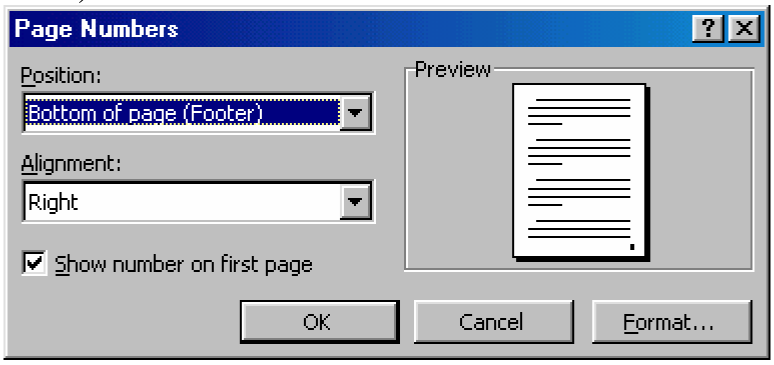
- Lists
  1. Bulleted and Numbered Lists
  2. Nested Lists
  3. Formatting Lists
- Tables
  - o Create a Table

▪ Highlight the items and click the **Numbered List** button on the formatting toolbar.

## Headers and footers

Do not confuse "Headers" with "Headings". **Headings** are formatted within the main text of your document. **Headers and footers** are created separately on the top and bottom margins respectively and appear on every page. They may contain document title, chapter titles, page number, number of pages, date of creation, creator, date last saved, filename, or pathname.

If simple page numbering is only required, go to the **Insert** menu and select **Page Numbers** (fig. 1.24).



Page Numbers

Position: Bottom of page (Footer)

Alignment: Right

[✓] Show number on first page

Fig. 1.24 Page Numbers dialog box

Adding **headers** or **footers**:

- In the **View** menu, select **Header/Footer**. The Header and Footer window and toolbar will appear (fig. 1.25).
- Move the cursor slowly over the toolbar buttons to see their functions. You can insert page numbers and the date from here. More useful options are available from **Insert Autotext** – these include "Page X of Y", and File Name. You can include any combination of items, along with your own text.
- Switch between headers and footers with the third button from the right. The other buttons on the right relate to document sections. If you want different headers and footers for different sections of your document, close the Header and Footer Toolbar, and divide your document into sections. Format items in a header or footer just as you would within the main document text.

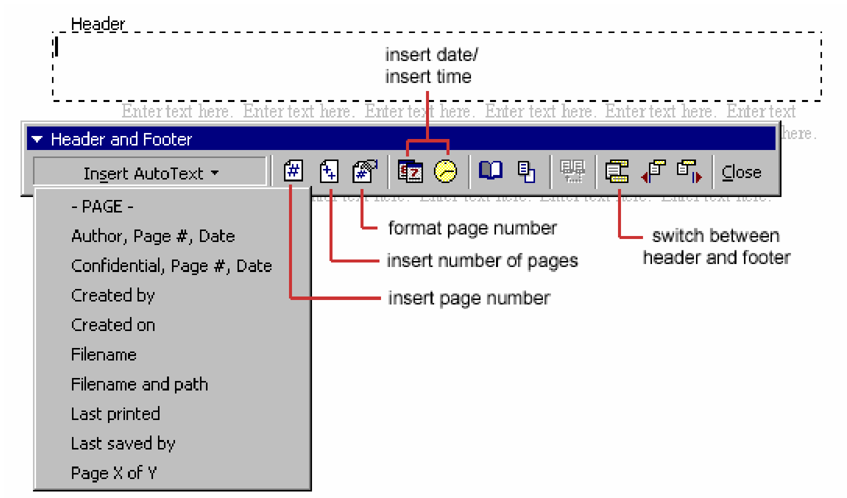


Fig. 1.25 Header and Footer window

If you want headers and footers positioned differently for odd and even pages, or on your first page, click on "Page Setup" in the Header and Footer Toolbar. The Page Setup dialog box will be displayed, with the Layout tab selected. Check the appropriate Header and Footer options.

**Paper size**

The page margins of the document can be changed using the rulers on the page (fig. 1.26) and the **Page Setup** window. The ruler method is discussed first:

- Move the mouse over the area where the white ruler changes to gray.

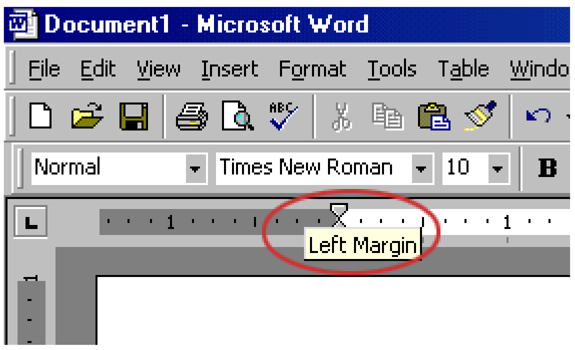
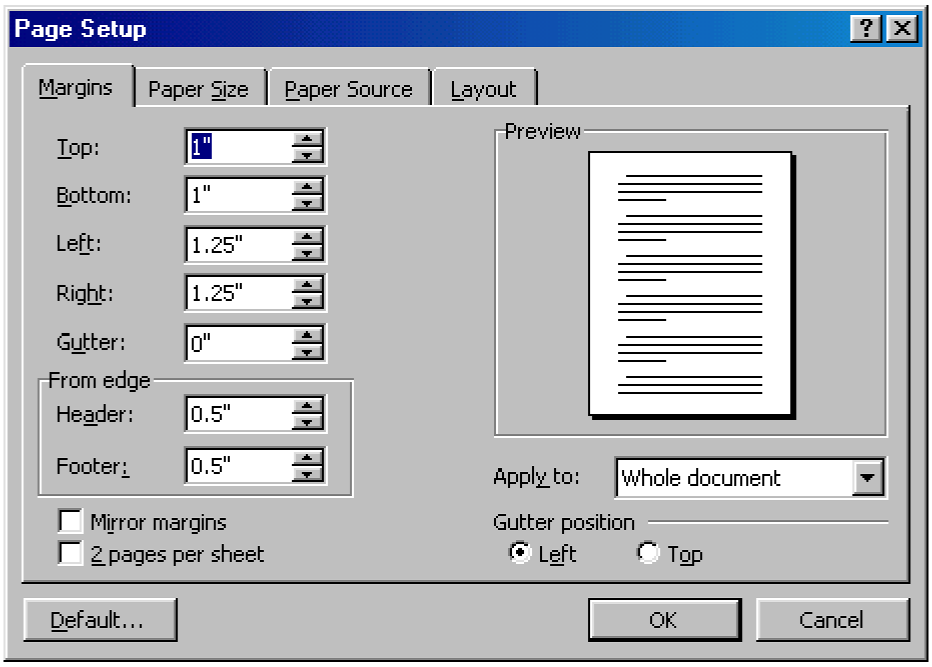

Document1 - Microsoft Word
File Edit View Insert Format Tools Table Windo
Normal
Times New Roman
10
B
Left Margin

Fig. 1.26 Left page margin on the horizontal ruler

- When the cursor becomes a double-ended arrow, click with the mouse and drag the margin indicator to the desired location.
- Release the mouse when the margin is set.

The margins can also be changed using the **Page Setup** dialog box:

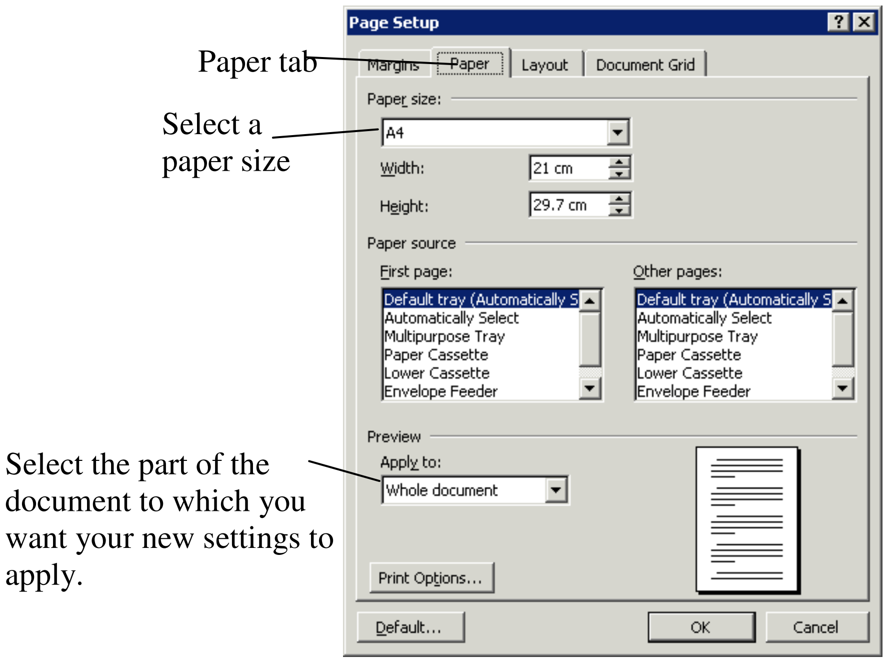
- In the **File** menu, click **Page Setup**. The Page Setup dialog box will appear. Click the **Margins tab** in the dialog box (fig. 1.27).
- Enter margin values in the **Top**, **Bottom**, **Left**, and **Right** boxes. The **Preview** window will reflect the changes.
- If the document has **Headers** and/or **Footers**, the distance this text appears from the edge of the page can be changed.
- Click **OK** when finished.



Top: 1"
Bottom: 1"
Left: 1.25"
Right: 1.25"
Gutter: 0"
Header: 0.5"
Footer: 0.5"
Mirror margins
2 pages per sheet
Apply to: Whole document
Gutter position: [✓ Left] Top

Fig. 1.27 Page Setup dialog box. Margins tab

Click the **Paper** tab to change the orientation of the page (fig. 1.28). In the **Paper Size** list select an appropriate paper size e.g. **A4** and click **OK**.



Select the part of the document to which you want your new settings to apply.

Paper size: A4
Width: 21 cm
Height: 29.7 cm
First page: Default tray (Automatically S, Automatically Select, Multipurpose Tray, Paper Cassette, Lower Cassette, Envelope Feeder
Other pages: Default tray (Automatically S, Automatically Select, Multipurpose Tray, Paper Cassette, Lower Cassette, Envelope Feeder
Apply to: Whole document

Fig. 1.28 Page setup dialog box

**Styles and Headings**

To allow Microsoft Word to generate a table of contents based on your text, you must use headings.

A **style** is a set of formatting characteristics that you can give to text. Word contains many built-in styles, such as Title, Heading 1, Normal, Body Text, and along with others.

When you want a top-level heading, do not just highlight the text and then format it to give the appearance you want. Instead, apply the **Title style**. For all major headings use the Heading 1 style. For subheadings or sections use style Heading 2 and so on. You can see how the style name for the headings reflects the logical structure of the document. To apply a style put the cursor in the text and then select the style from the Styles list (as shown in fig. 1.29). Alternatively, use **Format** – **Style** and select the style from the list that appears.

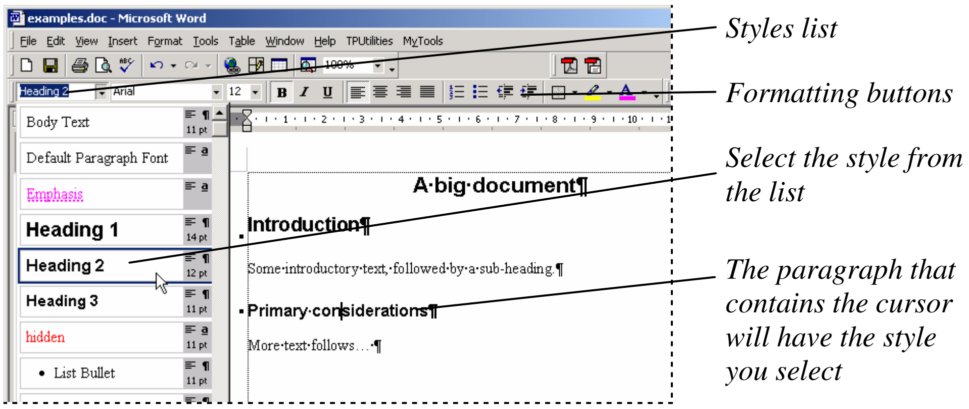

examples.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help TPUtilities MyTools
Heading 2
Arial
12
Body Text
Default Paragraph Font
Emphasis
Heading 1
Heading 2
Heading 3
hidden
List Bullet
A·big·document¶
Introduction¶
Some·introductory·text,·followed·by·a·sub-heading.¶
Primary·considerations¶
More·text·follows…·¶
*Styles list*
*Formatting buttons*
*Select the style from the list*
*The paragraph that contains the cursor will have the style you select*

Fig. 1.29 Select the style from the Styles list

Now, you may not like the appearance of your headings and subheadings. The great thing is that you can change all of them quickly. For example, you might want all your chapter headings to be 15pt (that is just less than about 1½ times the height of the usual text size), dark blue, bold and centered on the page.

To change the appearance of a particular style choose **Format** – **Style**. The Style box will appear. Highlight, for example, “Heading 1” and click **Modify**. Then, the Modify Style box will appear. Click **Format** – **Font** and change the Font color (blue) and the Size (15pt) option.

Click “OK” to return to the Modify Style box. Click **Format** – **Paragraph**. Change, for example, the Alignment to the Center. Then click “OK” repeatedly to close all the boxes. Now, all your top-level headings are centered, blue, and 15pt. When you make a single change, every instance of a style is changed.

## Automatic table of contents

With a large document you often want a table of contents that shows chapters and subheadings with corresponding page numbers. Now, you can do this the tedious way, by finding every instance of a heading and making a note of the page number. Then you can type out the headings and their corresponding page numbers. However, what do you think happens when you insert a few pages in the middle of the document? You have to remember to modify your table of contents. If you have used styles, Word can automatically generate a table of contents for you. If page numbers of headings change because you add or delete text, or if you change the heading text, all you need to do is press one key, and Word automatically re-generates a new table of contents.

You can insert a table of contents anywhere in your document; it need not be at the front. Although this is usually best to place it in the beginning.

To create a table of contents place the cursor where you want the table of contents to appear. Choose **Insert – Index and Tables**. The Index and Tables box appears (fig. 1.30). Select the **Table of Contents** tab. Click “OK”.

Word creates a table of contents from your headings and subheadings. You will have noticed that there are various options, such as whether or not you want the page number to be included. You can of course use these to specify the appearance of your table of contents.

If you click on a heading or page number in the table of contents, Word jumps to that page.

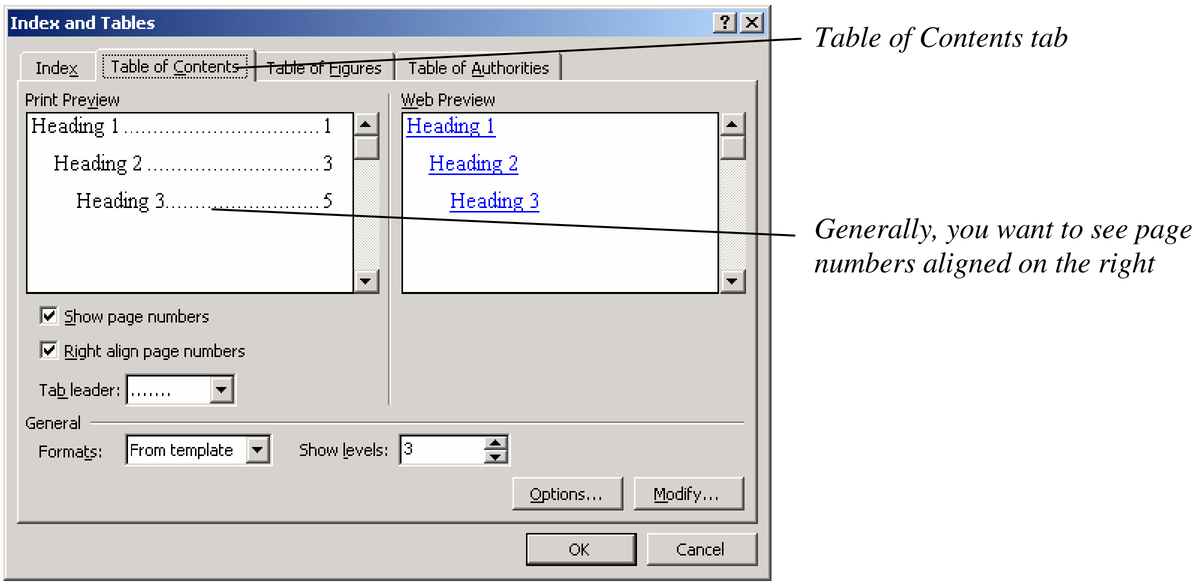


Table of Contents tab
Generally, you want to see page numbers aligned on the right

Index and Tables

Index | Table of Contents | Table of Figures | Table of Authorities

Print Preview

Heading 1 .................................. 1
Heading 2 ............................ 3
Heading 3 ..................... 5

Web Preview

Heading 1
Heading 2
Heading 3

[✓] Show page numbers
[✓] Right align page numbers
Tab leader: .......

General
Formats: From template    Show levels: 3

Options... Modify...
OK Cancel

Fig. 1.30 Index and Tables dialog box

## Format Painter

A handy feature for formatting text is the **Format Painter** located on the standard toolbar. For example, if you have formatting a paragraph heading with a certain font face, size, and style and you want to format another heading the same way, you do not need to manually add each attribute to the new headline. Instead, use the Format Painter by following these steps:

- Place the cursor within the text that contains the formatting you want to copy.
- Click the **Format Painter** button in the standard toolbar. Notice that your pointer now has a paintbrush beside it.
- Highlight the text you want to add the same format to with the mouse and release the mouse button.

To add the formatting to multiple selections of text, double-click the **Format Painter** button instead of clicking once. The format painter then stays active until you press the **ESC** key to turn it off.

## Tables

Tables are used to format all or part of the document into columns and rows.

There are several ways to build a table in Word. Begin by placing the cursor where you want the table to appear in the document and choose one of the methods. The most common method is: select **Table** – **Insert** – **Table** from the menu bar. Select the number of rows and columns for the table (fig. 1.31) and click "OK".

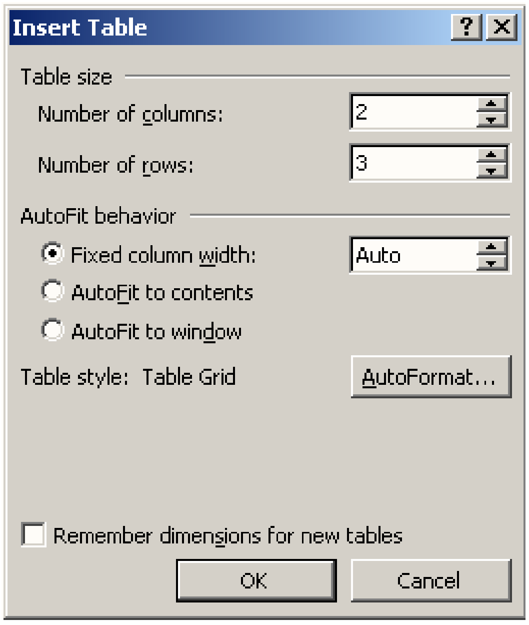

Insert Table

Table size
Number of columns: 2
Number of rows: 3

AutoFit behavior
(●) Fixed column width: Auto
( ) AutoFit to contents
( ) AutoFit to window

Table style: Table Grid
AutoFormat...

☐ Remember dimensions for new tables

OK Cancel

Fig. 1.31 Insert Table dialog box

Each block in a table is called a **cell**. Use the Tab key to move from cell to cell from left to right. Use Shift-Tab to move from one cell to another cell from right to left. You can also move to a cell by clicking in the cell. In addition, you can move around the table by using the left, right, up, and down arrow keys.

In each cell there's a sign like¤. That's the end-of-table-cell marker. It is analogous to ¶, which is the end-of-paragraph marker. (If you don't see the ¤, press the ¶ button on the Standard Toolbar.)

By default, the table is positioned just left of the left margin, and stretches to just right of the right margin. Word's default puts the table on the page so that text in the left column of the table will line up with text outside a table.

