1.2 C++ DATA TYPES

Data type is a very important concept in C/C++. Any date (constant or variable) is of a named type. The data type specifies what values that date can have and what operations can be done with it.

In C/C++ data types are:

1. Simple type:
   - The *int* type
   - The *float* and *double* types
   - The *char* type
   - The *pointer* type
   - The *bool* type
   - The *void* type
2. Derived types
   - The *painting* type
   - *Structure* type/class
   - The *enumeration* type

The int type

Allows memorization of **integer values** - positive or negative.

A date of type int occupies (as a rule) **4 bytes**; thus, it can store integer values from $[-2^{31},2^{31}-1]$, i.e. [−2,147,483,648,2,147,483,647].

e.g.: *int n=100;*

The float and double types

Memorize **real numbers**.

The decimal separator is the **dot**.

They are also called **floating point** types.

Real data can be given in **fixed** form or in **scientific** (exponential) form.

The float type is represented by **4 bytes**. The double type is represented by **8 bytes**.

e.g.: *float p = 3.14, r = 2.5;*

*double A = p * r * r;*

Real data can be written in scientific (exponential) form:

*double x = 1.24E+07; // means 1.24 * 10^7*

The char type

Is used for **ASCII characters**.

Stores **a single character**.

Is represented on **1 byte**.

Characters are delimited by an **apostrophe** (').

e.g.: *char c='A';*

The pointer type

A pointer data stores a **memory address** – for example the address of a variable.

The bool type

Certain operations performed on data result in truth values: **true** or **false**.

Type bool contains two values: **true** and **false**.

It is represented (as a rule) on **1 byte**.

Their **numerical values** are 1 and 0.

Are used in **conditional** and **repetitive** statements.

e.g.: *bool pp = false;*

The void type

The word void means "nothing" or "worthless".

Void type data have **no values** and **cannot be operated on**.

We use it for **functions** and **pointers**.

Type modifiers

They allow changing the way the internal representation of a date is made. They are:

- signed
- unsigned
- shorts
- long

They can be applied to types

- *int*
- *double*
- *char*

Pay attention to!

- In problems, if the integer data does not exceed (roughly) 2,000,000,000 we use the int type. For data that exceeds this value we will use the long long type.
- In C++, a char data does not store the character, but a number corresponding to the character. More details here.
- We cannot declare variables of type void.

Data type	Representation	Meaning

signed int	4 signed bytes	Same as *int*. Integer values from [-2^{31}, $2^{31}-1$], i.e. [−2147483648, 2147483647].
unsigned int	4 unsigned bytes	Natural values from [0, $2^{32}-1$], i.e. [0.4294967295].
long	4 signed bytes	Same as *int*. Equivalent to *long int*.
unsigned long	4 unsigned bytes	Same as unsigned int. Equivalent to *unsigned long int*.
short	2 signed bytes	Small integer values from [-2^{15}, $2^{15}-1$] , i.e. [−32768, 32767]. Equivalent to *short int*.
unsigned short	2 unsigned bytes	Small natural values from [0, $2^{16}-1$] , i.e. [0, 65535]. Equivalent to *unsigned short int*.
long long	8 signed bytes	Very large integer values from [-2^{63},$2^{63}-1$]. Equivalent to *long long int*.
unsigned long long	8 unsigned bytes	Very large natural values of [0, $2^{64}-1$]. Equivalent to *unsigned long long int.*
signed char	1 signed byte	Characters. Numerical values are from [-2^7, 2^7-1], i.e. [−128, 127].
unsigned char	1 unsigned byte	Characters. Numerical values are from [0, 2^8-1], i.e. [0, 255].
long double	10, 12, 16	Store large real numbers. The representation depends on the compiler, but it must take up at least as much space as *double*.