Datatypes

Data used in c program is classified into different types based on its properties. In c programming langauge, datatype can be defined as a set of values with similar characteristics. All the values in a datatype have the sameproperties.

Datatypes in c programming language are used to specify what kind of value can be stored in a variable. The memory size and type of value of a variable are determined by varible datatype. In a c program, each variable or constant or array must have a datatype and this datatype specifies how much memory is to be allocated and what type of values are to be stored in that variable or constant or array. The formal definition of datatype is as follows...

Definition

Datatype is a set of value with predefined characteristics. Datatypes are used to declare variable, constants, arrays, pointers and functions.

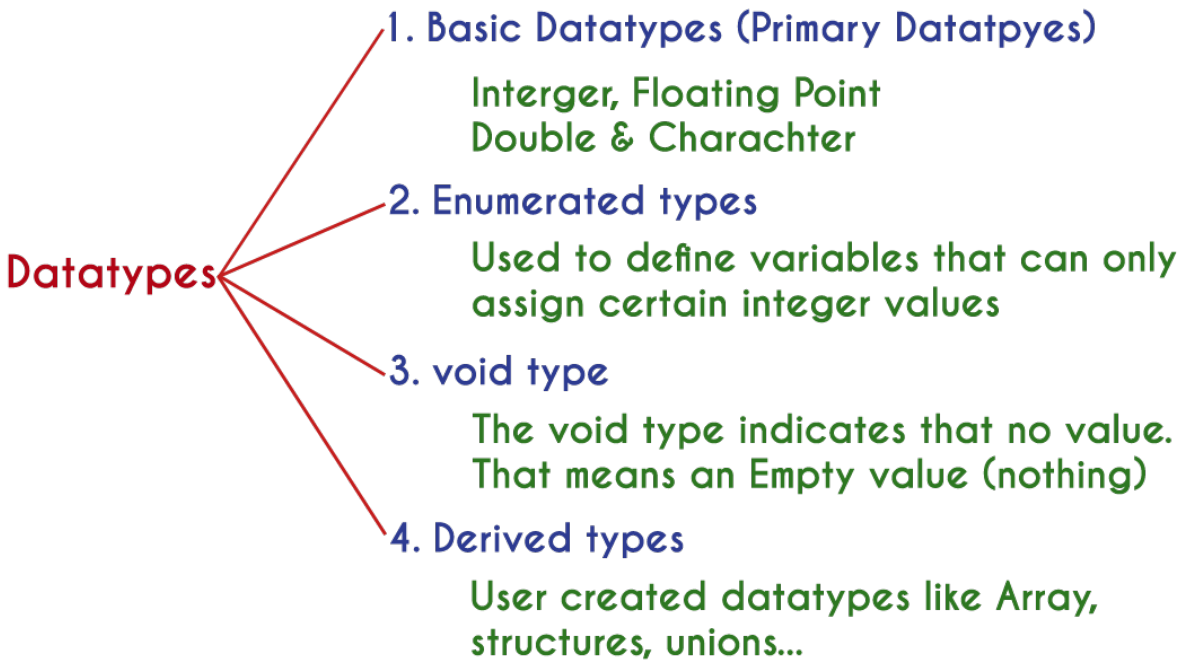
In c programming language, datatypes are classified as follows...

Primary Datatypes (Basic Datatypes OR Predefined Datatypes)

Derived Datatypes (Secondary Datatypes OR Userdefined Datatypes)

Enumeration Datatypes

Void Datatype



C Data Types

Basic Data Type int, char, float, double

Derived Data Type array, pointer, structure, union

Enumeration Data Type enum

Void Data Type void

Integer Types

The following table provides the details of standard integer types with their storage sizes and value ranges –