There is always a paragraph after a table. Even if the table is the last thing in the document, there will be a paragraph after it, and you can't delete that last paragraph mark.

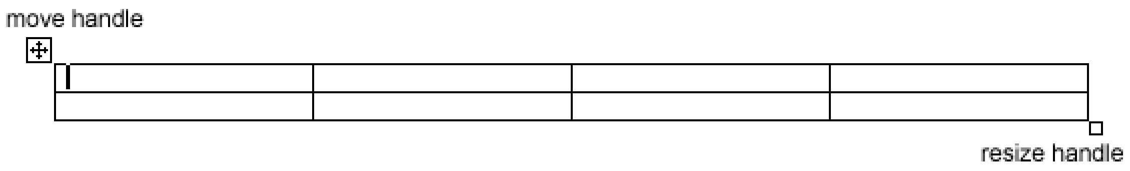
### Inserting Rows and Columns

Once the table is drawn, insert additional rows by placing the cursor in the row you want to be adjacent to. Select **Table** – **Insert** – **Rows Above** or **Rows Below**, or select an entire row and **right-click** with the mouse. Choose **Insert Rows** from the shortcut menu.

Much like inserting a row, add a new column by placing the cursor in a cell adjacent to where the new column will be added. Select **Table** – **Insert** – **Columns to the Left** or **Columns to the Right**. You may also select a column, **right-click** with the mouse, and select **Insert Columns**.

### Moving and Resizing a Table

A four-sided moving arrow and open box resizing handle will appear on the corners of the table if the mouse is placed over the table. Click and drag the four-ended arrow to move the table and release the mouse button when the table is positioned where you want it. Click and drag the open box handle to resize the table. Change the column widths and row heights by clicking the cell dividers and dragging them with the mouse.



## Tables and Borders Toolbar

The Tables and Borders toolbar allows you to add border styles, shading, text effects, alignment, and more options to your table. Access the toolbar by clicking **Table** – **Draw Table** or **View** – **Toolbars** – **Tables and Borders**.

You will need to highlight the cells of the table you want to format. Click and drag the mouse over the cells, or use the following shortcuts:

| Selection | Menu Method | Mouse Method |
|---|---|---|
| One cell | **Table – Select – Cell** | Click the bottom, left corner of the cell when a black arrow appears |
| One row | **Table – Select – Row** | Click outside the table to the left of the row |
| One column | **Table – Select – Column** | Click outside the table above the column when a black arrow appears |
| Several rows | **(none)** | Click outside the table to the left of the row and drag the mouse down |
| Several columns | **(none)** | Click outside the table above the column |
| Entire table | **Table – Select – Table** | Triple-click to the left of the table |

## Table Properties

The **Table Properties** dialog box (fig. 1.32) is used to modify the alignment of the table with text and the text within a table. Access the box by selecting **Tables** – **Table Properties**.

- **Size –** Check the **Preferred width** box and enter a value if the table should be an exact width.
- **Alignment –** Highlight the illustration that represents the alignment of the table in relation to the text of the document.
- **Text wrapping –** Highlight "None" if the table should appear on a separate line from the text or choose "Around" if the text should wrap around the table.
- **Borders and Shading –** Select from a number of border styles, colors, and widths (fig. 1.33). Click the "Shading" tab to change the background color and pattern.
- **Options –** Click the "Options" button in the **Table Properties** window. To change the spacing between the document text and

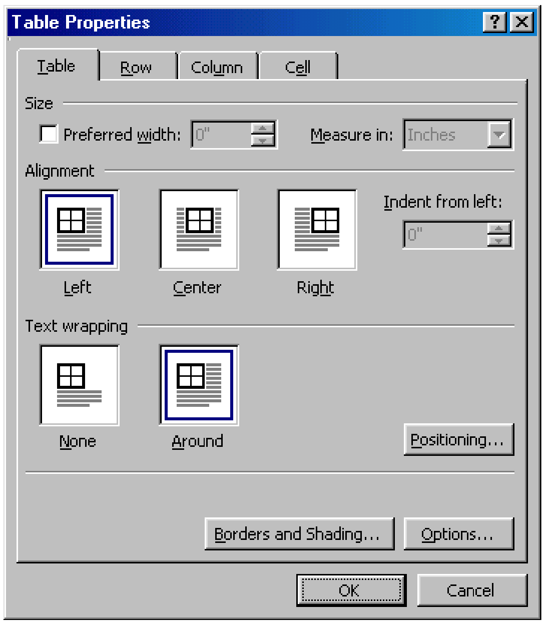
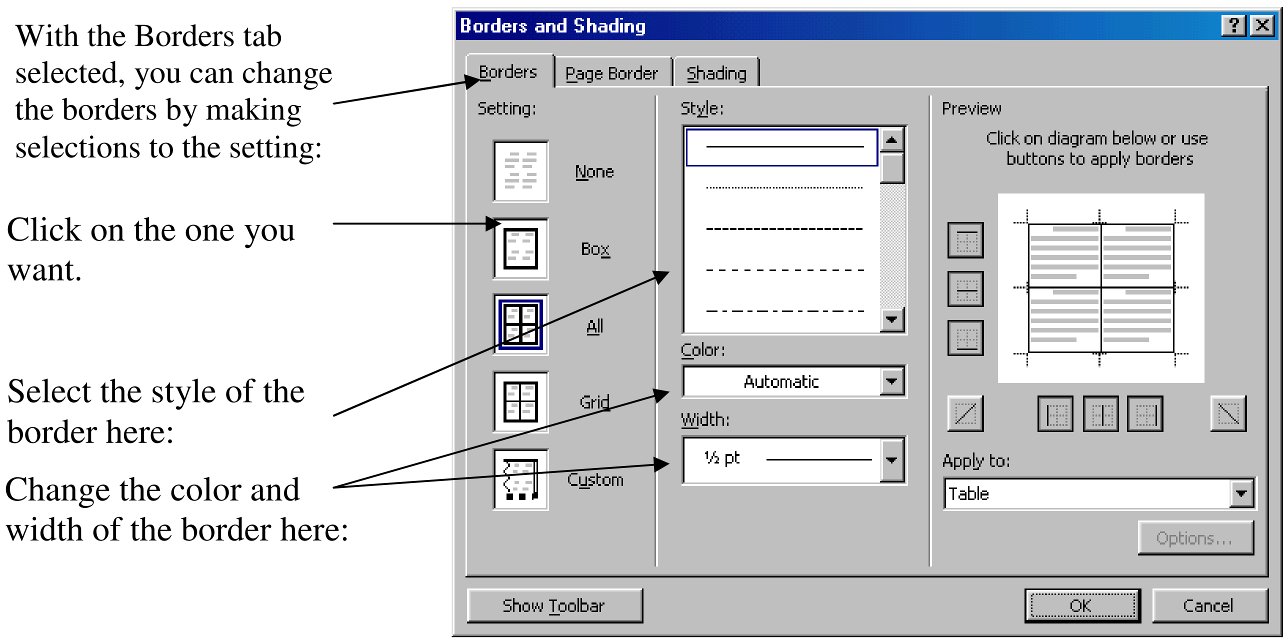


Fig. 1.32 Table properties window

the table borders under **Default cell margins**. Check the **Allow spacing between cells** box and enter a value to add space between the table cells.



With the Borders tab selected, you can change the borders by making selections to the setting:

Click on the one you want.

Select the style of the border here:

Change the color and width of the border here:

Borders and Shading

Setting: All

Color: Automatic

Width: ½ pt

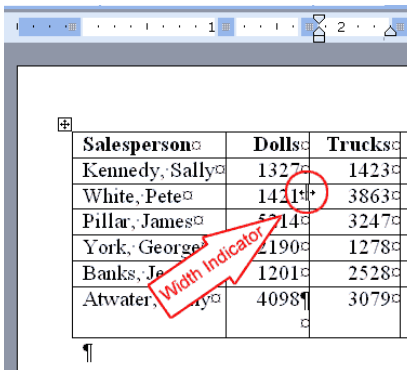
Apply to: Table

Click on diagram below or use buttons to apply borders

Fig. 1.33 Borders dialog box

**The Preview section** will show you what your borders will look like. You can change one part of the border selecting a new border style, color, and/or width, then clicking on the small boxes in the Preview section.

**NOTE:** The **Apply to**: box can be changed to select the entire table or cell.



| Salesperson | Dolls | Trucks |
|---|---|---|
| Kennedy, Sally | 1327 | 1423 |
| White, Pete | 1421 | 3863 |
| Pillar, James | 5214 | 3247 |
| York, George | 2190 | 1278 |
| Banks, Je | 1201 | 2528 |
| Atwater, yo | 4098 | 3079 |

Fig. 1.34 Resizing width of the column Dolls

## Resizing column width

You can resize your column widths by placing the cursor on the line that separates two columns. This causes the width indicator to appear. After the width indicator appears left-click and drag with the mouse to adjust the column width (fig. 1.34).

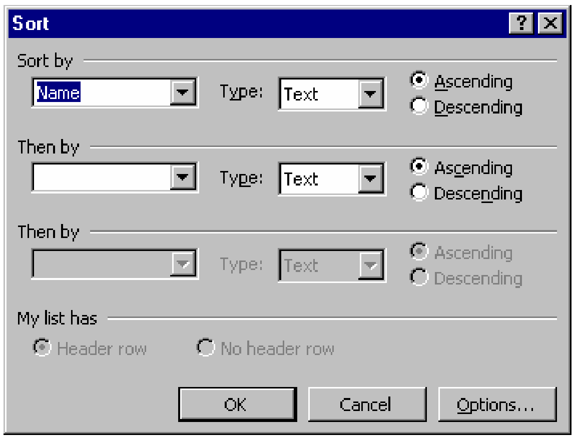
## Sorting data

With Microsoft Word, it is easy to sort the data in your table. To sort your table data by one of the fields:

1. Click anywhere on your table.
2. Choose **Table** – **Sort** from the menu.

You will see the Sort dialog box (fig. 1.35).

3. Select field in the **Sort By** field.
4. Select "Text", "Number", or "Date" in the Type field.
5. Select "Ascending" or "Descending".
6. Select **Header Row** (if your table has titles across the top of the table).
7. Click "OK".



Sort

Sort by: Name — Type: Text — [✓ Ascending] Descending

Then by: — Type: Text — [✓ Ascending] Descending

Then by: — Type: Text — [✓ Ascending] Descending

My list has: [✓ Header row] No header row

Fig. 1.35 Sort dialog box

## Deleting a Column or a Row

You can delete columns from your table.

- Place your cursor anywhere in the column (or row) that you want deleted.
- Choose **Table** – **Delete** – **Columns** (or **Rows**) from the menu.

## Merge Cell

Using Microsoft Word you can merge cells (convert two or more cells into one cell). Select cells you want to merge and choose **Table** – **Merge Cells** from the menu.

## Split a cell into multiple cells in a table

- Select the cells you want to split.
- On the **Table** menu, click **Split Cells**.
- Type the number of columns or rows you want to split each cell into.

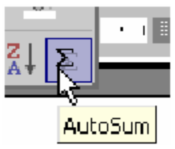


Fig. 1.36
Autosum button

## Applying calculations to a table

A simple formula can be applied to a numerical table within Word. To total a column or row of figures quickly:

- Click in the cell below or to the right of the data.
- In the **Tables & Borders** toolbar, click **Autosum** (fig. 1.36).

A wider range of formulas can be applied from the **Table** – **Formula** menu (fig. 1.37).

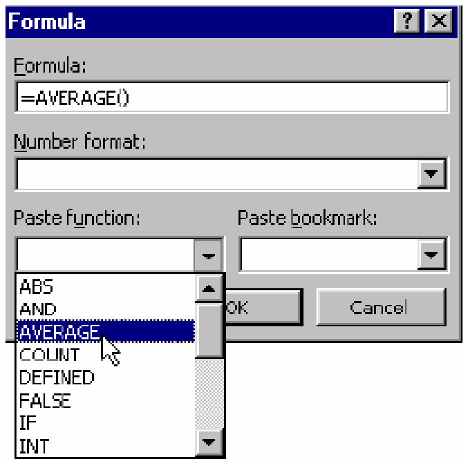


Fig. 1.37 Formula dialog box

## Table AutoFormat

You can use AutoFormats to apply borders, shade, use special fonts, and to color tables. Microsoft Word lists all Formats in the Table AutoFormat dialog box. While in the Table AutoFormat dialog box, click a format to see the format displayed in the Preview box. You can customize how the format is applied. Check the features you want in the Formats to Apply and the Apply Special Formats To Frames. Microsoft Word comes with a long list of AutoFormats.

## Converting a Table to Text

Creating tables out of text in Word is a relatively simple process. However, if you want to take text out of the table format, things seem a little trickier. You may think your only option is to cut the text from the table and paste it into a different section of your document.

Fortunately, Word has a simple solution to the problem. It gives you the option of converting tables to text. To convert a table to text, follow these steps:

1. Select your table.
2. Click the **Table** menu
3. In the **Convert** submenu, select **Table to Text…**
4. Select how you would like to separate the column entries.
5. Click **OK.**

## Tips for working faster with Word tables

Word tables have a million handy uses, from organizing tabular data to building an attractive page layout.

The following list of pointers (table 1.4) is a set of reliable timesavers for users who need to perform some basic table tasks without getting bogged down in feature subtleties.

**Table 1.4 Tips for working faster with Word tables**

**Selecting and rearranging**

| **Action** | **Function** |
|---|---|
| Select an entire table | Click the table move handle, visible when the mouse pointer is over the table in Print Layout View |
| Select a column | Position the mouse pointer above the top of the column so it turns into downward-pointing arrow and click |
| Select from the current cell to the top or bottom of the column | Press **Alt+Shift+Page Up** or **Alt+Shift+Page Down** |
| Select from the current cell to the beginning or end of the row | Press **Alt+Shift+Home** or **Alt+Shift+End** |

**Deleting**

| | |
|---|---|
| Delete a selected table | Press **Backspace** |
| Delete the *contents* of a selected table | Press **Delete**. (You can also delete the contents of specific cells by selecting them and pressing Delete) |

**Navigating**

| | |
|---|---|
| Jump from one cell to another | Press **Tab** (to move forward); press **Shift+Tab** (to move backward) |
| Jump to the first or last cell in a row | Press **Alt+Home** (to move to the first cell); press **Alt+End** (to move to the last cell) |
| Jump to the first or last cell in a column | Press **Alt+Page Up** (to move to the first cell; press **Alt+Page Down** (to move to the last cell) |

**Formatting**

| | |
|---|---|
| Split a table | Click in the row above which you want the split to occur and press **Ctrl+Shift+Enter**. (If you are at the beginning of the first table cell, this will insert a blank paragraph above the table) |

**Continuation of the Table 1.4 Tips for working faster with Word tables**

| **Action** | **Function** |
|---|---|
| Add a row to the bottom of a table | Click at the end of the last table cell and press **Tab** |
| Insert multiple rows in a table | Select as many rows as you want to add, right-click, and choose **Insert Rows**. Word will add the new rows above your selection. (The new rows will all be formatted the same as the first row in your selection) |
| Move a row (or rows) up or down | Select the row(s), hold down Alt+Shift, and press the up or down arrow key as many times as needed to move the selected row(s) to the spot you want |
| Automatically resize a column to fit its contents | Double-click on the boundary to the right of the column you're resizing |
| Resize a column without affecting the table width | Drag the right boundary of the column you want to resize. Word will adjust that column and the one on its right but keep the table the same width.<br>Or hold down Ctrl+Shift as you drag the boundary. Word will change the width of the column to the left and resize the columns to the right proportionally, leaving the table width unchanged |
| Resize a column with more precision | Hold down Alt as you drag a column boundary. Word will display the margins and column widths on the horizontal ruler. It will also give you finer control over the dragging process (similar to overriding the Snap to Grid feature for drawing objects) |
| Insert a tab in a table cell | Press **Ctrl+Tab** |

## Graphics in Word

In Word you can:

- Add Clip Art from Microsoft's extensive library (**Insert** – **Picture** – **Clip Art**)
- Place your own pictures in a document (**Insert** – **Picture** – **From file**)
- Create drawings and diagrams with Office drawing tools
- Add a wide range of effects: background patterns and pictures, page borders, and fancy headings with Word Art.

## Charts

You can create a chart by using existing data that is contained within a table in your document. To insert a chart in your document do the following:

1. Select your table (or part of it).
2. Click **Insert – Picture – Chart**.

Word will launch Microsoft Graph, which automatically creates a chart based on your table.

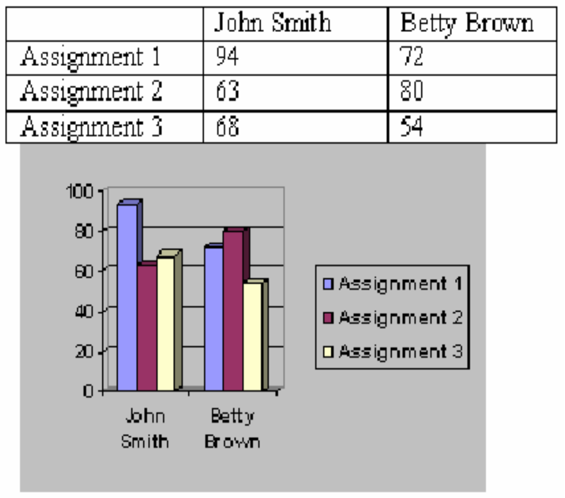
Additionally, Word adds two new menus, Data and Chart, to the menubar. These menus provide additional help with working with your chart.

You can modify the information displayed in the chart by modifying the data in the Datasheet view (just copy the corresponding data from your table).

There are many ways you can modify the chart visuals. Right-clicking on the chart provides options such as modifying the borders and shading. You can right-click on the chart elements to change shapes, change the chart type (doughnut, column, bubble, and pie charts), and much more.

Note that after editing the chart and returning to the rest of the Microsoft Word document, the Datasheet may disappear. Double-click the chart to bring the datasheet back and/or if you wish to modify the chart visuals.

Example of the chart is shown in fig. 1.38.



| | John Smith | Betty Brown |
|---|---|---|
| Assignment 1 | 94 | 72 |
| Assignment 2 | 63 | 80 |
| Assignment 3 | 68 | 54 |

Fig. 1.38 Example of the table and chart

## Spell Check

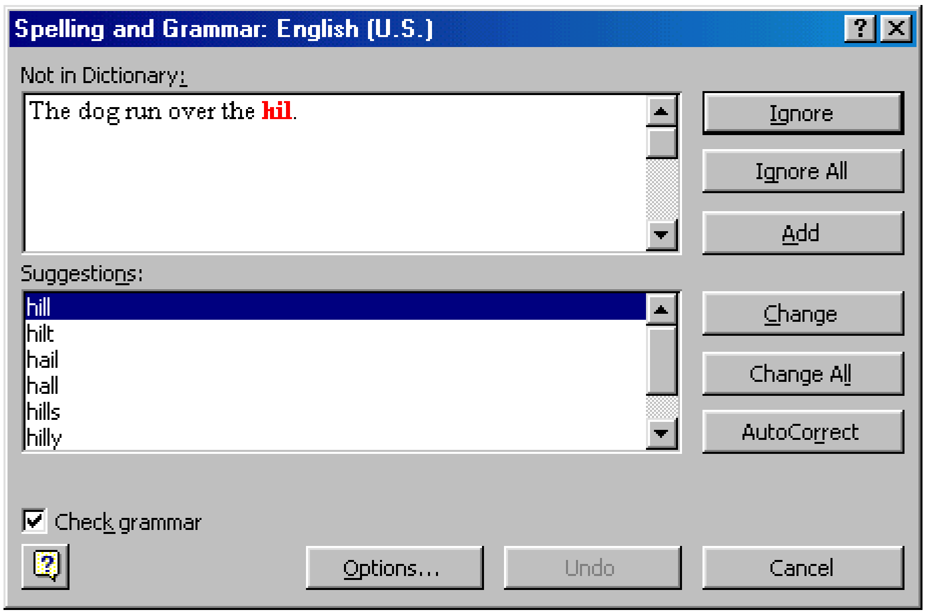
Word checks your spelling and grammar as you type. Spelling errors are displayed with a red wavy line under the word. Grammar errors are displayed with a green wavy line under the error. If you want to spell check your entire document, press F7 or click the spelling icon or choose **Tools** – **Spelling and Grammar** from the menu.

