**DATA TYPES**

Data types are means to identify the type of data and associated operations of handling it.

C++ data types are of two types – (i) Fundamental datatypes (ii) Derived datatypes.

Fundamental datatypes

Fundamental datatypes are those that are not composed of other datatypes.

**(i)** **int datatype (for integers):** Integers are whole numbers such as 5,39, -19,0 etc. They have no fractional parts. Integers are represented on C++ by int datatype.

To declare a variable the syntax is

<datatype> <variable name>; where <datatype> is a C++ datatype and <variable name> is an identifier. For example, int number;

Program to input an integer display its first three multiples.

#include<iostream>

using namespace std;

main()

{

int number,multiple1,multiple2,multiple3;

cout<<"Enter a number:";

cin>>number;

multiple1=number\*1;

multiple2=number\*2;

multiple3=number\*3;

cout<<"Entered number is:"<<number<<"\n\n";

cout<<"First three multiples of a given number are:";

cout<<multiple1<<","<<multiple2<<","<<multiple3;

return 0;

}

**(ii) char Data type (for characters):** Character can store any number of the C++ implementations basic character set. If a from this set is stored in a character variable, its value is equivalent to the integer code of that character.

**(iii) float Data Type (floating – point numbers)**: A number having fractional part is a floating – point number. For example, 3.14159 is a floating-point number.

**(iv) double Data Type (for double precision floating numbers):** The datatype double is also used for handling floating – point numbers. It (double data type) occupies twice as much memory as type float, and stores floating -point numbers with much larger range and precision (significant numbers after decimal point). It stands for double precision floating point.

**(v) void Data Type (for empty set of values and non-returning functions):** The void type specifies an empty set of values. It is used as the retorn type for functions that do not return a value.

DATA TYPE MODIFIERS

Except type void, the basic data types may have various modifiers preceding them.

The list of modifiers us given below:

signed

unsigned

long

short

**A. Integer Type Modifiers**

By using different number of bytes to store values, C++ offers three types of integers: short, int and long that can represent up to three different integer sizes.

Short is short name of short int and long is short name of long int.

The prefix unsigned makes the integer type not to hold negative values

Table: Integer types

|  |  |  |
| --- | --- | --- |
| Type | Size (in bytes) | Minimal Range |
| Short  Unsigned short  Signed short | 2  2  2 | -32768 to 32767  0 to 65535  Same as short |
| int  Unsigned int  Signed int | 2  2  2 | -32768 to 32767  0 to 65535  Same as int |
| long  Unsigned long  Signed long | 4  4  4 | -2,147,483,648 to 2,147,483,647  0 to 4,294,967,295  Same as long |

Character type modifiers: Char type can also be signed or unsigned.

Table: Char Types

|  |  |  |
| --- | --- | --- |
| Type | Size (in bytes) | Minimal Range |
| char | 1 | -128 to 127 |
| Unsigned char | 1 | 0 to 255 |
| Signed char | 1 | Same as char |

Floating point type modifiers:

Table: Floating point types

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Size (in bytes) | Minimal Range | Digits of precision |
| Float  Double  Long double | 4  8  10 | 3.4×10-38 to 3.4×1038-1  1.7×10-308 to 1.7×10308-1  3.4×10-4932 to 1.1×104932-1 | 7  15  19 |

**Derived Data Types**

From the fundamental types other types can be derived by using the declaration operators.

1. Arrays: Arrays refer to a named list of a finite number n of similar data elements.

Array can be one dimensional, two dimensional or multi-dimensional.

float a[3];

int b[2][4];

2. Functions: A function is a named part of a program that can be invoked from other parts of the program.

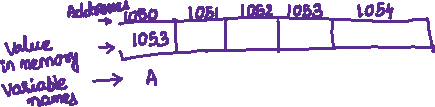
Example: float cube(float a)

{

Return a\*a\*a;

}

3. Pointers: A pointer is a variable that holds a memory address. This address is usually the location of another variable in memory



4. References: A reference is an alternative name for an object. A reference variable provides an alias for a previously defined variable.

5.Constants: The keyword const can be added to the declaration of an object to make that object a constant rather than a variable.

Const type name=value;

**User defined derived data types:**

There are some derived data types that are defined by the user.

1. Class: A class represents a group of similar objects. For instance

Class department

{

Char name[20];

Int num\_emp;

Char h\_o\_d[20];

Public:

Add();

Delete();

Modify();

Print();

}; department sales, purchase, import, accounts;

The above declaration declares a class department and the last line declares objects sales, purchase, import, accounts of class type department.

2. Structure: A structure is a collection of variables referenced under one name.

The following example creates a structure for student record:

Struct sturec

{

int rollno;

char name[20];

int class;

float marks;

char grade;

};

sturec newstu;

3. Union: A union is a memory location that is shared by two or more different variables.

Union share

{

int i;

char ch;

};

Union share cnvt;

In the union cnvt, both integer i and character ch share the same memory location.

4. Enumeration: An alternative method for naming integer constants is often more convenient than const.

enum{start,pause,go}; defines three integer constants, called enumerators and assigns values to them. Enumerator values are by default assigned increasing from 0, the above declaration is equivalent to writing

const int start=0;

const int pause=1;

const int go=2;