Type	Storage size(16 bit)	Storage Size(32bit)	Value range 16 bit	Value range 32/64 bit	Format Specifier
Char	1 byte	1	-128 to 127 or 0 to 255 (-2^7 to $+2^7$ -1) or (0 to 2^8 -1)	-128 to 127 or 0 to 255 (-2^7 to $+2^7$ -1) or (0 to 2^8 -1)	%c
unsigned char	1 byte	1	0 to 255 or (0 to 2^8 -1)	0 to 255 or (0 to 2^8 -1)	%c
signed char	1 byte	1	-128 to 127 or (-2^7 to $+2^7$ -1)	-128 to 127 or (-2^7 to $+2^7$ -1)	%c
Int	2 bytes	4	-32,768 to 32,767 or (-2^{15} to $+2^{15}$ -1)	-2,147,483,648 to 2,147,483,647 or ($-2^{31}$to $+2^{31}$ -1)	%d ,%i,%o,%0x ,%0X
unsigned int	2 bytes	4	0 to 65,535 or (0 to $+2^{16}$ -1)	0 to 4,294,967,295 Or 0 to $+2^{32}$ -1	%u
Short	2 bytes	2	-32,768 to 32,767 or (-2^{15} to $+2^{15}$ -1)	-32,768 to 32,767 or (-2^{15} to $+2^{15}$ -1)	%hd
unsigned short	2 bytes	2	0 to 65,535 or (0 to $+2^{16}$ -1)	0 to 65,535 or (0 to $+2^{16}$ -1)	%hu
Long	4 bytes	4	-2,147,483,648 to 2,147,483,647 or ($-2^{31}$to $+2^{31}$ -1)	-2,147,483,648 to 2,147,483,647 or ($-2^{31}$to $+2^{31}$ -1)	%ld
unsigned long	4 bytes	4	0 to 4,294,967,295 Or 0 to $+2^{32}$ -1	0 to 4,294,967,295 Or 0 to $+2^{32}$ -1	%lu
long long int		8		-(2^{63}) to (2^{63})-1	%lld
unsigned long long int		8		0 to 18,446,744,073,709,551,615 Or Or 0 to $+2^{64}$ -1	%llu

<climits> (limits.h) in C/C++
The maximum and minimum size of integral values are quite useful or in simple terms limits of any integral type plays quite a role in programming. Instead of remembering these values differnet macros can be used.

<climits>(limits.h) defines sizes of integral types.

This header defines constants with the limits of fundamental integral types for the specific system and compiler implementation used.

The limits for fundamental floating-point types are defined in <cfloat> (<float.h>).
The limits for width-specific integral types and other typedef types are defined in <cstdint> (<stdint.h>).

Different macro constants are :

1. CHAR_MIN :
Minimum value for an object of type char
Value of CHAR_MIN is either -127 (-2^7+1) or less* or 0

2. CHAR_MAX :
Maximum value for an object of type char
Value of CHAR_MAX is either 127 (2^7-1) or 255 (2^8-1) or greater*

3. SHRT_MIN :
Minimum value for an object of type short int
Value of SHRT_MIN is -32767 ($-2^{15}+1$) or less*

4. SHRT_MAX :
Maximum value for an object of type short int
Value of SHRT_MAX is 32767 ($2^{15}-1$) or greater*

5. USHRT_MAX :
Maximum value for an object of type unsigned short int
Value of USHRT_MAX is 65535 ($2^{16}-1$) or greater*

6. INT_MIN :
Minimum value for an object of type int
Value of INT_MIN is -32767 ($-2^{15}+1$) or less*

7. INT_MAX :
Maximum value for an object of type int
Value of INT_MAX is 32767 ($2^{15}-1$) or greater*

8. UINT_MAX :
Maximum value for an object of type unsigned int

Value of UINT_MAX is 65535 (2^{16}-1) or greater*

9. LONG_MIN :
Minimum value for an object of type long int
Value of LONG_MIN is -2147483647 (-2^{31}+1) or less*

10. LONG_MAX :
Maximum value for an object of type long int
Value of LONG_MAX is 2147483647 (2^{31}-1) or greater*

11. ULONG_MAX :
Maximum value for an object of type unsigned long int
Value of ULONG_MAX is 4294967295 (2^{32}-1) or greater*

12. LLONG_MIN :
Minimum value for an object of type long longint
Value of LLONG_MIN is -9223372036854775807 (-2^{63}+1) or less*

13. LLONG_MAX :
Maximum value for an object of type long longint
Value of LLONG_MAX is 9223372036854775807 (2^{63}-1) or greater*

14. ULLONG_MAX :
Maximum value for an object of type unsigned long longint
Value of ULLONG_MAX is 18446744073709551615 (2^{64}-1) or greater*