If you want to spell check part of your document, highlight the area you want to spell check and then press **F7**, click the spelling icon , or choose **Tools** – **Spelling and Grammar** from the menu.

The **Spelling and Grammar** dialog box will notify you of the first mistake in the document and misspelled words will be highlighted in red (fig. 1.39). If the word is spelled correctly, click the **Ignore** button or click the **Ignore All** button if the word appears more than once in the document.

If the word is spelled incorrectly, choose one of the suggested spellings in the **Suggestions** box and click the **Change** button or **Change All** button to correct all occurrences of the word in the document. If the correct spelling is not suggested, enter the correct spelling in the **Not In Dictionary** box and click the **Change** button.

If the word is spelled correctly and will appear in many documents you type (such as your name), click the **Add** button to add the word to the dictionary so it will no longer appear as a misspelled word.


Spelling and Grammar: English (U.S.)

Not in Dictionary:

The dog run over the hil.

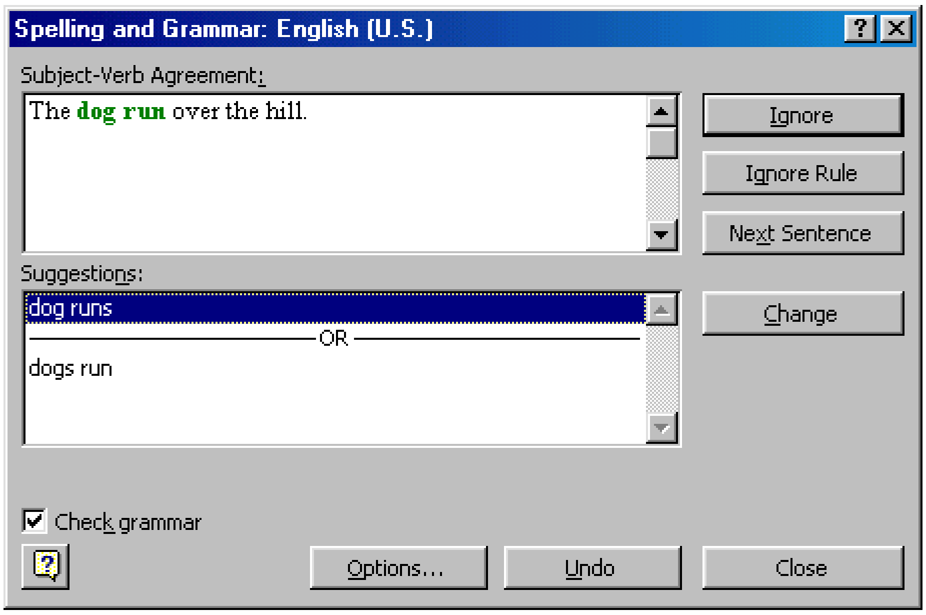
Suggestions:

hill
hilt
hail
hall
hills
hilly

Fig. 1.39 Spelling and Grammar dialog box with spelling error

As long as the **Check Grammar** box is checked in the **Spelling and Grammar** dialog box, Word will check the grammar of the document in addition to the spelling. If you do not want the grammar checked, remove the checkmark from this box. Otherwise, follow these steps for correcting grammar:

- If Word finds a grammar mistake, it will be shown in the box as the spelling error. The mistake is highlighted in green text (fig. 1.40).


Spelling and Grammar: English (U.S.)

Subject-Verb Agreement:

The dog run over the hill.

Suggestions:

dog runs
OR
dogs run

Fig. 1.40 Spelling and Grammar dialog box with grammar error

- Several suggestions may be given in the **Suggestions** box. Select the correction that best applies and click **Change**.
- If no correction is needed, click the **Ignore** button.

## Microsoft Equation Editor

To write a formula in your document:

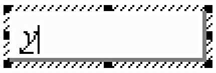
1. In the **Insert** menu, click **Object**.
2. In the Insert Object dialog box, click the **Create New** option, and then click **Microsoft Equation 3.0**.

Equation Editor will insert a blank working area into your document, and the Equation toolbar and menus will appear within the Office program's window.

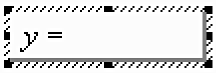
The formula you insert is an embedded object created by the Equation Editor program.

Use the following instructions to write a mathematical expression:

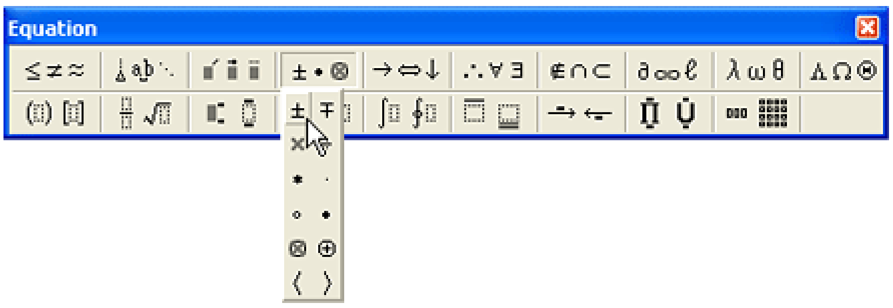
1. To enter numbers or variables, simply type them using the keyboard, such as the ***y*** typed to begin the following example equation:



2. To enter a mathematical operator that appears on the keyboard, such as the plus sign (+), the minus sign (–), or the equals sign (=), you can simply type it. For instance, you could add an equals sign to the example equation, as shown here:



3. To enter an operator or symbol that does not appear on the keyboard, click the appropriate button on the top row of the Equation toolbar and then click the desired symbol on the drop-down menu of symbols. For example, clicking the symbol shown in fig. 1.41 will add a plus-or-minus symbol:



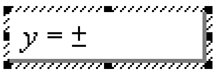
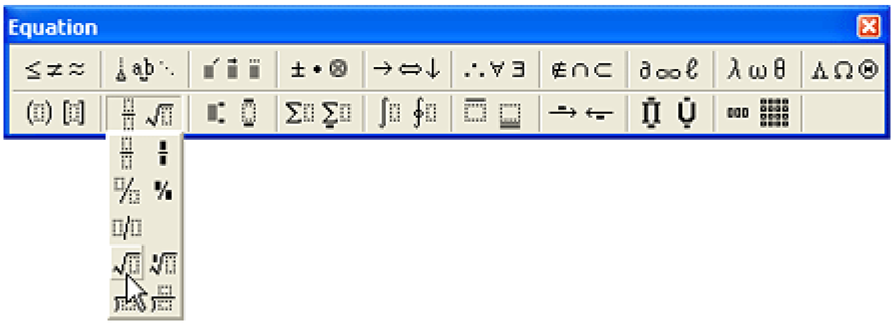


Fig. 1.41 Microsoft Equation toolbar (± symbol selected)

4. To enter an expression such as a fraction, square root, exponent, or integral, click the appropriate button on the bottom row of the Equation toolbar, and then click one of the templates on the drop-down menu. For example, click the following template to add a square-root expression to the example equation, as shown below (fig. 1.42):



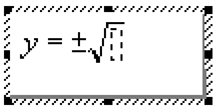
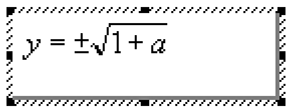


Fig. 1.42 Microsoft Equation toolbar ($\sqrt{\ }$ symbol selected)

5. Enter the desired numbers and variables into the area marked by dotted lines within the template. For example, you could type the following into the radical expression in the example equation:



$y = \pm\sqrt{1 + a}$

# Lecture 2. Introduction to programming. C++ Builder IDE

## Some Remarks about Programming

**Computer programming** (often shortened to **programming** or **coding**) is the process of writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs. This source code is written in a programming language. The code may be a modification of an existing source or something completely new. The purpose of programming is to create a program that exhibits a certain desired behavior (customization). The process of writing a source code requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.

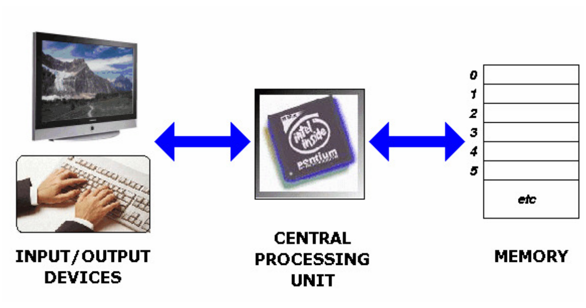
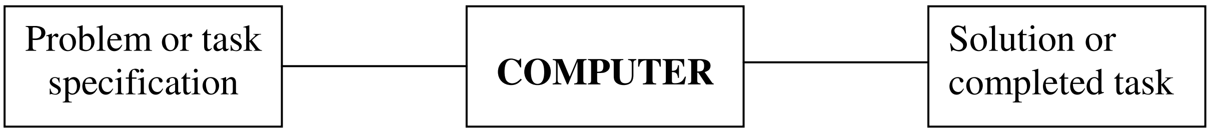


Fig. 2.1 Computer

In other words, programming is the craft of transforming requirements into something that a computer can execute.

An idealized picture is:



Unfortunately things are not as easy as they appear. In particular, the "specification" cannot be given to the computer using natural language. Moreover, it cannot just be a description of the problem or task, but it has to contain information about how the problem is to be solved or how the task is to be executed.

A **program** is a set of instructions executed by the Central Processing Unit (CPU) one after another. Those commands are generally very simple (like sums, multiplications, reading data from the Random Access Memory (RAM)), but are combined to do more complicated tasks. Typically a program consists of thousands to millions of such simple commands.

Each program must be written in a special language, which is called **programming language**. There are many different programming languages and many ways to classify them. For example, "**high-level**" programming languages are languages whose syntax is relatively close to natural language, whereas the syntax of "**low-level**" languages includes many technical references to the nuts and bolts (0’s and 1’s, etc.) of the computer. "**Declarative**" languages, as opposed to "**imperative**" or "**procedural**" languages, enable the programmer to minimize his or her account of how the computer is to solve a problem or produce a particular output. "**Object-oriented languages**" reflect a particular way of thinking about problems and tasks in terms of identifying and describing the behavior of relevant "objects". One of the most popular programming languages is C++. It includes facilities for object-oriented programming, as well as for more conventional procedural programming.

**The Origins of C++**

C++ was developed by Bjarne Stroustrup of AT&T Bell Laboratories in the early 1980's, and is based on the C language. The name is a pun; "++" is a syntactic construct used in C (to increment a variable), and C++ is intended as an incremental improvement of C. Most of C is a subset of C++, so that most C programs can be **compiled** (i.e. converted into a series of low-level instructions that the computer can execute directly) using a C++ compiler.

C is in many ways difficult to categorize. In comparison to assembly language, it is high-level, but it nevertheless includes many low-level facilities to directly manipulate the computer's memory. It is therefore an excellent language for writing efficient "systems" programs. However, for other types of programs, the C code can be difficult to understand, and C programs can therefore be particularly prone to certain types of error. The extra object-oriented facilities in C++ are partly included to overcome these shortcomings.

## ***Introduction to Borland C++ Builder IDE***

C++ is a powerful general-purpose programming language. It can be used to create small programs or large applications. It can be used to make CGI scripts or console-only DOS programs. C++ allows you to create programs to do almost anything you need to do.

An Integrated Development Environment (IDE) is an application that provides a friendly interface for creating computer programs. We will write our programs in Borland C++ Builder (BCB) IDE.

BCB uses a "visual" programming paradigm with C++ as its underlying language. This means you use the mouse to design the look of your application such as arranging "components" (buttons, scroll bars, menus etc) on skeleton windows ("forms"). This leaves you with a series of files (C++ source code, C++ headers, forms definitions and resources) to which you add the code which makes your application carry out its function. The attributes of each component (size, position, font of lettering etc) are specified interactively at design time or may be modified dynamically by the program.

There is no need to master the complexities of Windows programming, as the components act as an interface between your program and the operating system.

A Borland C++ Builder IDE screen is shown in fig. 2.2.

Across the top of the window is the title bar with the "minimize", "maximize" and "close" buttons on the right-hand side. Under the title bar are ten menus.

The **File**, **Edit** and **Search** menus have all the usual Windows options, whilst the other menus contain options specific to the C++ Builder environment.

The **View** menu allows you to view a number of windows containing program and environment information.

The **Project** menu is for the manipulation of project files and control of all aspects of a project, allowing you to compile all or part of a program.

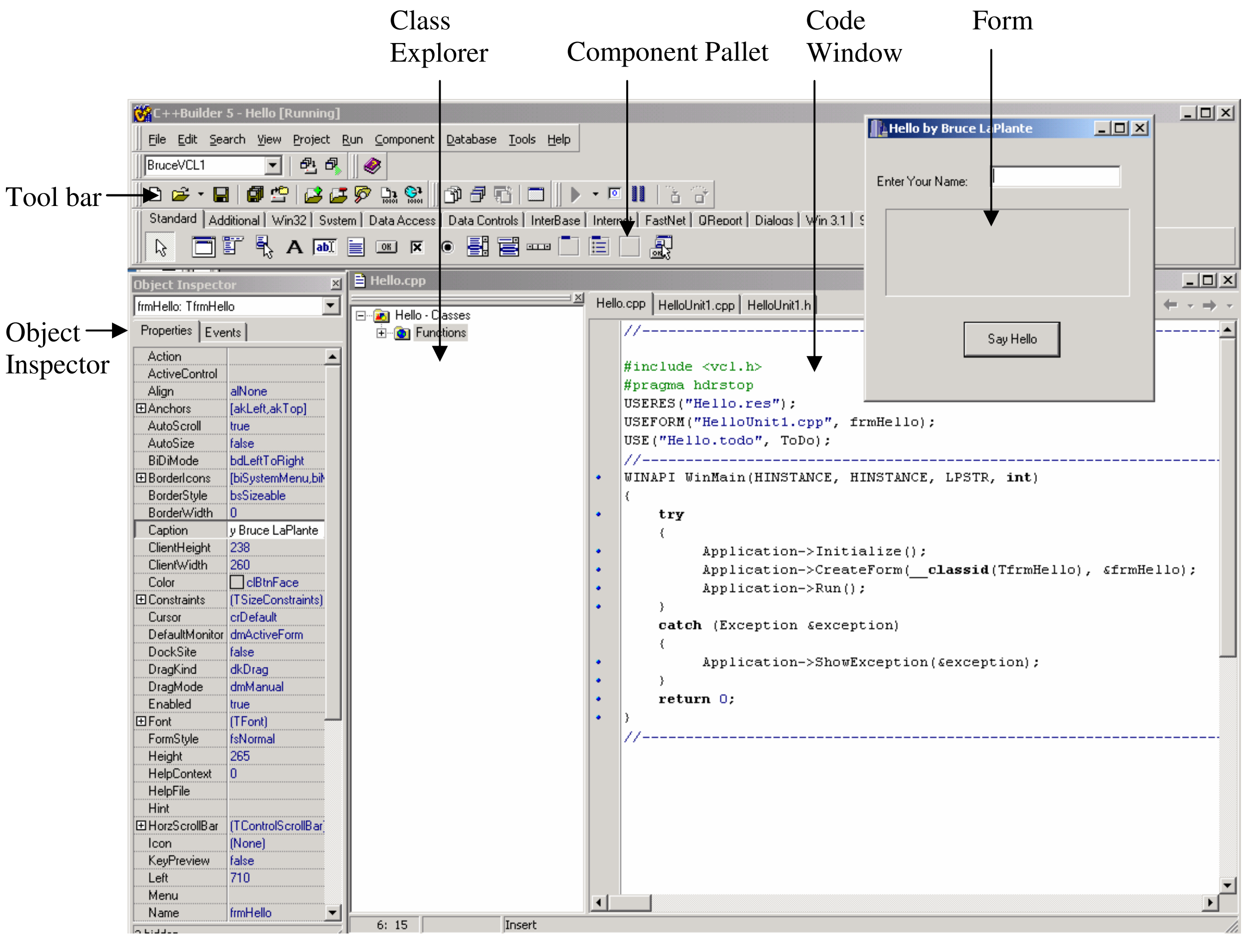
The **Run** menu provides facilities testing and debugging the application.

The **Component** menu allows for defining, specification and insertion of external objects into the application.

The **Database** menu provides ways of interacting with several types of live database.

The **Tools** menu allows you to run other programs, tools and utilities useful in the development of Windows applications, without leaving the IDE.

The **Help** menu gives you access to on-line help including tutorials, programming syntax, examples and library information.


Class Explorer
Component Pallet
Code Window
Form
Tool bar
Object Inspector
C++Builder 5 - Hello [Running]
File Edit Search View Project Run Component Database Tools Help
BruceVCL1
Standard Additional Win32 System Data Access Data Controls InterBase Internet FastNet QReport Dialogs Win 3.1
Hello by Bruce LaPlante
Enter Your Name:
Say Hello
Object Inspector
frmHello: TfrmHello
Properties Events
Hello.cpp
Hello - Classes
Functions
Hello.cpp HelloUnit1.cpp HelloUnit1.h
6: 15 Insert

| | |
|---|---|
| Action | |
| ActiveControl | |
| Align | alNone |
| ⊞Anchors | [akLeft,akTop] |
| AutoScroll | true |
| AutoSize | false |
| BiDiMode | bdLeftToRight |
| ⊞BorderIcons | [biSystemMenu,bi |
| BorderStyle | bsSizeable |
| BorderWidth | 0 |
| Caption | y Bruce LaPlante |
| ClientHeight | 238 |
| ClientWidth | 260 |
| Color | ☐ clBtnFace |
| ⊞Constraints | (TSizeConstraints) |
| Cursor | crDefault |
| DefaultMonitor | dmActiveForm |
| DockSite | false |
| DragKind | dkDrag |
| DragMode | dmManual |
| Enabled | true |
| ⊞Font | (TFont) |
| FormStyle | fsNormal |
| Height | 265 |
| HelpContext | 0 |
| HelpFile | |
| Hint | |
| ⊞HorzScrollBar | (TControlScrollBar |
| Icon | (None) |
| KeyPreview | false |
| Left | 710 |
| Menu | |
| Name | frmHello |

