

WHAT YOU ALREADY KNOW

- ❑ How to program computer:
 - Understand problem
 - Find out solution
 - Plan how to implement solution
 - Write program

Using Object centered design

- Identify objects and their types
- Identify operations needed to solve problem
- Arrange operations in sequence of steps(algorithm), which when applied to objects solve problem

HOW IS PROGRAM ORGANIZED

- ❑ Program code resides in “source” files (one or more).
- ❑ Source files has extension cc (example Hello.cc).
- ❑ You may use “header” files (with .h extension) to store declarations.
- ❑ Each program must have “main” function, which is place where execution of program starts.
- ❑ Order in which statements of program are executed is sequential!

Example:

```
main()
{
    int I;           // Declare variable
    cin >> I; // Read from input to variable
    cout << I;      // Write to output contents of variable
}
```

REPRESENTATION OF STANDARD TYPES

Binary code:

Internally, everything inside computer is represented in binary code.

Smallest unit is called bit, it can has value zero or one.

Group of eight bits is called Byte.

Number of bits that can fit into processor register is called word (can be bigger than Byte).

Example:

```
char a = 23;
```

Internally

a = 00010111 in binary code

REPRESENTATION OF STANDARD TYPES

Octal and Hexadecimal code:

Octal literals are starting with 0, hexadecimal with 0x

Example: 012 = 10 in decimal code

0x12 = 18 in decimal code

Real types

- float - usually 32 bit
- double - usually 64 bit
- long double - usually 96 or 128 bit

Real types can be represented as "fixed-point" or "floating-point".

Fixed-point:

$m.n$

Where integer part m or decimal part n can be omitted.

5.0 0.5 5. .5

REPRESENTATION OF STANDARD TYPES

- ❑ Floating-point:
- ❑ xEn or xen
- ❑ Where x is integer or fixed point real literal and n is integer exponent.
- ❑ Example: 12 billion
 - $0.12e11$
 - $1.2E10$
 - $12.0E9$
 - $12.e9$
 - $12E9$

Warning!

Internally compiler use type *double* when dealing with float. Some programmer never use *float*, instead always use *double* for real objects.

REPRESENTATION OF STANDARD TYPES

Character literals are written in C++ as single character symbol enclosed in apostrophes.

`'A', '@', '3', '+'`

Compiler stores this literals using their numeric codes in ASCII (in this case 65,43,51 and 124).

Some characters has special purpose in C++, to describe them there are **escape sequences**.

`'\"` will print apostrophe

`'\n'` will print newline character

ESCAPE SEQUENCES

Newline(NL or LF)	<code>\n</code>
Horizontal tab	<code>\t</code>
Vertical tab	<code>\v</code>
Backspace	<code>\b</code>
Carriage return	<code>\r</code>
Form feed	<code>\f</code>
Alert (BEL)	<code>\a</code>
Backslash	<code>\\</code>
Question mark	<code>\?</code>
Apostrophe	<code>\'</code>
Double quote	<code>\"</code>
With octal code	<code>\ooo</code>
With hexadecimal code	<code>\xhhh</code>

STRINGS

- Related to characters.
- It's sequence of characters enclosed in double quotes.

Example

```
"Hello world!"
```

```
"Enter \"id\" on one line \rand your name on another.\n"
```

Escape sequences can be used within string literal!

```
"\n\ta"
```

EXPLICIT TYPE CONVERSION

- ❑ type (expression)
- ❑ (type) expression

- ❑ Where type is valid C++ type and expression is any C++ expression.
- ❑ Example:
 - ❑ `int a = 0;`
 - ❑ `float b = 5.4;`
 - ❑ `a = (int) b;`
 - ❑ `a = int(b);`

EXAMPLE

- ❑ Problem:
Calculate wages for employee.

Known objects:

keyboard	cin
screen	cout
employee	empNumber
hours worked	hours
hourly rate	rate
wages	wages

Known operations:

read from input	>>
write to output	<<
compute wage	hour*rate

EXAMPLE - CONTINUE

□ Sample solution:

- Prompt user for employee number
- Prompt user for number of hours employee worked
- Prompt user for hour rate of employee
- Compute wages earned by employee
- Print result to screen

EXAMPLE - PROGRAM

Program:

```
#include <iostream.h>
int main()
{
    cout << "Enter employee number: ";
    int empNumber;           //employee number
    cin >> empNumber;
    cout << "Enter the hours worked and the pay rate: ";
    double hours, rate;
    cin >> hours >> rate;
    double wages = hours*rate;
    cout << "\nEmployee number: " << empNumber
        << "\nHours worked: " << hours
        << "\nHourly rate: " << rate
        << "\nTotal wages: $" << wages << endl;
    return 0;
}
```