NOTE** the actual value depends on the particular system and library implementation, but shall reflect the limits of these types in the target platform.
Your Machine's values might depending upon whether it is 32 bit machine or 64 bit machine.
Compatibility :
LLONG_MIN, LLONG_MAX and ULLONG_MAX are defined for libraries complying with the C standard of 1999 or later (which only includes the C++ standard since 2011: C++11).
Two Applications of these MACROS are Checking for integer overflow and Computing minimum or maximum in an array of very large or very small elements.
Below Program will display the respective values for your machine:

```
// C++ program to demonstrate working of
```

```
// constants in climits.
#include <limits.h>
#include <stdio.h>

int main()
{
   Printf("CHAR_MIN :%d", CHAR_MIN);
   Printf("CHAR_MAX :%d",CHAR_MAX);
   Printf("SHRT_MIN :%hd",SHRT_MIN);
   Printf("SHRT_MAX :%hd",SHRT_MAX);
   Printf("USHRT_MAX :%hu",USHRT_MAX);
   Printf("INT_MIN :%d ",INT_MIN);
   Printf("INT_MAX :%d ",INT_MAX );
   Printf("UINT_MAX :%u",UINT_MAX);
   Printf("LONG_MIN :%ld ",LONG_MIN);
   Printf("LONG_MAX :%ld ",LONG_MAX);
   Printf("ULONG_MAX :%lu ",ULONG_MAX);
   Printf("LLONG_MIN :%lld ",LLONG_MIN);
   Printf("LLONG_MAX :%lld ",LLONG_MAX);
   Printf("ULLONG_MAX :%llu",ULLONG_MAX);
   return0;
}
```

Output (Machine dependent):

```
CHAR_MIN : -128
CHAR_MAX : 127
SHRT_MIN : -32768
SHRT_MAX : 32767
USHRT_MAX : 65535
INT_MIN : -2147483648
INT_MAX : 2147483647
UINT_MAX : 4294967295
LONG_MIN : -9223372036854775808
LONG_MAX : 9223372036854775807
ULONG_MAX : 18446744073709551615
LLONG_MIN : -9223372036854775808
LLONG_MAX : 9223372036854775807
ULLONG_MAX : 18446744073709551615
```

If we do not specify either signed or unsigned, most compiler will assume the type to be signed. so signed int x; can be written as int x; short and long can be used alone as type specifiers.

- short=short int long=long int Short int x; can be wriiten as short x;

signed and unsigned can also be used alone as type specifiers

.• signed int=signed unsigned int=unsigned unsigned int x; can be wriiten as unsigned x;

Floating point type

Floating types are used to store real numbers.

Size and range of floating type on 16-bit machine

Type	Size(bytes)	Range
Float	4	3.4E-38 to 3.4E+38
Double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

The following table provide the details of standard floating-point types with storage sizes and value ranges and their precision −

Type	Storage Size 16 bit	Storage Size 32/64 bit	Value Range(16 bit)	Value range 32/64 bit	Format specifier	Precision
float	4 byte	4byte	3.4 E -38 to 3.4E+38	1.2E-38 to 3.4E+38	%f ,%e,%g	6 decimal places
double	8 byte	8byte	1.7 E-308 to 1.7E+308	2.3E-308 to 1.7E+308	%lf ,%le,%lg	15 decimal places
long double	10 byte	16 byte	3.4E-4932 to 1.1E+4932	3.4E-4932 to 1.1E+4932	%Lf ,%Le , %Lg	19 decimal places

The header file float.h defines macros that allow you to use these values and other details about the binary representation of real numbers in your programs. The following example prints the storage space taken by a float type and its range values −

```c
#include<stdio.h>
#include<float.h>

int main(){
printf("Storage size for float : %d \n",sizeof(float));
printf("Minimum float positive value: %E\n", FLT_MIN );
printf("Maximum float positive value: %E\n", FLT_MAX );
printf("Precision value: %d\n", FLT_DIG );

printf("Minimum float positive value: %lE\n", DBL_MIN );
printf("Maximum float positive value: %lE\n", DBL_MAX );
printf("Minimum float positive value: %LE\n", LDBL_MIN );
printf("Maximum float positive value: %LE\n", LDBL_MAX );

return0;
}
```

When you compile and execute the above program, it produces the following result on Linux −

```
Storage size for float : 4
Minimum float positive value: 1.175494E-38
Maximum float positive value: 3.402823E+38
Precision value: 6
```