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
USERES("Hello.res");
USEFORM("HelloUnit1.cpp", frmHello);
USE("Hello.todo", ToDo);
//---------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
         Application->Initialize();
         Application->CreateForm(__classid(TfrmHello), &frmHello);
         Application->Run();
    }
    catch (Exception &exception)
    {
         Application->ShowException(&exception);
    }
    return 0;
}
//---------------------------------------------------------------------
```

Fig. 2.2 C++ Builder IDE screen

The main part of the IDE screen is taken up by a blank **form**. You will place different components in it. It will allow the user of your program to interact with the computer. The easiest way to add a control to a form is to click it on the Component Palette (fig. 2.2), and click on the form. At the beginning we will use controls such as: Button, Edit Box, Label, and Memo.

Each application is represented by a **project**. A new project automatically contains the following files:

- Project1.cpp – a source-code file associated with the project.
- Unit1.cpp – a source-code file associated with the main project form. It is called a **unit file**.
- Unit1.h – a header file associated with the main project form. It is called a **unit header file**.
- Unit1.dfm – a resource file that stores information about the main project form. It is called a **form file**.

You should save these files to your folder by choosing **Save All** from the **File** menu. You do not need to worry about these files, but do not delete them.

## Setting Properties

A **property** is anything that characterizes or describes an object. The properties include the name of the control, its color, dimensions, and many other features that set a control apart from the others. These are visual characteristics accessible every time the object is displayed.

The properties of a control can be changed either at design or run time. To change the properties of a control at design time, you will use the **Object Inspector** (fig. 2.2).

Each field on the Object Inspector has two sections: the property's name and its field value. The box on the right side of each name represents the value of the property that you can set for an object.

The **Caption** of a form or a control is the word or group of words that display(s) on the form's title bar or on the control.

An important property of a control is its **Came**. The name of a control allows you and the compiler to relate to the control.

Depending on what you are trying to do, sometimes you will not want the user to see a control until another action has occurred. To hide a control, use the **Visible** property. It toggles the appearance and the disappearance of a control. Choose **False** if you do not want to see a button or **True** if you want to see it.

## The Label Control

A label is a control that serves as a guide to the user. It provides a static text that the user cannot change but can read to get information about another control on the form. The programmer can also use it to display simple information to the user. The most used property of the label control is the caption. The **Caption** controls what the user would see or read.

## Edit Boxes

An Edit Box is a Windows control used to get or display text for the user's interaction. Typically, an Edit Box serves as a place to fill out and provide information. Like most other controls, an edit box should be accompanied by a Label that defines its purpose.

The most important Edit Box property is the **Text**. When you add an Edit control, C++ Builder initializes it with the name of the control; this would be Edit1 for the first edit box, Edit2 for the second, etc. If you want the control to display text when the form launches, type the text in the Text property field. Otherwise, if you want the Edit Box to be empty when it comes up for the first time, delete the content of the Text field.

Another important property of the Edit control is the **Name**. It makes it easy for you to identify the control and its events when writing code.

## The Command Button

A Button is a Windows control used to initiate an action. From the user's standpoint, a button is useful when clicked, in which case the user positions the mouse on it and presses one of the mouse's buttons.

There are various kinds of buttons. The most popular button used in Windows applications is a Command Button – a rectangular control that displays a word or a short sentence that directs the user to access, dismiss, or initiate an action or a suite of actions. From the programmer's standpoint, a button needs a host; this could be a form, a toolbar, or another container.

To add a command button to your form, click the **Button** control from the **Standard** tab of the Component Palette and click on the form.

From the user's point of view, the only things important about a button are the message it displays and the action it performs. The **Caption** of a button is a word or phrase displayed on top of the control, indicating what the button is used for. To change the caption on a control, in the Object Inspector, click the word Caption and type the desired caption.

To the programmer, one of the most important properties of any control is the **Name**. This allows you and the compiler to know what control you are referring to when the program is running. By default, the first button you add to a form is named Button1, the second would be Button2. Since a program usually consists of many buttons, it would be a good idea to rename your buttons and give them meaningful names. The name should help you identify what the button is used for. For example, OKButton, ExitButton, SolveButton.

The typical action a user performs with a button is **click** it. To initiate the **OnClick** event on a button, **double-click it**.

There are many other controls we will get acquainted with later.

When the **interface** of the program is ready, it is time to write a **code** of the program that directs the computer as to what to do, when, and how. This is done in an appropriate window called the Code Editor (fig. 2.2). To access the Code Editor, if you have a form opened, you can press F12.

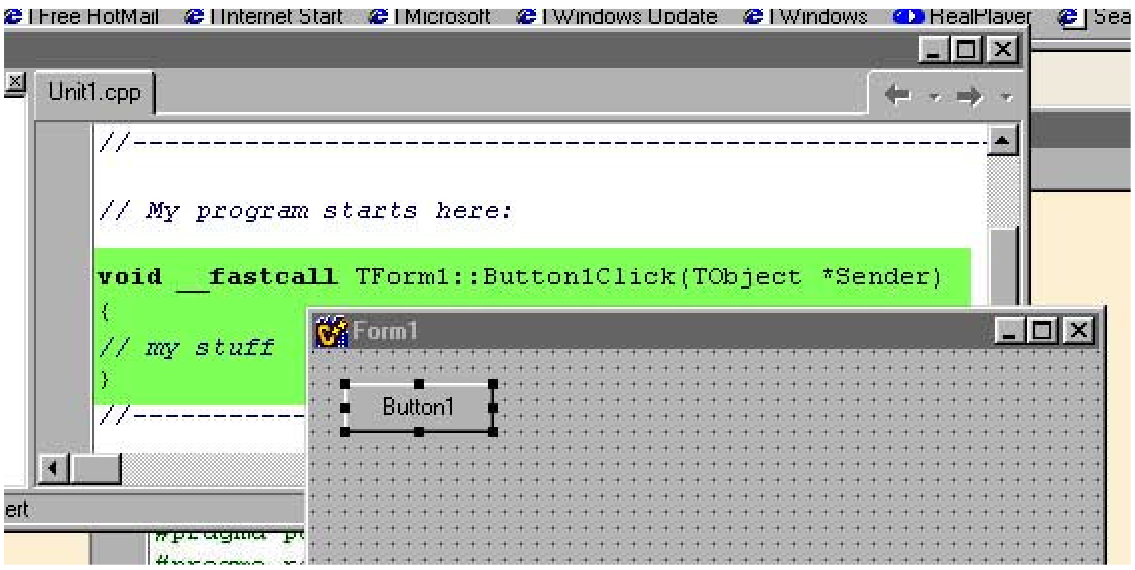
## Controls Events

An event is an action that occurs in the life of a control. The job of a programmer consists of writing instructions, what computer must do, when an event occurs.

The users of your program will mainly use a mouse and keyboard to interact with your application. These two objects are the primary sources of using events. We will consider two main events – OnClick and OnChange. OnClick event occurs when a user clicks the Button or other control. OnChange event occurs, for example, when a user writes something in the Edit Box.

**Event handlers** are special functions which are invoked automatically when a certain event occurs. Almost all components have a set of event handlers, which can be seen on the **Events** page of the **Object Inspector**.

To program an event (write event's handler), you must double click on the control, which "owns" the event. The name of an event is made up of a combination of a control that "owns" the event and the name of the event. For example, if a button called Button1 fires an OnClick event, the compiler would call it Button1Click event (fig. 2.3).

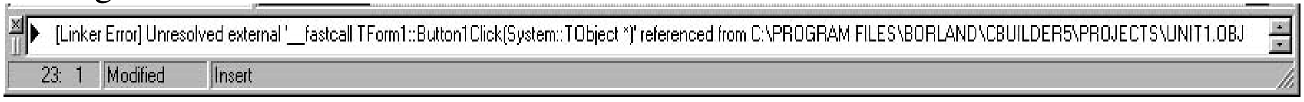


```
//-----------------------------------------------------------------

// My program starts here:

void __fastcall TForm1::Button1Click(TObject *Sender)
{
// my stuff
}
//-----------
```

Fig. 2.3 Button1 and its onClick event's handler

Do not try to write or paste-in the code of the event handler header yourself. In the fig. 2.3, Button1 has been placed on Form1 and has been double-clicked to create the highlighted event handler. When an event handler is created properly (through the Borland IDE) many references to it are also created and stored in files other than Unit1.cpp. When an event handler is destroyed properly (through the Borland IDE) all references to it in other files are also destroyed.

To DELETE an event handler simply delete the contents of the handler (inside the braces {}). When you link and compile the program, Borland will then delete the event-handler and all references to it. However, you may delete the button yourself. DO NOT delete the event-handler yourself, if you do you will get the following error message:



```
[Linker Error] Unresolved external '__fastcall TForm1::Button1Click(System::TObject *)' referenced from C:\PROGRAM FILES\BORLAND\CBUILDER5\PROJECTS\UNIT1.OBJ
23: 1   Modified   Insert
```

Another way to create an event handler:

1. Select a control that "owns" the event.
2. In the Events tab of Object Inspector select appropriate event in the left part (for example, onMouseMove).
3. Double click in the right part of this event.

## Code Editor

Besides designing applications, one of your most regular jobs will consist of writing code that directs the computer as to what to do, when, and how to do it. This is done in an appropriate window called the **Code Editor**.

The Code Editor is a featured text editor adapted for coding purposes. It is programmed to recognize the parts of a program that are recognized by C++ or not. To access the Code Editor, if you have a form opened, you can press F12. The Code Editor manages your jobs by organizing its files into property pages (also called tabs). If your project contains more than one file, you can click the desired tab to access one of the files.

To display the header file of the form, you can right-click the source file and click Open Source/Header File. Indeed, this action is used to toggle both displays. Since the source and the header file go in pair (when using classes), they hold the same name but have different extensions:

When your program is ready, you can **execute** and test it. To execute a program, you can press F9 or you can use the main menu where you would click **Run** – **Run**. In the toolbar, you can also click the Run button ▶. The executing of a program begins from compiling (translation from C++ to machine language). If the program is successfully compiled it begins to execute.

Our purpose is to learn basic commands of C++ and how to use these commands in program.

**Remember, that:**

1. each operator in your program should end with a **semicolon** (**;**)
2. C++ is **case sensitive** (C++ distinguishes lower case and uppercase letters).

## *Variables and Data Types*

We can work with various kinds of information, such as numbers, characters, strings, sets of numbers. We can use integer numbers or numbers with a floating point. Different types of information demand different amounts of memory space and allow doing different operations with them. For example, we can add and multiply numbers, but we cannot do the same with strings.

All data, with which the program works, are during its execution in the RAM. There the information is stored in variables.

**Variable** is a piece of memory in which we store information during the program's work. Each variable has a name. The name of a variable:

- Starts with an underscore "_" or a Latin letter, lowercase or uppercase, such as a letter from ***a*** to ***z*** or from ***A*** to ***Z***. Examples are `Name`, `gender`, `_Students`, `pRice`
- Can include letters, underscore, or digits. Examples are: `keyboard`, `Master`, `Junction`, `Player1`, `total_grade`, `_Score_Side1`
- Cannot include special characters such as !, %, ], or $
- Cannot include an empty space
- Cannot be any of the reserved words (the Application 3 contains the list of reserved words)
- Should not be longer than 32 characters (although allowed)

Note, that **A** and **a** are different names.

We access to data, which are stored in a variable, on its name. Variables must be **declared** before they may be used. The syntax to declare a new variable is to write the specifier of the desired data type (like **int**, **bool**, **float**...) followed by a valid variable identifier:

data_**type** **name**_of_the_variable;

A data **type** provides two valuable pieces of information to the compiler:

- the **amount of space** the variable will use;

- the **kind of information** allowed.

After declaring a variable, the compiler reserves a space in memory for that variable.

**The basic numeric data types**

**int** – stores an integer value. For example: 3, -56, 1000;

**float** – stores a floating point or real value. For example: 3.7, -56.0, 1000.87;

**char** – stores a single character. A literal or constant character value is a value enclosed in single quotes for example: ‘K’, ‘9’,’*’;

**bool** – stores a value representing true or false. We look at this in detail in the next lecture.

Full information about numeric data types in C++ is shown in table 2.1.

**Examples of declaration**:

```
int a;          // a is an integer variable
int a1, c;      // a1 and c are integer variables
float b, x;     // b and x are floating point variables
```

The integer data types **char**, **short**, **long** and **int** can be either signed or unsigned depending on the range of numbers needed to be represented. Signed types can represent both positive and negative values, whereas unsigned types can only represent positive values (and zero). This can be specified by using either the specifier **signed** or the specifier **unsigned** before the type name. For example:

```
unsigned int Num;
signed int M;
```

By default, if we do not specify either a signed or unsigned compiler assume the type to be signed.

**Table 2.1 Numeric data types in C++**

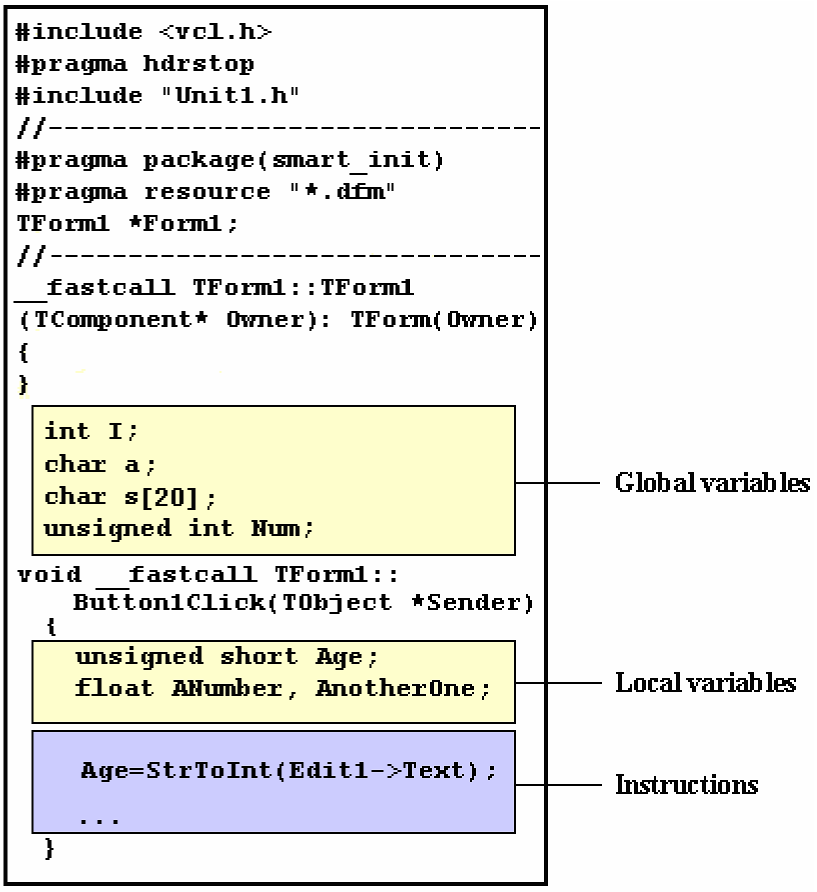
| Name | Description | Size | Range |
|---|---|---|---|
| **char** | character or small integer | 1 byte | signed: -128 to 127<br>unsigned: 0 to 255 |
| **short int (short)** | short integer | 2 bytes | signed: -32768 to 32767<br>unsigned: 0 to 65535 |
| **int** | integer | 4 bytes | signed: -2147483648 to 2147483647<br>unsigned: 0 to 4294967295 |
| **long int (long)** | long integer | 4 bytes | signed: -2147483648 to 2147483647<br>unsigned: 0 to 4294967295 |
| **bool** | boolean value. it can take one of two values: true or false | 1 byte | true or false |
| **float** | floating point number | 4 bytes | +/- 3.4e +/- 38 (~7 digits) |
| **double** | double precision floating point number | 8 bytes | +/- 1.7e +/- 308 (~15 digits) |
| **long double** | long double precision floating point number | 8 bytes | +/- 1.7e +/- 308 (~15 digits) |

It is possible to **initialize** a variable's value at the declaration:

```
int a=1;
float b=8.3;
```

## Scope of variables

A variable can be either of global or local scope. A **global** variable is a variable declared in the main body of the source code, outside all functions, while a **local** variable is declared within the body of a function or a block. Variables I, a, s, Num in fig. 2.4 are global. Variables Age, ANumber, AnotherOne are local.



```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//--------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//--------------------------------
__fastcall TForm1::TForm1
(TComponent* Owner): TForm(Owner)
{
}
 int I;
 char a;
 char s[20];
 unsigned int Num;
void __fastcall TForm1::
    Button1Click(TObject *Sender)
  {
    unsigned short Age;
    float ANumber, AnotherOne;

    Age=StrToInt(Edit1->Text);
    ...
  }
```

Fig. 2.4 Scope of variables

Global variables can be referred from anywhere in the code, even inside functions, whenever they are after a variable's declaration.

The scope of local variables is limited to the block enclosed in curly brackets ({}) where they are declared. For example, if they are declared at the beginning of the function body (like in function Button1Click in fig. 2.4) their scope is between its declaration point and the end of that function. In fig. 2.4, this means that if another function exists in addition to Button1Click (for example Button2Click), the local variables declared in Button1Click could not be accessed from this function.

## Assignment (=)

To set or change a variable's value, it must be **assigned**. The assignment operator assigns a value to a variable.

```
a = 5;
```

This statement assigns the integer value 5 to the variable ***a***. The part at the left of the assignment operator (=) is known as **lvalue** (left value) and the right one as **rvalue** (right value). The lvalue has to be a variable whereas the rvalue can be either a constant, a variable, the result of an operation or any combination of these.

The most important rule when assigning is the right-to-left rule: "The assignment operation always takes place from right to left, and never the other way around":

```
a = b;
```

This statement assigns the value contained in variable ***b*** (the rvalue) to the variable ***a*** (the lvalue). The value, that was stored until this moment in ***a***, is not considered in this operation. In fact, that value is lost.

Consider also that we are only assigning the value of ***b*** to ***a*** at the moment of the assignment operation. Therefore, a later change of ***b*** will not affect the new value of ***a***.

Consider the variables:

```
int a, c;
float b;
double x;
```

Examples of assignment:

`a=15;`
`b=-2.7;`
`x=2+5.1;` (after this ***x*** is 7.1)
`b=a;` (in this case a value of ***b*** will be 15.0)
`c=x;` (in this case a value of ***c*** will be 7, digits after point are rejected)
`a=a+1;` (the value of variable ***a*** increases on 1, now **a** is equal to 16)

The following expression is valid in C++:

```
a = b = c = 5;
```

It assigns 5 to the all the three variables: ***a***, ***b*** and ***c***.

**Arithmetic operators ( +, -, *, /, % )**

The five arithmetical operations supported by the C++ language are: **+** (addition), **-** (subtraction), ***** (multiplication), **/** (division), **%** ( modulus).

Modulo is the operation that gives the remainder of a division of two values. For example, if we write:

```
a = 11 % 3;
```

then the variable ***a*** will contain the value 2, since 2 is the remainder when dividing 11 by 3.

Examples of arithmetic operators (let int a=5, float b=6.5, int c,x):

```
x=3-a;          // x=-2
c=a*(b-a);      // c=7, digits after point are rejected
a++;            // a=6
c--;            // c=6
b=c/2;          // b=3
b=(a+1)/3;      // 7/3=2, b=2
a=10%4;         // a=2 – reminder of the division
```

Note that int divided by int gives int.

There are some issues which need your attention when dealing with arithmetic operations.

**Compound assignment (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)**

When we want to modify the variable's value with performing an operation on the value currently stored in that variable we can use compound assignment operators:

| expression | is equivalent to |
|---|---|
| `a += b;` | `a = a + b;` |
| `a -= 5;` | `a = a - 5;` |
| `a /= b;` | `a = a / b;` |
| `a *= b + 1;` | `a = a * (b + 1);` |

**Increase and decrease (++, --)**

The increase operator (**++**) and the decrease operator (**--**) increase or reduce by one the value stored in a variable. They are equivalent to **+=1** and to **-=1**, respectively. Thus:

```
c++;
c+=1;
c=c+1;
```

are all equivalent in its functionality. These three statements increase on one the value of c.

A characteristic of this operator is that it can be used both as a prefix and as a suffix. That means that it can be written either before the variable identifier (++a) or after it (a++). Although in simple expressions like a++ or ++a, both have exactly the same meaning, in other expressions in which the result of the increase or decrease operation is evaluated as a value in an outer expression they may have an important difference in their meaning. In the case that the increase operator is used as a **prefix** (++a) the value is increased **before** the result of the expression is evaluated and therefore the increased value is considered in the outer expression; in case that it is used as a **suffix** (a++) the value stored in ***a*** is increased **after** being evaluated and therefore the value stored before the increase operation is evaluated in the outer expression. Notice the difference:

| Example 1 | Example 2 |
|---|---|
| `B=3;` | `B=3;` |
| `A=++B;` | `A=B++;` |
| *//A contains 4, B contains 4* | *//A contains 3, B contains 4* |

In Example 1, ***B*** increases before its value is copied to ***A***. While in Example 2, the value of ***B*** is copied to ***A*** and then ***B*** increases.

**Examples** (let x=1, y=2)

```
x = x + 1;   // This increments x, x=1
x++;         // This increments x, x=2
++x;         // This increments x, x=3
```

```
z = y++;        // After this z = 2, y = 3
z = ++y;        // After this z = 4, y = 4

y = y - 1;      // This decrements y, y=3
y--;            // This decrements y, y=2
--y;            // This decrements y, y=1
y = 3;
z = y--;        // After this z = 3, y = 2
z = --y;        // After this z = 1, y = 1

a = a + 12;     // This adds 12 to a
a += 12;        // This adds 12 more to a
a *= 3.2;       // This multiplies a by 3.2
a -= b;         // This subtracts b from a
a /= 10.0;      // This divides a by 10.0
```

## Precedence of operators

When writing complex expressions with several operands, we may have some doubts about which operand is evaluated first and which later. For example, in this expression:

```
a = 5 + 7 % 2
```

we may doubt if it really means:

```
a = 5 + (7 % 2)       // with a result of 6,
```

or

```
a = (5 + 7) % 2       // with a result of 0
```

The correct answer is the first of the two expressions, with a result of 6. There is an established order with the priority of each operator, and not only the arithmetic ones (those whose preference come from mathematics) but for all the operators which can appear in C++. From greatest to lowest priority, the priority order is shown in Application 2.

All these precedence levels for operators can be manipulated or become more legible by removing possible ambiguities using parentheses signs ( and ), as in previous examples.

If you want to write a complicated expression and you are not completely sure of the precedence levels, always include parentheses. It will also become a code that is easier to read.

## Preprocessors: #include

The preprocessor is used to give a special instruction to the compiler. It is typically placed at the beginning of a line and it is followed by a word that would perform an action. There are various actions you could ask the compiler to perform.

The **#include** is used to include an external file in the current file. The file to include is called a **header file**, a **library**, or another kind of file you want the compiler to consider before going any further.

## *Input/Output*

We can use the assignment statement to set a variable value. Another way to give some value to a variable is to input it from the keyboard (or from the form). For this purpose we must place Edit Boxes on the form for each variable. By default, the content of a text control, such as an Edit Box, is a string. If you want the value or content of such a control to participate in a mathematical operation, you must first convert a string value to a valid data type.

To read (input) the value from the Edit1 in ***integer*** variable ***x*** we must write:

```
x=StrToInt(Edit1->Text);
```

This command converts Edit1->Text (a string) to an integer value with the **StrToInt** function and then assign this value to ***x***.

To read (input) the value from the Edit1 in the ***float*** variable ***x*** we must write:

```
x=StrToFloat(Edit1->Text);
```

This command converts Edit1->Text (a string) to a float value with the **StrToFloat** function and then assign this value to ***x***.

If we want to get a value of a variable, we must output it on the form also in the Edit Box.

To write (output) the value of ***integer*** variable ***x*** in Edit1 we must write:

```
Edit1->Text=IntToStr(x);
```

This command converts an integer number ***x*** to the string with an **IntToStr** function and then writes this value in Edit1.

To write (output) the value of a ***float*** or double variable ***x*** in Edit1 we must write:

```
Edit1->Text=FloatToStr(x);
```

This command converts the float or double number ***x*** to the string with the **FloatToStr** function and then writes this value in Edit1.

Another way to convert a float or double to string is to use the **FormatFloat** function:

```
Edit1->Text=FormatFloat(“0.00”,x);
```

The number will result in 2 digits after the decimal point.

### Comments

Comments are parts of the source code disregarded by the compiler. They simply do nothing. Their purpose is only to allow the programmer to insert notes or descriptions embedded within the source code.

C++ supports two ways to insert comments:

```
// line comment
/* block comment */
```

The first of them, known as line comment, discards everything from where the pair of slash signs (//) is found up to the end of that same line. The second one, known as a block comment, discards everything between the /* characters and the first appearance of the */ characters, with the possibility of including more than one line.

### Example 2.1 of the project

The task: *Input one integer and one float numbers and calculate their sum.*

The form of the project contains the following components: three Labels, three Edit Boxes and three Buttons. You can see their captions in table 2.2:

**Table 2.2 Captions of the components**

| **Name** | **Caption** |
|---|---|
| Label1 | Input integer number: |
| Label2 | Input float number: |
| Label3 | Sum= |
| Button1 | Calculate |
| Button2 | Clear |
| Button3 | Exit |

The form of the program with inputted values and calculated sum is shown in fig. 2.5:

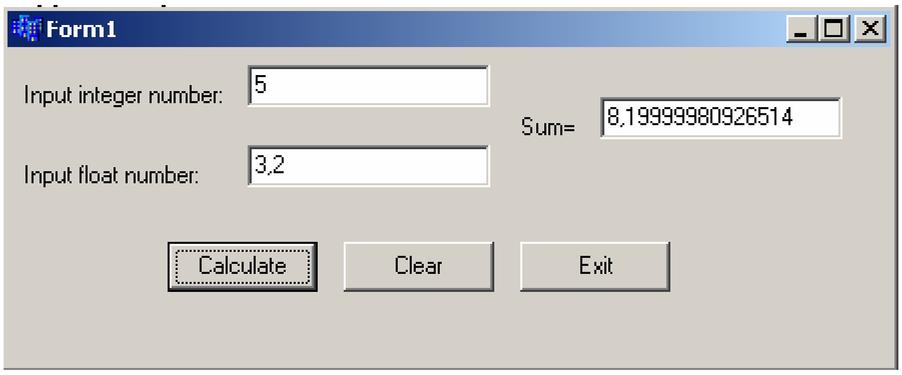


Fig. 2.5 Form of the project with result

Complete code of the program:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

//"Calculate" button
void __fastcall TForm1::Button1Click(TObject *Sender)
{
int a;                //integer variable a
float b;              //float variable b
float sum;            //sum is also float
a=StrToInt(Edit1->Text);          //input a from Edit1
b=StrToFloat(Edit2->Text);            //input b from Edit2
sum=a+b;             //calculate the sum
Edit3->Text=FloatToStr(sum);          //output result in Edit3
        //the            same             is:            Edit3-
    >Text=FormatFloat("0.00",sum);
}
//-------------------------------------------------------------------------
//"Clear" button
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Edit1->Clear();
Edit2->Clear();
Edit3->Clear();
}
//-------------------------------------------------------------------------
//"Exit" button
void __fastcall TForm1::Button3Click(TObject *Sender)
{
Close();
}
```

If we write
`FormatFloat("0.00",sum)`
instead of
`FloatToStr(sum)`,
then the sum will have only 2 digits after the decimal point and the form with result will be as the following (fig. 2.6).

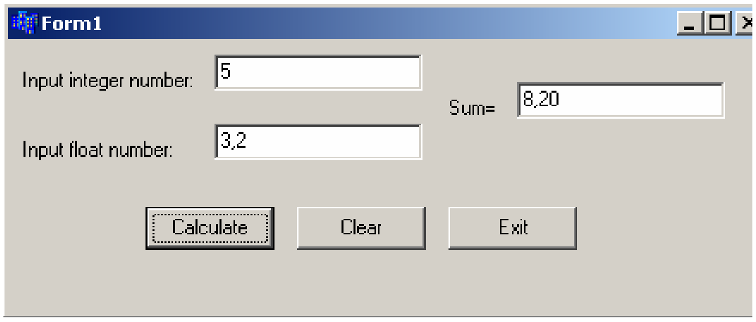

Form1
Input integer number: 5
Input float number: 3,2
Sum= 8,20

Fig. 2.6 Form of the project with alternative

## Lecture 3. Arithmetic expressions in C++. Programs with linear structure

By means of a mathematical library in C ++ it is possible to calculate values of complex mathematical expressions (formulas).

**Preprocessor instructions**

The preprocessor runs before your compiler each time the compiler is invoked. The preprocessor translates any line that begins with a pound symbol (#) into a special command, getting your code file ready for the compiler.

**Include** is a preprocessor instruction that says, "What follows is a filename. Find that file and read it in right here." The angle brackets around the filename tell the preprocessor to look in all the usual places for this file. If your compiler is set up correctly, the angle brackets will cause the preprocessor to look for the file in the directory that holds all the H files for your compiler.

### *Math Functions*

Math library contains a lot of mathematical functions. If you are going to use arithmetic functions, you must include header file **math.h** to your program and write:

#include <math.h>

at the beginning of the program. The short list of mathematical functions:

int **abs** (int x) – computes absolute value of integer ***x***
double **fabs** (double x) – computes absolute value of float ***x***
double **sqrt**(double x) – computes the square root of ***x***
double **pow** (double x, double y) – computes ***x*** raised to the power ***y***
double **exp**(double x) – computes exponential of ***x***
double **log**(double x) – computes ***ln x***
double **log10** (double x) – computes log to the base ***10*** of ***x***
double **sin**(double x) – computes sine of angle in radians
double **cos**(double x) – computes cosine of angle in radians
double **tan**(double x) – computes tangent of angle in radians
double **acos**(double x) – computes arc cosine of ***x***
double **asin**(double x) – computes arc sine of ***x***
double **atan**(double x) – computes arc tangent of ***x***

**Examples of mathematical expressions in C++**

Now we will write the following mathematical formulas on C++ language.

1. $Y = \frac{4x^2 - 3^x}{2x^2 + 1} + \frac{\ln x}{2x + 3}$

```
Y=(4*x*x-pow(3,x))/(2*x*x+1)+log(x)/(2*x+3)
```

2. $N = \frac{3y^2 + \sqrt{y+1}}{\ln(p + y) + e^p}$

```
N=(3*y*y+sqrt(y+1))/(log(p+y)+exp(p)
```

3. $f = \dfrac{\cos^7 bx^5 - (\sin a^2 + \cos(x^3 + z^5 - a^2))}{(\arcsin a^2 + \arccos(x^7 - a^2))}$

```
f=(pow(cos(b*pow(x,5)),7)-sin(a*a)+cos(pow(x,3)+
    pow(z,5)-a*a)))/ (asin(a*a)+acos(pow(x,7)-a*a))
```

4. $y=\lg^5\left(x^3 \ln^6\left(x^2+1.7\right)^9 + e^{\sqrt{x}}\right)^2 - arctg\left|\sqrt[3]{x} + tg^3 x^8\right|$

```
y=pow(log10(pow(pow(x,3)*pow(log(pow(x*x+1.7,9)+
    exp(sqrt(x)),2),6)),5)-atan(abs(pow(x,1./3)+
    pow(tan(pow(x,8)),3)))
```

5. $y = \sin^4\left(a^2+b^2\right);\ a = \sqrt{b+t};\ t = b^2 + k^3$

```
t=b*b+pow(k,3);
a=sqrt(b+t);
y=pow(sin(a*a+b*b),4);
```

**Example 3.1 of project with mathematical expression**

The task: *Enter* ***x*** *from the screen and calculate*: $Y = \dfrac{4x^2 - 3^x}{2x^2+1} + \dfrac{\ln x}{2x+3}$

The form will look like in fig. 3.1:

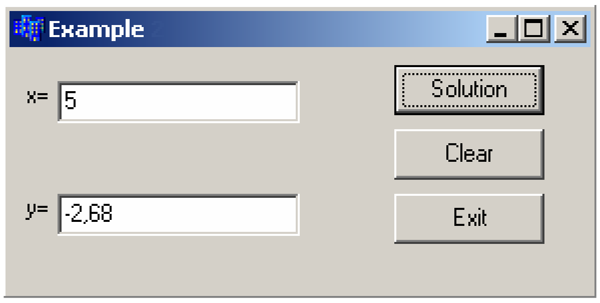


Fig. 3.1 Form of the project with result

The complete program is:

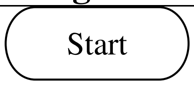
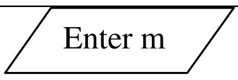
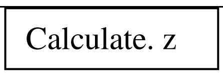
```
//---------------------------------------------------------------------------
#include <vcl.h>
#include <math.h>     //for math functions
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//"Calculate" button
//---------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
float x, y;                       //declare float variables
x=StrToFloat(Edit1->Text);        //read the value of x from Edit1
y=(4*x*x-pow(3,x))/(2*x*x+1)+log(x)/(2*x+3);
    //calculate y
Edit2->Text =FormatFloat("0.00",y);//output the result in Edit2
}
//---------------------------------------------------------------------
//"Clear" button
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Edit1->Clear();
Edit2->Clear();
}
//---------------------------------------------------------------------
//"Exit" button
void __fastcall TForm1::Button3Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------
```

## Flowcharts

For visualization the algorithm can be presented in the form of flowcharts by means of the special standard figures. Some of these figures are shown in Table 3.1.

**Table 3.1 Some of standard figures which are used in flowcharts**

| **figure** | **description** |
|---|---|
|  | Begin or finish program |
|  | Input or output values |
|  | Calculations and other actions |

Flowchart of example 3.1 is shown in fig. 3.2:

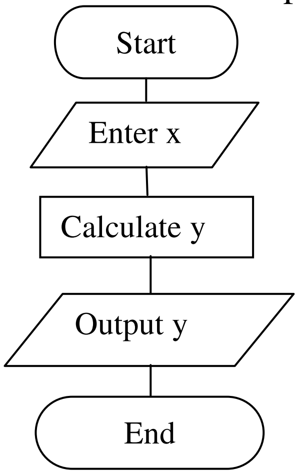


Fig. 3.2 Flowchart of the program (example 3.1)

**Constants**

Constants are expressions with a fixed value. If the value must not change in a program, it would be declared as a constant. Examples:

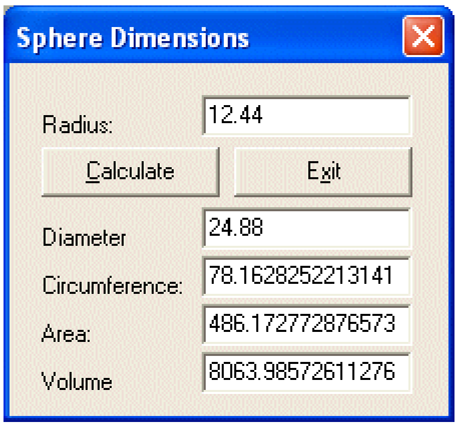
```
const int n=10;    //integer constant n has value 10
const float d=2.5;     //float constant d has value 2.5
```

The math.h library defines many constants. One of them is: M_PI. Its value is $\pi \approx 3.14$.

**Example 3.2**

The task: *Input the radius of the sphere and calculate the values of the sphere dimensions: diameter, circumference, area and volume.*

The form with results is shown in fig 3.3:



Radius: 12.44
Diameter 24.88
Circumference: 78.1628252213141
Area: 486.172772876573
Volume 8063.98572611276

Fig. 3.3 Form with results

The text of the program:

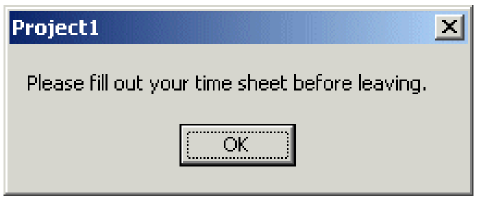
```
#include <math.h>
//-----------------------------------------------------------------------
//"Calculate" button
void __fastcall TForm1::btnCalculateClick
                (TObject *Sender)
{
double Rad, Diam, Circumf, Ar, Vol;
Rad = StrToFloat(Edit1->Text);
Diam = Rad * 2;
Circumf = Rad * 2 * M_PI;
Ar = Rad * Rad * M_PI;
Vol = pow(Rad, 3)* 4.00 * M_PI / 3;
Edit2->Text = Diam;
Edit3->Text = Circumf;
Edit4->Text = Ar;
Edit5->Text = Vol;
}
//---------------------------------------------------------------------------
```

## Message box

**A message box** is a relatively small dialog box used to display a message and provide one or more buttons. A message box is used to provide information to the user or to request a decision (by the user). By clicking on one of the buttons, the user makes a decision and the program continues.

The **ShowMessage()** function provides the most fundamental message box of Borland's applications. This function takes one string argument and does not return any value. It displays a message to the user. He can close it by clicking the OK button. Here is an example (fig. 3.4).

```
ShowMessage("Please fill out your time sheet before leaving.");
```


Project1

Please fill out your time sheet before leaving.

Fig. 3.4 Message box

## *Console application*

The **console** is a screen monitor and the keyboard considered to be a uniform device. The console application is the program for which the input device is the keyboard, and the output device is the monitor working in a mode of the character information (the character, digit and special signs).

Console application does not have visual forms.

To create a console application, you may use C++ Builder:

1. Start Borland C++ Builder. From the main menu, click File – New – Other (or File – New). You will see the window as in Fig. 3.5.

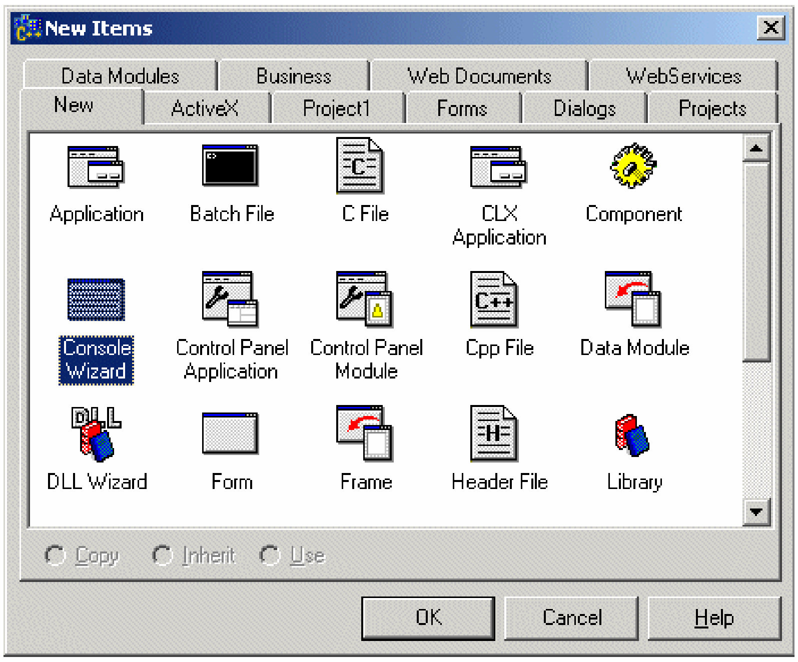
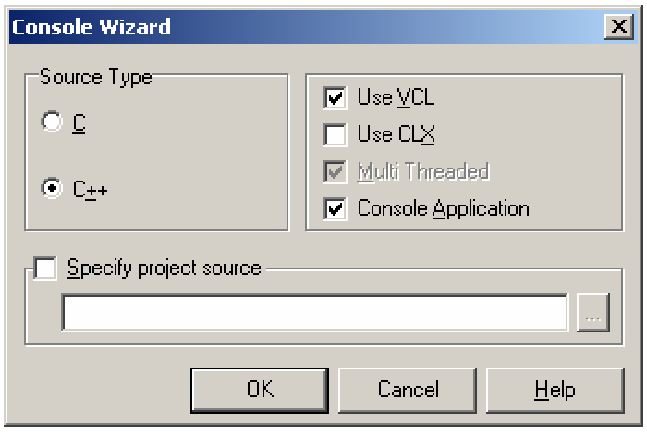


Fig. 3.5 New Items dialog box

2. From the New Items dialog box, click the Console Wizard icon and click OK.
3. In the Console Wizard dialog, make sure the C++ radio button is selected (fig. 3.6):


Console Wizard

Source Type
- ( ) C
- (•) C++

- [✓] Use VCL
- [ ] Use CLX
- [✓] Multi Threaded
- [✓] Console Application

[ ] Specify project source

Fig. 3.6 Console Wizard dialog box

4. Click OK.

Every console application starts with a function called **main()**. Notice that the console application has no form. When we run the console application, we see a black window instead of the form.

## Console input/output

The execution of a console application occurs in a black screen. To enter data, it is necessary to write a special command. At execution, this commands the program to stop and wait, until the user enters data and presses the Enter key. An output of results is made in the same black screen and always requires remarks and hints to the user.

To output text in the console window, we use the **cout<<** command and write text in inverted commas:

```
cout<<"text";
```

To output the value of a variable, we also use the **cout<<** command and write a variable without inverted commas. For example, to output the value of variable ***x***, we write:

```
cout<<x;
```

To output the variable ***x*** with a hint, we write:

```
cout<<″Value of variable x is: ″<<x;
```

If we want to move the cursor on a new line after the output of the information, it is necessary to write **<<endl** at the end of the command:

```
cout<<″Value of variable x is: ″<<x<<endl;
```

To input the value of a variable ***x***, we use the **cin>>** command:

```
cin>>x;
```

After the program finishes its work, the black window closes. Usually it happens so quickly that we do not have the opportunity to see results. To pause the program we must write **getch** at the end of it before `return 0`:

```
getch();
```

The same action is when we use cin.get().

**Notice!** To use cin and cout we must write

```
#include <iostream.h>
```

at the beginning of the program.

To use getch() we must write

```
#include <conio.h>
```

at the beginning of the program.

**Example 3.2 of console program**

The following program demonstrates console output:

```
// Listing 2.2 using cout
#include <iostream.h>
#include <conio.h>
int main()
{
cout << "Hello there.\n";
cout << "Here is 5: " << 5 << "\n";
cout << "The manipulator endl writes a new line to the screen."
     <<endl;
cout << "Here is a very big number:\t" << 70000 << endl;
cout << "Here is the sum of 8 and 5:\t" << 8+5 << endl;
cout << "Here's a fraction:\t\t" << (float) 5/8 << endl;
cout << "And a very very big number:\t" << (double) 7000 * 7000
     <<endl;
getch ();
return 0;}
```

The result of this program:

```
Hello there.
Here is 5: 5
The manipulator endl writes a new line to the screen.
Here is a very big number:      70000
Here is the sum of 8 and 5:     13
Here's a fraction:              0.625
And a very very big number:     4.9e+07
Don't forget to replace Jesse Liberty with your name...
Jesse Liberty is a C++ programmer!
```

On line **2**, the statement `#include <iostream.h>` causes the iostream.h file to be added to your source code. This is required if you use `cout` and its related functions.

On line **3**, the statement `#include <conio.h>` causes the conio.h file to be added to your source code. This is required if you use `getch()` function.

On line **6**, the simplest use of `cout`, printing a string or series of characters, is shown. The symbol '**\n**' is a special formatting character. It tells `cout` to print a newline character to the screen.

Three values are passed to `cout` on line **7**, and each value is separated by the insertion operator. The first value is the string "Here is 5: ". Note the space after the colon. The space is part of the string. Next, the value 5 is passed to the insertion operator and the newline character (always in double quotes or single quotes). This causes the line

```
Here is 5: 5
```

to be printed to the screen. Because there is no newline character after the first string, the next value is printed immediately afterwards. This is called concatenating of two values.

On line **8**, an informative message is printed, and then the manipulator `endl` is used. The purpose of `endl` is to write a new line on the screen.

On line **9**, a new formatting character, **\t**, is introduced. This inserts a tab character and is used on lines 8-12 to line up the output. Line 9 shows that not only integers, but long integers as well can be printed. Line **10** demonstrates that `cout` will do simple addition. The value of 8+5 is passed to `cout`, but 13 is printed.

On line **11**, the value 5/8 is inserted into `cout`. The term `(float)` tells `cout` that you want this value evaluated as a decimal equivalent, and so a fraction is printed. On line **12** the value 7000 * 7000 is given to `cout`, and the term `(double)` is used to tell `cout` that you want this to be printed using scientific notation.

**Example 3.3 of console application with mathematical expression**

The task: *Enter **k** from the screen and calculate:*

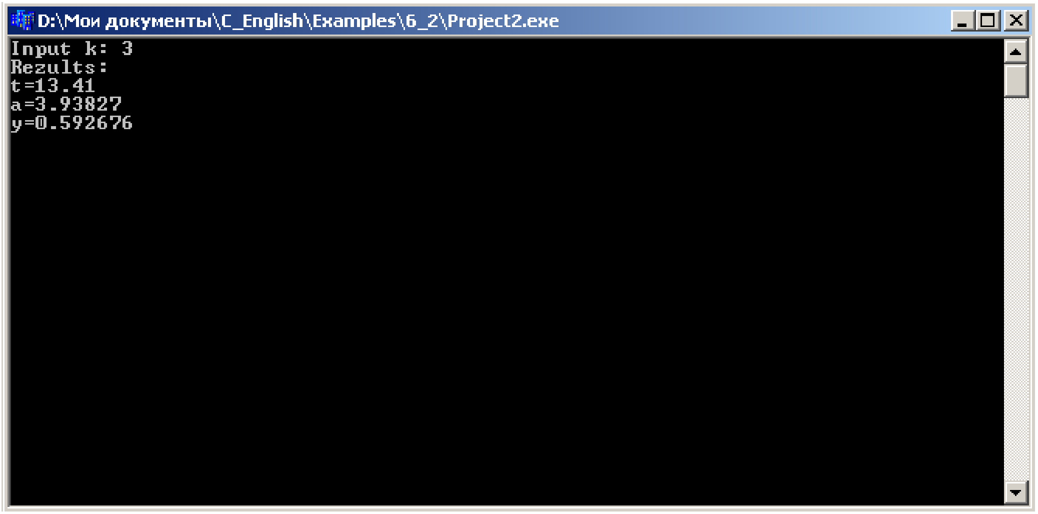
$y = \sin^4(a^2 + b^2); a = \sqrt{b + t}; t = b^2 + k^3$,

*where constant b=2.1.*

The complete program:

```
//-----------------------------------------------------------------------
#include <iostream.h>
#include <conio.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop
//-----------------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
    const double b=2.1;
     double a, k, t, y;
     cout<<"Input k: ";    //print a prompt to input k
     cin>>k;               //input k
    //calculation
    t=b*b+k*k;
     a=sqrt(b+t);
     y=pow(sin(a*a+b*b),4);
     //output rezults
    cout<<"Rezults:"<< endl;
     cout<<"t="<<t<<endl;
     cout<<"a="<<a<<endl;
     cout<<"y="<<y<<endl;
     getch();              //pause the program
     return 0;
}
//-----------------------------------------------------------------------
```

Results of the program are shown in fig. 3.7.


D:\Мои документы\C_English\Examples\6_2\Project2.exe

```
Input k: 3
Rezults:
t=13.41
a=3.93827
y=0.592676
```

Fig. 3.7 Results of example 3.3

If you run this program again, you will see the previous results in the black screen. If you want to clear the screen every time you run the program, you must use the **clrscr()** command from conio.h:

```
clrscr();
```

There is one more way to input and output in console. You may use the **scanf** and **printf** commands from stdio.h.

## printf

The **printf** command can be used to output text or data in the console window. It allows outputting formatted data. The number of required arguments varies, but the first argument you pass should be a **string**.

Recall that a string must be surrounded by double quotation marks. For example, output the string constant "Hello World!":

printf("Hello World!\n");

The '\n' is the **new line** character and acts like a line break.

Printing out a variable, requires you to embed a format specifier in your text string, and to pass extra arguments to the printf function. An example:

```
printf("x equals %d \n", x);
```

If x=5, the result of this statement is:

```
x equals 5
```

This statement prints out the value of ***x***. We had to pass the value of ***x*** into the printf function. When you pass arguments to functions, you separate each one with a comma – here, "x equals %d \n" is the first argument, and ***x*** is the second argument.

There are several format specifiers. The one you use should depend on the type of variable you wish to print out. Here are common ones:

| **Format Specifier** | **Type** |
|---|---|
| %d (or %i) | int |
| %c | char |
| %f | float |
| %lf | double |
| %s | string |
| %x | hexadecimal |

## Two or More Format Specifiers

You may use as many format specifiers as you want with printf, just as long as you pass the correct number of arguments.

The sequence of the arguments matters. The first one should correspond to the first format specifier in the string and so on. For example:

```
    printf("a=%d, b=%d, c=%d\n", a, b, c);
```

If ***a***, ***b***, and ***c*** are integers, this statement will print the values in the correct order. Rewriting the statement as

```
    printf("a=%d, b=%d, c=%d\n", c,a,b);
```

will still cause the program to compile OK, but the values of ***a***, ***b***, and ***c*** will be displayed in the wrong order!

## Scanf

**Scanf** allows reading variable values from the keyboard. It takes at least two arguments. The first one is a string that can consist of format specifiers. The rest of the arguments should be variable names preceded with the address-of operator. For example:

```
    printf("Enter a number ");
    printf(" and press Enter: ");
    scanf("%d", &a);
```

**&** is known as the **address-of** operator. You will use it a lot more when you meet **POINTERS** at a later date.

We can picture the previous scanf statement by this: "Read in an integer from the input string, then go to the address of the variable called ***a***, and put the value there". Remember that a variable is like a container in your computer's memory; each one has a different address.

As with printf, the number of arguments after the string argument should match the number of format specifiers contained in that string. Similarly, the type of the format specifier should match the type of the corresponding variable. The ordering of the variables also matters.

If you have multiple format specifiers within the string argument of scanf, you can input multiple values. All you need to do is separate each format specifier with a **delimiter**, a string that separates variables. For convenience, the delimiter should be one character that is a punctuation mark, like a comma or a space. As a default, scanf stops reading in a value when a space, tab or Enter is used.

Consider

```
    scanf("%d %d", &x, &y);
```

(assume that ***x*** and ***y*** have been declared beforehand!). If we enter:

***1 2***

and press Enter, ***1*** will get assigned to ***x***, and ***2*** will get assigned to ***y***. But if we enter

***1***, ***2***

and press Enter, ***x*** would equal ***1***, but ***y*** would not get assigned ***2*** because scanf was not expecting a comma in the input string.

Now consider:

```
    scanf("%d, %d, %d", &x,&y,&z);
```

If we enter

***1 2 3***

and press enter, ***1*** will get assigned to ***x*** but ***2*** and ***3*** will not get assigned to ***y*** or ***z***, simply because we do not separate the numbers with commas.

Scanf ignores spaces, tabs and carriage returns immediately after the delimiters.

Do not put a space, tab, or carriage return before the delimiter!

Note that you should not put a delimiter after the last format specifier!

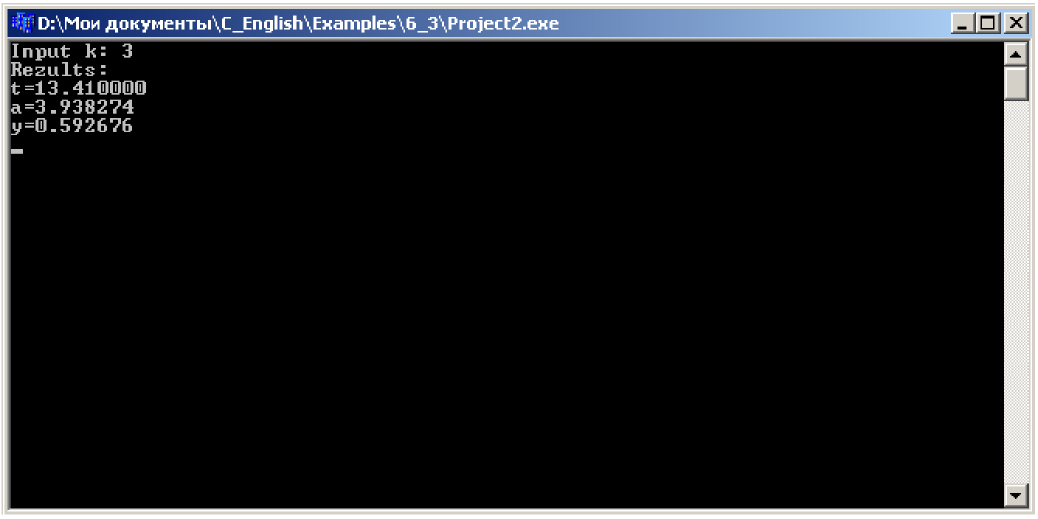
**Example 3.4 of console application with mathematical expression and printf/scanf**

The task: *Enter **k** from the screen and calculate:*
$y = \sin^4(a^2 + b^2); a = \sqrt{b+t}; t = b^2 + k^3$, *where constant b=2.1. Use printf and scanf.*

The complete program:

```
//----------------------------------------------------------------------
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop
//----------------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
        const double b=2.1;
        double a, k, t, y;
        printf("Input k: ");
        scanf("%lf",&k);
        t=b*b+k*k;
        a=sqrt(b+t);
        y=pow(sin(a*a+b*b),4);
        printf("Rezults:\n");
        printf("t=%f\n",t);
        printf("a=%f\n",a);
        printf("y=%f\n",y);
        getch();
        return 0;
}
//----------------------------------------------------------------------
```

Results of the program are shown in fig. 3.8.



```
Input k: 3
Rezults:
t=13.410000
a=3.938274
y=0.592676
```

Fig. 3.8 Results of the program

If you want to see two digits after a decimal point, you must write precision:

```
            printf("t=%5.2f\n",t);
```

where 5 is the total number of digits (2 before decimal point, 2 after it and the point itself) and 2 is the number of digits after point.

## Lecture 4. Conditions. If and switch conditional statements

A program usually is not limited to a linear sequence of instructions. During its process it may bifurcate, repeat code, or make decisions (fig. 4.1). For that purpose, C++ provides control structures that serve to specify what and how our program has to perform.

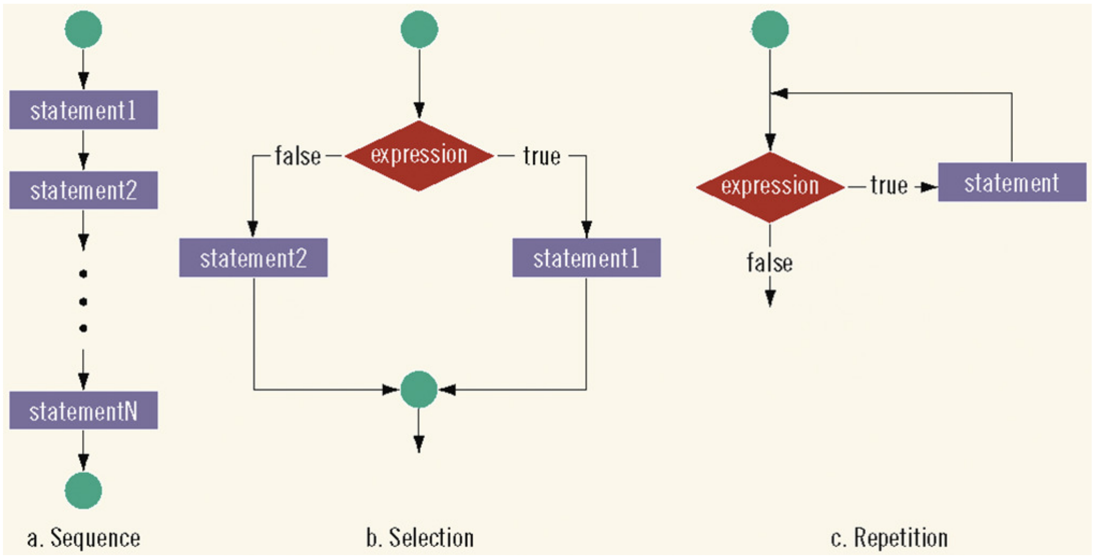


Fig. 4.1 Flow of execution variants

### *Conditions*

C++ provides six relational operators for comparing numeric quantities (table 4.1). Relational operators evaluate to **1** (representing the ***true*** outcome) or **0** (representing the ***false*** outcome).

**Table 4.1 Relational operations in C++**

| **Operator** | **Name** | **Example** |
|---|---|---|
| == | equality | 5==5 *//gives 1* |
| != | inequality | 5!=5 *//gives 0* |
| < | less than | 5 < 5.5 *// gives 1* |
| <= | less than or equal | 5 <= 5 *// gives 1* |
| > | greater than | 5 > 5.5 *// gives 0* |
| >= | greater than or equal | 6.3 >= 5 *// gives 1* |

It is possible to unite two or more conditions in one by means of logical **AND**, **OR**, and **NOT**:

**&&** – logical **AND**
|| – logical **OR**
**!** – logical **NOT**
**Examples of conditions:**
`x>0` (x is a positive number)
`x==9` (x equals to 9)
`x!=y` (x is not equal y)

`x%2==0` (x is an even number)
`x%6!=0` (6 is not multiple to x)
`x>0&&x<10` (0<x<10, x is greater than 0 **and** less than 10)
`x<0 || x>10` (x<0 **or** x>10)
`!x` (x==0)

Priority of relational operations is lower than arithmetical operations (look the Application 2).

## *If statement*

The **if** statement allows your program to make a decision. Its syntax is:

```
if (condition)
 { action1 }
else
 { action2 }
```

If the **condition** in parenthesis is true, all code inside the first braces is executed, if it is not, the code inside the first braces is ignored and the code inside the second braces after **else** is executed.

**Condition** is a value or an expression that is used to determine which code block is executed, and the curly braces act as "begin" and "end" markers. Note that conditions always are written in parenthesis.

If you have only one statement in **action1** or **action2**, then the correspondent braces are not required.

Pay your attention, that commands in **action1** are written a little more to the right of the **if** statement and commands in **action2** are written a little more to the right of the **else** statement. It is done for visual selection of these commands. The program becomes clearer, it is easier to find and correct errors in it.

**Examples** of use conditions in the program:

**1.** Calculate the quotient of two numbers (if ***y*** is not 0, we calculate the quotient).

```
if (y!=0)
    r=x/y;
```

**2.** Calculate the square root of the variable ***x***, if it is possible (if ***x*** is not negative, we calculate the square root, otherwise we output the message with error).

```
if (x>=0)
    y=sqrt(x);
else
    ShowMessage("Can not calculate");
```

**3.** If variables ***x*** and ***y*** are equal, calculate their product. Otherwise, calculate an absolute value of their difference.

```
if(x==y) r=x*y;
else r=fabs(x-y);
```

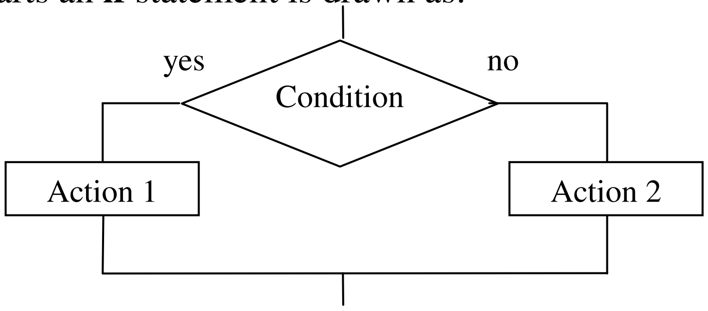
In the previous examples braces are not necessary.

**4.** If absolute values of x and y are less than 10, increase them twice; otherwise reduce them twice.

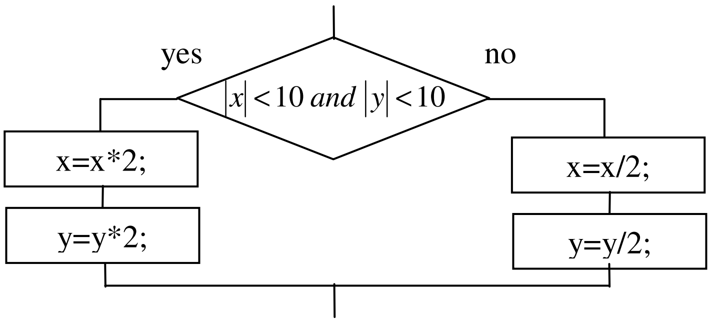
```
if (fabs(x)<10&&fabs(x)<10)
  { x=x*2;
    y=y*2; }
else
  { x=x/2;
    y=y/2; }
```

In this example two statements must be executed if the condition is true. Therefore, we must use braces.

Statements in braces are called a **block of statements.**
In the flowcharts an **if** statement is drawn as:



The flowchart of the last example is:



**Nesting if – else – if statement**

There may be any statements instead of **Action1** or **Action2**, including another **if** statement. Example: to define the sign of the variable ***x*** (it may be positive, negative or zero).

```
if (x>0) ShowMessage (“positive”);
else
    if (x<0) ShowMessage(“negative”);
    else ShowMessage (“zero”);
```

First `else` means that ***x*** is a negative number or zero. Therefore, we should check both of these possibilities.

**Example 4.1 of program with branching**

The task: *Convert marks from 100-balls system to 5-balls system.*

The program with if statements:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    int m100, m5;
    m100=StrToInt(Edit1->Text);
    if (m100>=95) m5=5;
    if (m100>=80 && m100<95) m5=4;
    if (m100>=60 && m100<80) m5=3;
    if (m100<60) m5=2;
    Edit2->Text=IntToStr(m5);
}
```

The same program with nesting if – else – if.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    int m100, m5;
    m100=StrToInt(Edit1->Text);
    if (m100>=95) m5=5;
    else
         if (m100>=80) m5=4;
         else if (m100>=60) m5=3;
                else m5=2;
    Edit2->Text=IntToStr(m5);
}
```

In the second variation of the program, it is not necessary to check the right border of the interval.

**Conditional (Ternary) Operator**

The conditional operator **(?:)** is C++'s only ternary operator; that is, it is the only operator to take three terms.

The conditional operator takes three expressions and returns a value:

```
(expression1) ? (expression2) : (expression3)
```

This line is read as "If **expression1** is true, return the value of **expression2**; otherwise, return the value of **expression3**." Typically, this value would be assigned to a variable.

**Example 4.2 A demonstration of the conditional operator**

This console program shows an **if** statement rewritten using the conditional operator.

```
 1 | #include <iostream.h>
 2 | int main()
 3 | {
 4 |   int x, y, z;
 5 |   cout << "Enter two numbers.\n";
 6 |   cout << "First: ";
 7 |   cin >> x;
 8 |   cout << "\nSecond: ";
 9 |   cin >> y;
10 |   cout << "\n";
11 |
12 |   if (x > y)
13 |     z = x;
14 |   else
15 |     z = y;
16 |
17 |   cout << "z: " << z;
18 |   cout << "\n";
19 |
20 |   z =  (x > y) ? x : y;
21 |
22 |   cout << "z: " << z;
23 |   cout << "\n";
24 |   return 0;
25 |  }
```

Result of the program:

```
Enter two numbers.
First: 5

Second: 8

z: 8
z: 8
```

Three integer variables are created: ***x***, ***y***, and ***z***. The first two are given values by the user. The **if** statement on line 12 tests to see which is larger and assigns the larger value to ***z***. This value is printed on line 17.

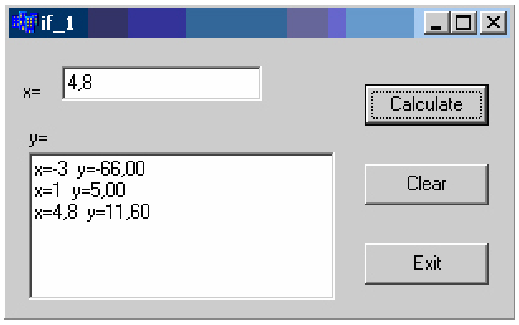
The conditional operator on line 20 makes the same test and assigns ***z*** the larger value. It is read like this: "If ***x*** is greater than ***y***, return the value of ***x***. Otherwise, return the value of ***y***." The value returned is assigned to ***z*** and printed on line 22. As you can see, the conditional statement is a shorter equivalent to the **if - else** statement.

## Example 4.3 of program with branching

The task: *Enter a value of **x** and calculate the value of **y**:*

$$Y=\begin{cases}2x^3+4x & x\le -1\\ x+4 \quad \text{when} & -1<x<3\\ 2x+2 & x\ge 3\end{cases}$$

The form of the project with results is in fig. 4.2:



```
x=-3  y=-66,00
x=1  y=5,00
x=4,8  y=11,60
```

Fig. 4.2 The form of the project and results (example 4.3)

The code of the program:

```
//---------------------------------------------------------------------------
#include <math.h>
…
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    double x, y;
    x=StrToFloat(Edit1->Text);
    if(x<=-1) y=2*pow(x,3)+4*x;
    if(x>-1 && x<3) y=x+4;
    if(x>=3) y=2*x+2;
    Memo1->Lines->Add("x="+FloatToStr(x)+"
    y="+FormatFloat("0.00",y));
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    Edit1->Clear();
    Memo1->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
    Close();
}
//---------------------------------------------------------------------------
```

**Example 4.4 of program with branching**

The task: *Input coordinates of points A(x1, y1), B(x2, y2), C(x3, y3) from the screen. Define whether these points are on one line. Output the answer with the message.*

The form of the project and results (for different inputted coordinates) are shown in fig. 4.3 and 4.4:

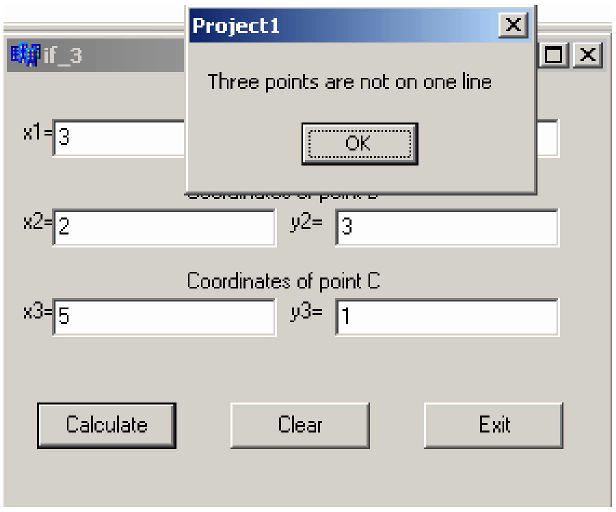


Fig. 4.3 The form of the project with negative result

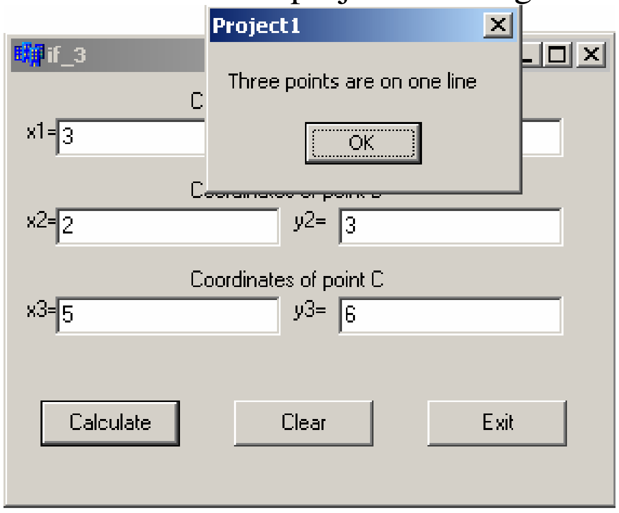


Fig. 4.4 The form of the project with positive result

The code of "Calculate" button:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    double x1, y1, x2, y2, x3, y3, a, b;
    x1=StrToFloat(Edit1->Text);
    y1=StrToFloat(Edit2->Text);
    x2=StrToFloat(Edit3->Text);
    y2=StrToFloat(Edit4->Text);
    x3=StrToFloat(Edit5->Text);
    y3=StrToFloat(Edit6->Text);
    if ((y2-y1)*(x3-x1)==(x2-x1)*(y3-y1))
        ShowMessage("Three points are on one line");
    else
        ShowMessage("Three points are not on one
    line");
}
```
The flowchart of the "Calculate" button is shown in fig. 4.5:

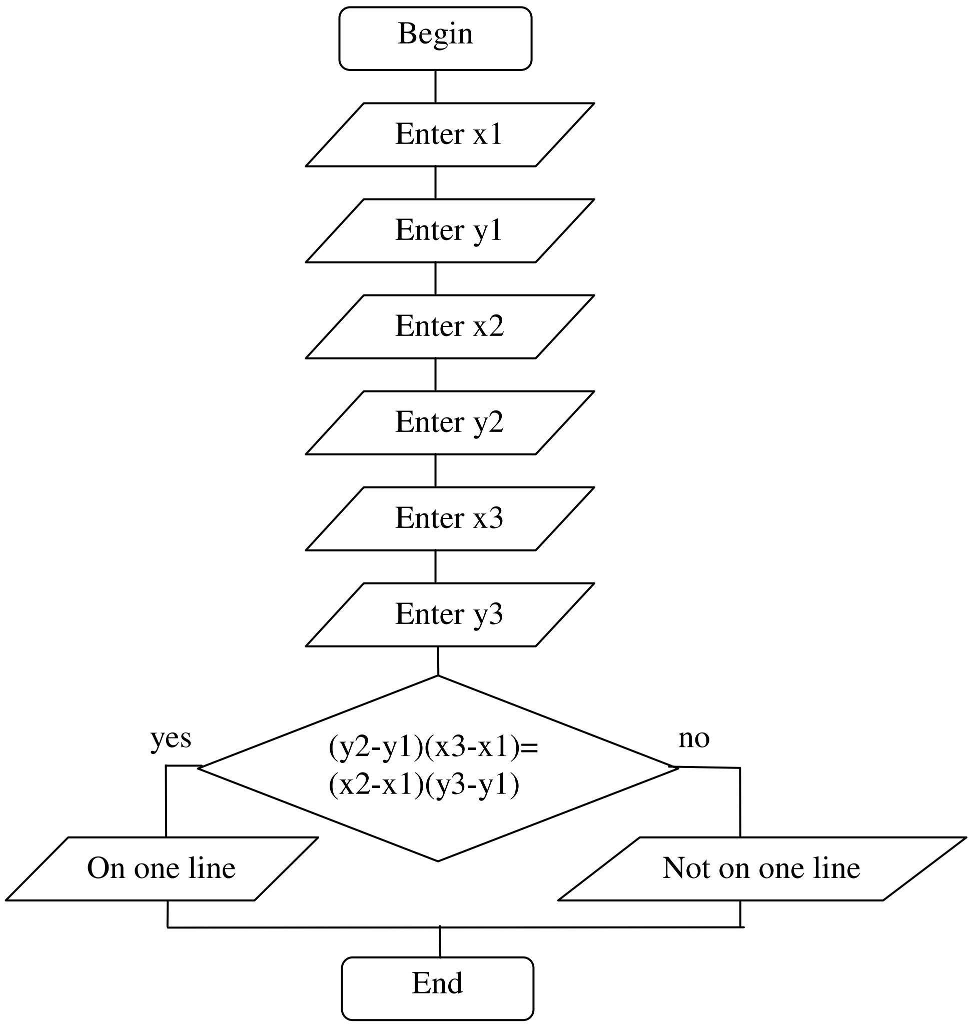


Fig. 4.5 The flowchart of the "Calculate" button (example 4.4)

## ***The switch statement***

The next branching statement is called a **switch** statement. A switch statement is used in place of many **if** statements. The **switch statement** allows selecting the value of expression from one of several integer values and then executing actions accordingly. The switch statement has the following syntax:

```
switch(expression){
   case constant1: action_1; break;
   case constant2: action_2; break;
    ...
   case constantn: action_n; break;
   default: action_n+1; // execute if expression is none of the above
}
```

The first *expression* (called the switch tag) is evaluated, and the outcome is compared to each of the numeric constants (called case labels) in the order in which they appear, until a match is found. The statements following the matching case are then executed. Each case may be followed by a zero or more statements (not just one statement). If **break** is absent, the commands following the next cases are also executed. The final default case is optional and is exercised if none of the earlier cases provide a match.

**Example 4.5 of program with switch statement**

The task: *Input the week day order number and show message with its name (1-Monday and so on).*

The fragment of the program:

```
int n=StrToInt(Edit1->Text);
switch (n){
    case 1: ShowMessage ("Monday"); break;
    case 2: ShowMessage ("Tuesday"); break;
    case 3: ShowMessage ("Wednesday"); break;
    case 4: ShowMessage ("Thursday"); break;
    case 5: ShowMessage ("Friday"); break;
    case 6: ShowMessage ("Saturday"); break;
    case 7: ShowMessage ("Sunday"); break;
    default: ShowMessage ("The number is not
correct"); break;
}
```

Note that each case ends with a **break** statement. It stops the execution of a program block. In this example if we have found a necessary case, **break** will prevent us from checking other cases.

**Example 4.6 of program with switch statement**

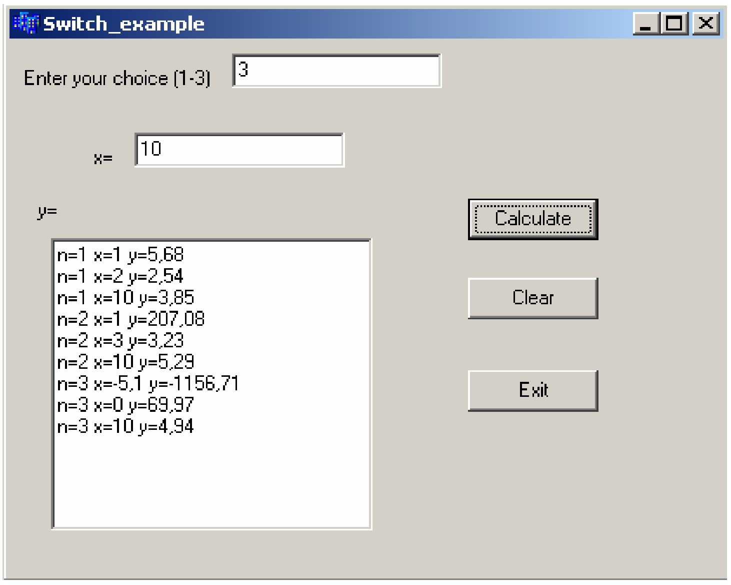
The task: *Make a flowchart and a project of the program which calculates the values of function **y** for three variations of parameter values according to given formulas.*

$$y = \begin{cases} a^5 + arctg\left(\cos^2(a+b)x\right) + \sqrt{\sin^{-2} x}, & \text{if } \quad x \le a \\ \sqrt{\left(a^3 - bx\right)^2 + 7} + \cos^3 zx, & \text{if } \ a < x < \ln b \\ \ln\left|a + bx + zx^2\right|, & \text{if } \quad x \ge \ln b \end{cases}$$

1. $a = 1.3;\ b = 4.9;\ z = \sin\left|tg(bx)\right|$
2. $a = 8.9;\ b = 6.4;\ z = tg\left|\sin(bx)\right|$
3. $a = -4.1;\ b = 5.3;\ z = \cos\left|tg(bx)\right|$

The user must input ***x*** and his choice ***n*** (integer numbers 1, 2 or 3). Variables ***a***, ***b***, and ***z*** must be calculated in dependence on this choice. When the values of ***a***, ***b***, ***z*** are defined, the program calculates the value of ***y***.

The form of the project with results is shown in fig. 4.6:



```
Enter your choice (1-3)  3
x=  10
y=
n=1 x=1 y=5,68
n=1 x=2 y=2,54
n=1 x=10 y=3,85
n=2 x=1 y=207,08
n=2 x=3 y=3,23
n=2 x=10 y=5,29
n=3 x=-5,1 y=-1156,71
n=3 x=0 y=69,97
n=3 x=10 y=4,94
```

Fig. 4.6 The form with results

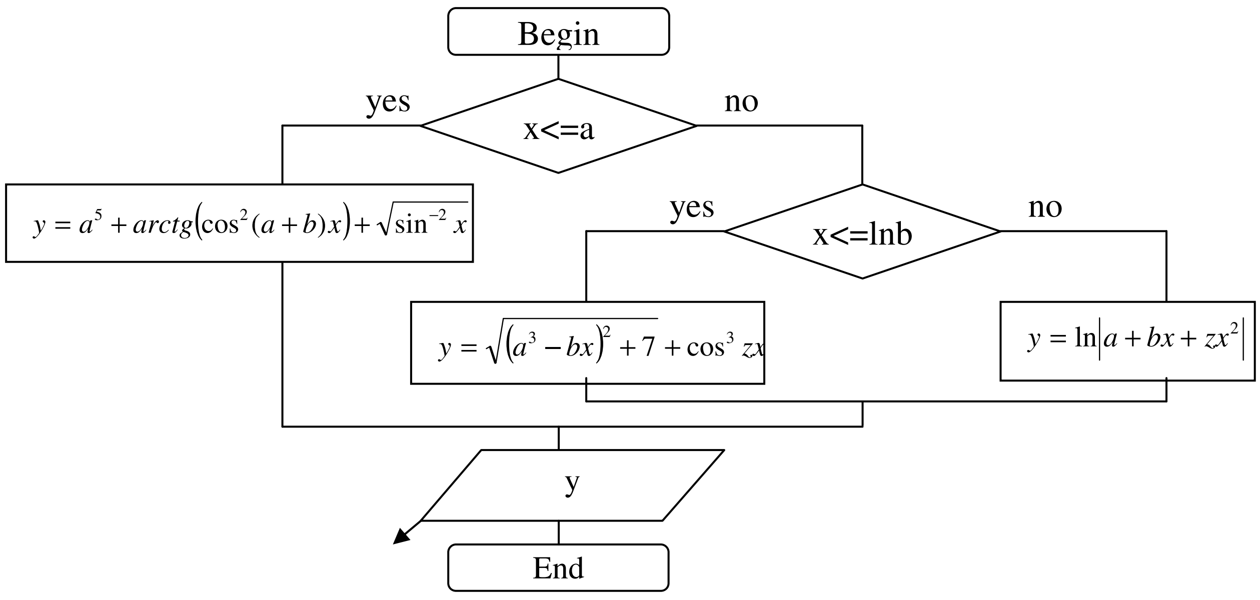
The code of the program:

<u>I method (with function which calculates y)</u>

```
#include <math.h>
......
//-----------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    Edit1->Clear();
    Memo1->Clear();
}
//-----------------------------------------------------------------------
//definition of function func
double func(double x, double a, double b, double z)
{
double y;
if (x<=a)
    y=pow(a,5)+atan(pow(cos((a+b)*x),2))+
        sqrt(pow(sin(x),-2));
else
    if (x<=log(b))
        y=sqrt(pow(pow(a,3)-b*x,2)+7)+
            pow(cos(z*x),3);
    else
        y=log(fabs(a+b*x+z*x*x));
return y;
}
//-----------------------------------------------------------------------
//Button1
void __fastcall TForm1::Button1Click(TObject *Sender)
{
```

```
int n;
double x,Y,a,b,z;
x=StrToFloat(Edit2->Text);
n=StrToInt(Edit1->Text);
//calculation of a, b, z in dependence on the choice of  n
switch(n){
    case 1:  a=1.3; b=4.9;
            z=sin(fabs(tan(b*x))); break;
    case 2:  a=2.9; b=6.4;
            z=tan(fabs(sin(b*x))); break;
    case 3:  a=-4.1; b=5.3;
            z=cos(fabs(tan(b*x))); break;
    default:
        ShowMessage("The choice is not correct");
}
Y=func(x, a, b,z);                //call of function func
Memo1->Lines->Add("n="+IntToStr(n)+"x="+
    FloatToStr(x)+" y="+FormatFloat("0.00",Y));
}
//-----------------------------------------------------------------------
```

The flowchart of subroutine is shown in fig. 4.7:



Begin
yes
x<=a
no
$y = a^5 + arctg\left(\cos^2(a+b)x\right) + \sqrt{\sin^{-2} x}$
yes
x<=lnb
no
$y = \sqrt{\left(a^3 - bx\right)^2 + 7 + \cos^3 zx}$
$y = \ln\left|a + bx + zx^2\right|$
y
End

Fig. 4.7 The flowchart of the subroutine (function evaluation)

The flowchart of the program is shown in fig. 4.8.

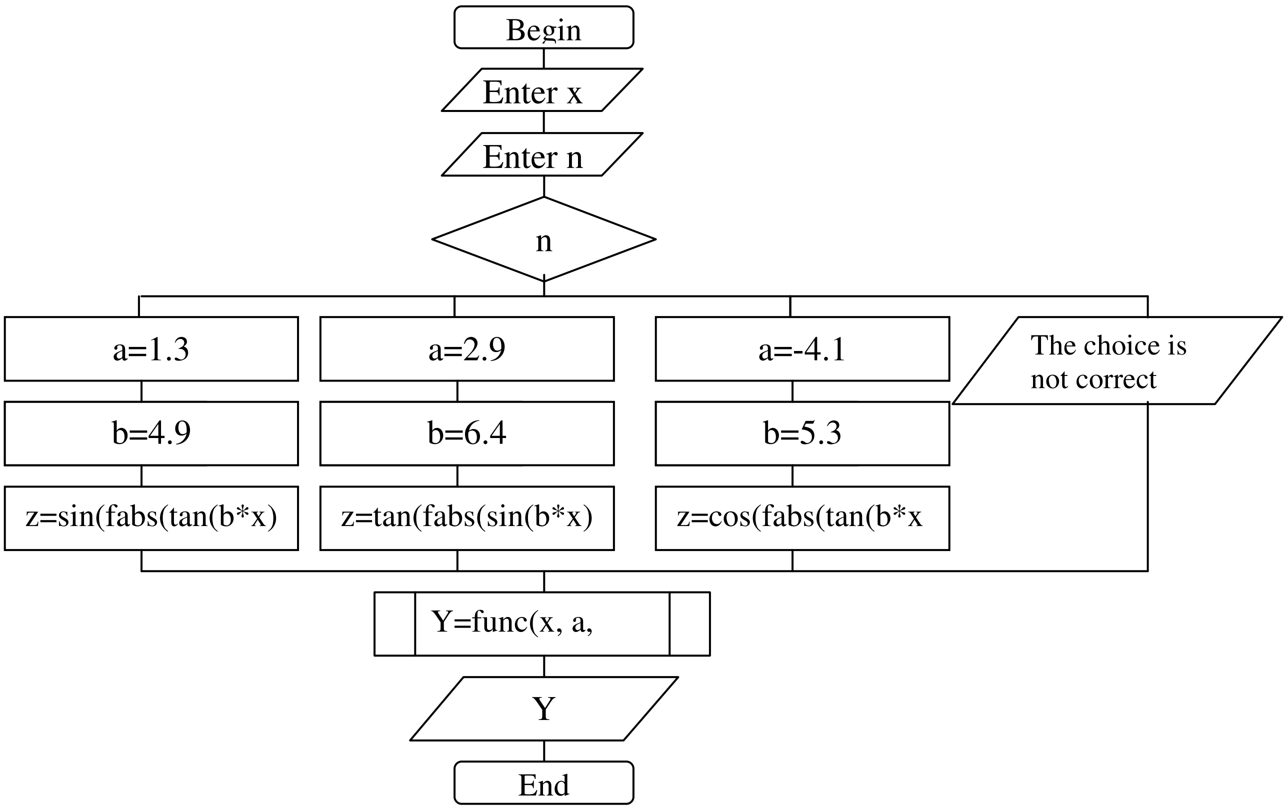


Fig. 4.8 The flowchart of the Button1Click function

<u>II method (without function which calculates y)</u>

```
//-----------------------------------------------------------------------
#include <math.h>
......
//-----------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
int n;
double x,y,a,b,z;
x=StrToFloat(Edit2->Text);
n=StrToInt(Edit1->Text);
//calculation a, b, z in dependence on the choice n
switch(n){
    case 1:  a=1.3; b=4.9;
             z=sin(fabs(tan(b*x))); break;
    case 2:  a=2.9; b=6.4;
             z=tan(fabs(sin(b*x))); break;
    case 3:  a=-4.1; b=5.3;
             z=cos(fabs(tan(b*x))); break;
    default:
         ShowMessage("The choice is not correct");
}
//calculation of value of y
if (x<=a)
   y=pow(a,5)+atan(pow(cos((a+b)*x),2))+
        sqrt(pow(sin(x),-2));
else
    if (x<=log(b))
       y=sqrt(pow(pow(a,3)-b*x,2)+7)+pow(cos(z*x),3);
    else
       y=log(fabs(a+b*x+z*x*x));
Memo1->Lines->Add("n="+IntToStr(n)+
" x="+FloatToStr(x)+" y="+FormatFloat("0.00",y));
}
//---------------------------------------------------------------------
```

Flowchart of the program is shown in fig. 4.9:

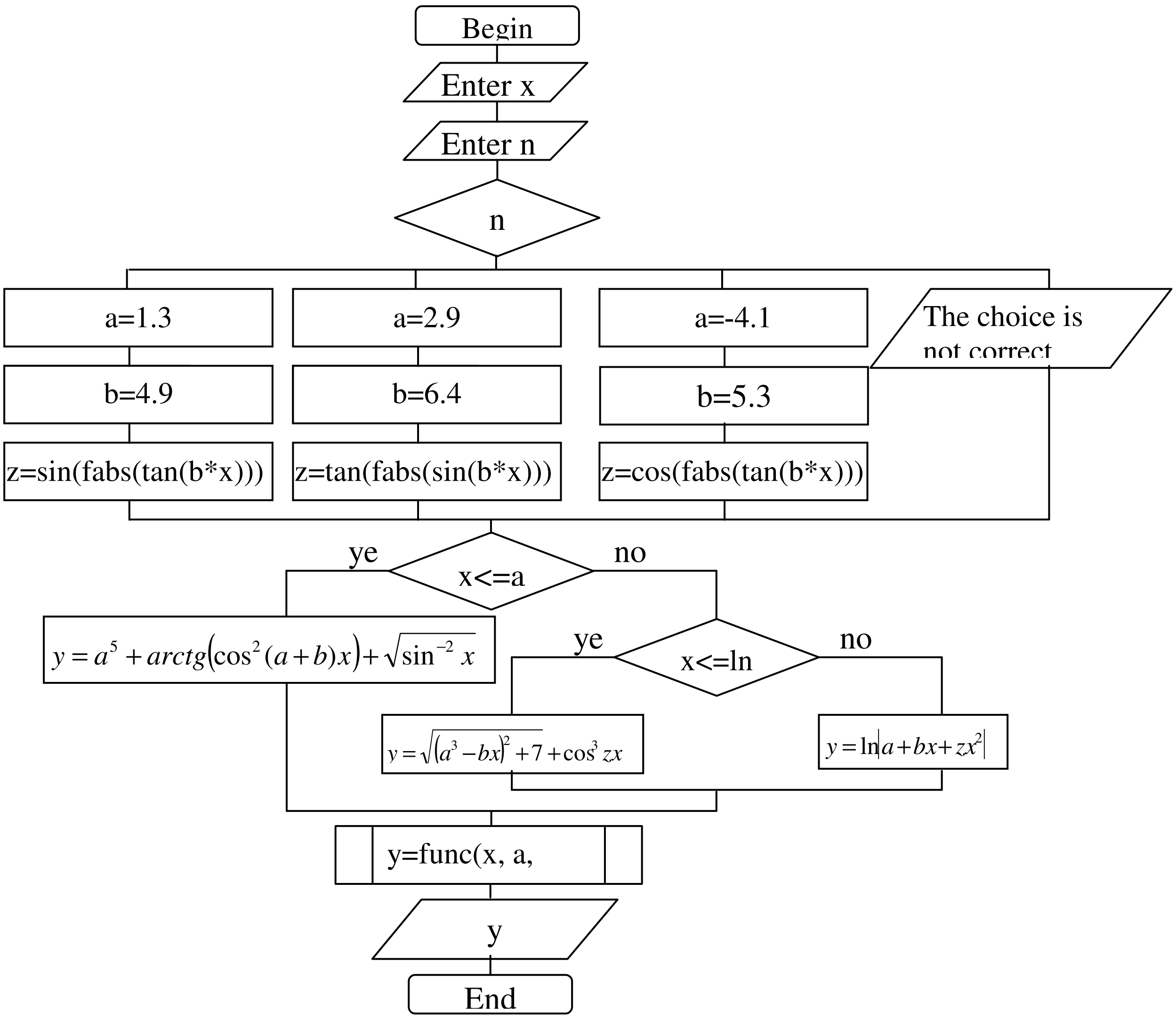
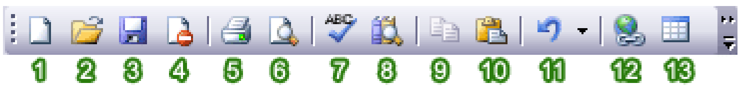


Fig. 4.9 The flowchart of the program

# Application 1 Description of Microsoft Word Standard and Formatting Toolbars

## The Standard Toolbar



1. **New Blank Document:**

   To begin a new document, click on the New Blank Document icon, shaped like a blank sheet of paper.

2. **Open**

   Clicking on this icon opens up a previously saved document on your computer.

3. **Save**

   Clicking on the Save icon saves the document you are currently working on. If you are saving a document for the first time, you can click on this button. However, if you want to save a new file from a preexisting document, then you must go to the menu bar and select File – Save As and give the file a new name. When working on any document, you should be sure to save frequently, so that you don't lose any work.

4. **Permission**

   Microsoft has enabled Information Rights Management (IRM) within the new version of Word, which can help protect sensitive documents from being copied or forwarded. Click this for more information and options.

5. **Print**

   Clicking on the Print icon automatically prints the document currently active in Word. If you wish to explore more print options, then go to the menu bar and select File – Print

6. **Print Preview**

   To get an idea of the appearance of your document in print before you actually print it out, you can click on this icon to view your document from a zoom-out distance.

7. **Spelling and Grammar**

   Clicking begins a review of your document in search of spelling and grammatical errors that may need to be corrected.

8. **Copy**

   Copy the current selection to the clipboard, which can then be pasted elsewhere in the document, or into a completely separate program/document.

9. **Paste**

   Clicking on the Paste button inserts the text that has been most recently added to the Clipboard (the text would have been added there by Cutting or Copying). With Paste, you can either insert the copied text into a document or replace selected text.

10. **Undo Typing**

    The Undo Typing button goes back and removes the last addition or change made to your document.

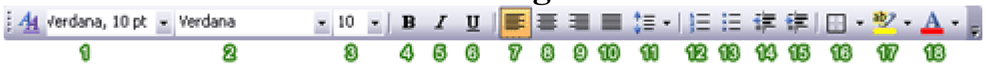
11. **Insert Hyperlink**

    You may find that you want to make links to a particular web site, web page, or some other kind of online file in your Word document. Using the Insert Hyperlink button, you can turn selected text into hyperlinks. When the icon is clicked, a window will appear that will allow you to insert the URL (web address) of the web page you want to link to. You can type in the URL yourself or insert a preexisting bookmark. Once the link is inserted, the link in your Word document can be clicked and the web page will open up in a web browser.

**12.Insert Table**

When this icon is clicked, a small window will appear in the form of a grid of squares. Use this window as a guide to indicate how many rows and columns you would like your table to contain. Once selected, a table will automatically appear in Word. Clicking the Tables and Borders button will allow you to modify the table. To modify an aspect of the table, select, or place the cursor in, the area and apply changes such as borders and colors.

**The Formatting Toolbar**



1. **Style**

   Styles in Word are used to quickly format portions of text. For example, you could use the “Normal” or “Default Paragraph Font” for the body text in a document. There are also three preset styles made for headings.

2. **Font**

   Font is a simple but important factor in Word documents. The choice of font (the style of the text itself) can influence the way others view documents, either on the screen or in print. For example, Arial font looks better on screen, while Times New Roman is clearer in print. To apply a font to text, select desired text with your cursor, and choose a font from the font drop down menu.

3. **Font Size**

   You may encounter times in which you need to display some text larger or smaller than other text. Selecting desired text with the cursor and choosing a font size from the drop down menu changes the size of text.

4. **Bold**

   Places the text in **bold**.

5. **Italic**

   Places the text in *italics*.

6. **Underline**

   Underlines the text.

7. **Align Left**

   Aligns the selection to the left of the screen/paper.

8. **Center**

   Aligns the selection to the center of the screen/paper.

9. **Align Right**

   Aligns the selection to the right of the screen/paper.

10. **Justify**

    Aligns the selection to both the left and right of the screen/paper.

11. **Line Spacing**

    Adjust the line spacing (single-spaced, double-spaced, etc.)

**12.Numbering**

Create a numbered list.

**13.Bullets**

Create an unordered, bulleted list.

**14.Decrease Indent**

Decreases the indentation of the current selection (to the left).

**15.Increase Indent**

Increases the indentation of the current selection (to the right).

**16.Outside Border**

Places a border around the current selection; click the drop-down for a wide selection of bordering options.

**17.Highlight**

Highlight the current selection; default color is yellow.

**18.Font Color**

Change the font color; the default/automatic color is black.

## Application 2 Precedence of operators

| Level | Operator | Description | Grouping |
|---|---|---|---|
| 1 | :: | scope | Left-to-right |
| 2 | () [] . -> ++ -- dynamic_cast static_cast reinterpret_cast const_cast typeid | postfix | Left-to-right |
| 3 | ++ -- ~ ! sizeof new delete | unary (prefix) | Right-to-left |
|  | * & | indirection and reference (pointers) |  |
|  | + - | unary sign operator |  |
| 4 | (type) | type casting | Right-to-left |
| 5 | .* ->* | pointer-to-member | Left-to-right |
| 6 | * / % | multiplicative | Left-to-right |
| 7 | + - | additive | Left-to-right |
| 8 | << >> | shift | Left-to-right |
| 9 | < > <= >= | relational | Left-to-right |
| 10 | == != | equality | Left-to-right |
| 11 | & | bitwise AND | Left-to-right |
| 12 | ^ | bitwise XOR | Left-to-right |
| 13 | \| | bitwise OR | Left-to-right |
| 14 | && | logical AND | Left-to-right |
| 15 | \|\| | logical OR | Left-to-right |
| 16 | ?: | conditional | Right-to-left |
| 17 | = *= /= %= += -= >>= <<= &= ^= \|= | assignment | Right-to-left |
| 18 | , | comma | Left-to-right |

## Application 3 The list of reserved words

The C++ compiler has a list of words reserved for its own use and you must not use any of these words to name your objects or functions. The reserved words are:

| C and C++ Reserved Words | | | | |
|---|---|---|---|---|
| auto | do | goto | signed | union |
| break | double | if | sizeof | unsigned |
| case | else | int | static | void |
| char | enum | long | struct | volatile |
| const | extern | register | switch | while |
| continue | float | return | typedef | |
| default | for | short | | |

Some of these words are used by Borland C++ Builder or could come in conflict with the libraries used in Microsoft Windows. Therefore, avoid using these additional words:

| C++ Reserved Words | | | | |
|---|---|---|---|---|
| asm | dynamic_cast | namespace | reinterpret_cast | try |
| bool | explicit | new | static_cast | typeid |
| catch | false | operator | string | typename |
| class | friend | private | template | union |
| cin | inline | protected | this | using |
| const_cast | interrupt | public | throw | virtual |
| cout | mutable | register | true | wchar_t |
| delete | | | | |

Here are other words you should avoid using when programming in Borland C++ Builder:

| Other Reserved Words | | | | |
|---|---|---|---|---|
| __export | __near | _huge | ada | PASCAL |
| __fastcall | __pascal | _import | AnsiString | False |
| __huge | __rtti | _interrupt | entry | FALSE |
| __import | __saveregs | _loadds | far | True |
| __int16 | __seg | _near | fortran | TRUE |
| __int32 | __ss | _pascal | huge | |
| __int64 | __stdcall | _saveregs | near | |
| __int8 | __thread | _seg | pascal | |
| __interrupt | __try | _ss | | |
| __loads | | _stdcall | | |

Avoid starting the name of a variable with two underscores; sometimes, the compiler would think that you are trying to use one of the words reserved for the compiler.

# Application 4 Vocabulary

## Lecture 1

| English | Russian |
|---|---|
| Hardware | Оборудование |
| Alignment | Выравнивание |
| Arrange | Упорядочить (отсортировать) |
| Binary number system | Двоичная система счисления |
| Bold | Полужирный |
| Borders and shading | Границы и заливка |
| Bullets or numbering | Маркированный или нумерованный список |
| Case | Корпус |
| Cell | Ячейка |
| Central processing unit | Центральный процессор |
| Copy | Копировать |
| Cut | Вырезать |
| Desktop | Рабочий стол |
| Dialog box | Диалоговое окно |
| Disk drive | Дисковод |
| Explorer | Проводник |
| Extension | Расширение |
| Hexadecimal number system | Шестнадцатиричная система счисления |
| Highlight | Выделять |
| Indent | Отступ |
| Inkjet printer | Струйный принтер |
| Input | Ввод |
| Insertion point | Курсор, точка ввода |
| Italic | Курсив |
| External peripherals | Внешние устройства |
| Line spacing | Межстрочный интервал |
| Motherboard | Материнская плата |
| Margin | Поле документа |
| Number system | Система счисления |
| Octal number system | Восьмиричная система счисления |
| Output | Вывод |
| Page setup | Параметры страницы |
| Paragraph | Абзац |
| Paste | Вставить |
| Power supply | Блок питания |
| Random access memory | Оперативная память |
| Recycle bin | Корзина |
| Run | Запуск |
| Shortcut | Ярлык |

| English | Russian |
|---|---|
| Software | Программное обеспечение |
| Start button | Кнопка «Пуск» |
| System bus | Системная шина |
| Tabs | Табуляция |
| Taskbar | Панель задач |
| Toolbar | Панель инструментов |
| Underline | Подчёркивать |

**Lecture 2**

| English | Russian |
|---|---|
| Application | Приложение |
| Assignment | Присваивание |
| Button | Кнопка |
| Code of the program | Текст программы |
| Compiler | Компилятор |
| Edit box | Текстовое поле |
| Float | Вещественное (число) |
| Integer | Целое (число) |
| Integrated Development Environment | Интегрированная среда разработки |
| Label | Надпись |
| Memo | Поле мемо (большое текстовое поле) |
| Modulo | Операция получения остатка от деления |
| Precedence | Предшествование (порядок выполнения) |
| Programming language | Язык программирования |
| Property | Свойство |
| Variable | Переменная |

**Lecture 3**

| English | Russian |
|---|---|
| Flowchart of the algorithm | Блок-схема алгоритма |
| Absolute value | Абсолютная величина (модуль) |
| Console | Консоль (монитор и клавиатура) |
| Constant | Константа (постоянная величина) |
| Cosine | Косинус |
| Exponential | Экспонента (e в степени x) |
| Header file | Заголовочный файл |
| Message box | Сообщение |
| Power | Степень |
| Sine | Синус |
| Square root | Квадратный корень |
| Tangent | Тангенс |

## Lecture 4

| **English** | **Russian** |
|---|---|
| Linear sequence | Линейная последовательность |
| Braces | Фигурные скобки |
| Choice | Выбор |
| Code block | Блок команд |
| Condition | Условие |
| Even | Чётное (число) |
| Multiple to smth. | Кратный чему-то |
| Nesting | Вложенный |
| Odd | Нечётный |
| Parenthesis | Круглые скобки |
| Relational operator | Операция отношения |
| Subroutine | Подпрограмма |
| Switch | Переключатель (оператор выбора вариантов) |

# Application 5 The list of functions and instructions

**Type conversion and input/output**

| Function or instruction | Explanation | Example |
|---|---|---|
| StrToInt | Converts string to int | x=StrToInt(Edit1->Text); |
| StrToFloat | Converts string to float | x=StrToFloat(Edit1->Text); |
| IntToStr | Converts int to string | Edit1->Text=IntToStr(x); |
| FloatToStr | Converts float to string | Edit1->Text=FloatToStr(x); |
| FormatFloat | Converts float to string | Edit1->Text=FormatFloat(“0.00”,x); |
| Memo->Lines->Add | Output in Memo | Memo1->Lines->Add("x="+ FloatToStr(x)+"  y="+ FormatFloat("0.00",y)); |
| ShowMessage | Output the message | ShowMessage(“Incorrect number”); |

**Math functions (math.h)**

| Function | Explanation | Example |
|---|---|---|
| abs | absolute value of integer x | Y= abs (x ) |
| fabs | absolute value of float x | Y= fabs (0.5*x ) |
| sqrt | square root of x | Y= sqrt(x) |
| pow | x raised to the power y | Y= pow (x, 4) |
| exp | exponential of x | Y= exp(x) |
| log | ln x | Y= log(x) |
| log10 | log to the base 10 of x | Y= log10 (x ) |
| sin | sine of angle in radians | Y= sin(x) |
| cos | cosine of angle in radians | Y= cos(3*x) |
| tan | tangent of angle in radians | Y= tan(M_PI+x) |
| acos | arc cosine of x | Y= acos(x) |
| asin | arc sine of x | Y= asin(x) |
| atan | arc tangent of x | Y= atan(x) |

**Console input/output**

| Function or instruction | Explanation | Example |
|---|---|---|
| cout<< | output | cout<<”Value of variable x is: ”<<x; |
| cin>> | input | cin>>x; |
| scanf | formatted input | scanf("%d", &a); |
| printf | formatted output | printf("x equals %d \n", x); |
| getch | pause | getch(); |
| clrscr | clears the screen | clrscr(); |

**Conditional statements**

| Instruction | Explanation | Example |
|---|---|---|
| if | Checks the condition and do action in dependence on the result of condition | if (x>0) y=sqrt(x);<br>else ShowMessage (“Error”); |
| switch | Selects on of the variants | switch(n){<br>  case 1: ShowMessage (“One”); break;<br>  case 2: ShowMessage (“Two”); break;<br>  case 3: ShowMessage (“Three”); break;<br>  default: ShowMessage (“Incorrect number”);<br>} |