||Datatype||

* Datatypes are 4 types,
  1. Basic :- char, int, float, double
  2. Empty :- void
  3. Derived :- array, pointer
  4. User Defined :- struct, union, typedef, enum

* Modifiers present in c++ are,
  Signed, unsigned, short, long, long long

* Qualifiers present in c++ are,
  Const, volatile

* Storage classes,
  auto, static, extern, register

→ Datatype concept is required to allocate memory.
→ Modifier " " " " decide the size of memory
→ qualifier " " " " " behaviour "
→ storage class " " " " " scope of memory

Char datatype :-

→ The size of char datatype is 1 byte.
→ char const is represented with in single code (' ').
→ char const is an interal const because for every char there is an integer const & that is known as ASCII value
→ char type support Signed & unsigned modifier & doesnot support short, long & long long modifier
→ The size of Signed char & unsigned char is 1 byte.
→ The default modifier for char is signed.
→ Signed char supports both +ve & -ve value, but unsigned char supports only +ve value.

→ -ve char stored in the memory in the form of 2's compliment.
→ The range of signed char in form -128 to 127.
→ The range of unsigned char is form 0 to 255.

To print char to its ASCII value

```
char x = 'A';
Printf ("%d\n", x); // 65
```

To check signed char range

```
signed char a=0;
while (1)
{
Printf ("%d\n", a);
a++;
usleep (30000);
}
```

To check unsigned char range

```
unsigned char a=0;
while (1)
{
Printf ("%d\n", a);
a++;
usleep (30000);
}
```

int datatype :-

→ The size of int datatype is 4 byte.
→ Integer datatype supports all type of constants except string constant
→ It supports all modifier.
→ Signed & unsigned modifier doesnot come together.
→ short, long & long long " " " "
→ long signed is the default modifier for int datatype.

→ Int variable stored in memory in the form of endianess.
→ Endianess is the process of byte ordering.
→ Endianess is of two types:
  @ Little endianess (dcba format)
    Ex :- intel processor
  @ Big endianess (abcd format)
    Ex :- motrolla, power pc processor
→ when a +ve integer stored in memory it is stored a/c to endianess
→ when a -ve data stored in memory first it is encrypted a/c to 2's complement rule the stored a/c to endianess.

Float datatype :-

→ First float value converted into binary form.
→ The binary is normalised to form a decimal mantisa & exponent $(1.m * e^n)$
→ The exponent value will be added with 127 if float type.
→ The added value is converted into binary.
→ Mantisa bits are be stored in mantisa place & set sign bit also.

Ex :- no = 4.5

Float no $\xrightarrow{\text{binary}}$ 100.10 $\xrightarrow{\text{normalize}}$ $1.0010 * e^2$.

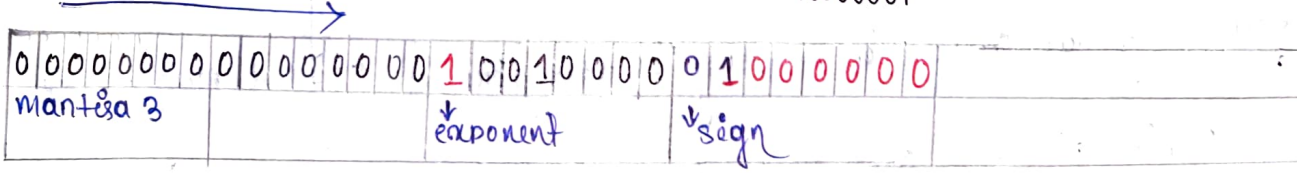
↓

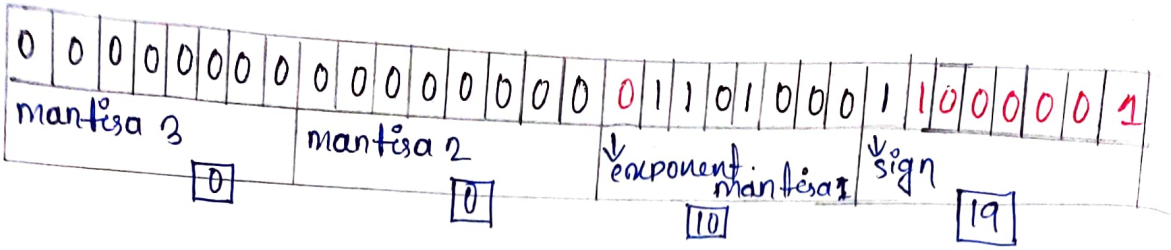
127 + 2

↓

129

10000001





Program 1 :-

```
Float x=4.5;
unsigned char *p = &x;
Printf ("%d\n", *(p+0)); //0
Printf ("%d\n", *(p+1)); //0
Printf ("%d\n", *(p+2)); //144
Printf ("%d\n", *(p+3)); //64
```

→ Floating type are used to store real numbers.
→ Float keyward is used to refer float data type.
→ The size of float datatype is 4 byte.
→ Format specifiers of double type is %f.
→ Float supports every type of constant except string constant.
→ Float doesnot support any modifier.
→ Float variable stored in memory in the form of exponent & mantisa order.
→ To work with 4 byte according to endianess but to work with 4 byte a/c to exponent & mantisa rule for float type pointer is required.
→ negative float stored in memory in the form of signed magnitude concept.

1 bit sign → first byte last bit is sign bit.
8 bit exponent → first 7 bit & second byte last bit
23 bit mantisa → rest bits are

```
#include "stdio.h"
main()
{
Float no1=45, no2 = -67.78;
Printf ("%f %f", no1, no2);
```

Double datatype :-

→ Double types are used to store real numbers.
→ 'double' keyward is used to refer double data type.
→ The size of double datatype is 8 byte.
→ Format Specifiers of double type is "%.lf".
→ Double datatype doesnot support string constant.
→ Double datatype doesnot support support short, signed & unsigned modifiers.
→ The size of long double is 12 byte.
→ Long long modifiers doesnot affect the double datatype.

Note

Float (4 byte / 32 bit)
1) Signed bit 1
2) Exponent bit 8
3) Mantisa bit 23

number will be added with ex bit = 2^ex bit-1 = 2^8-1 = 128-1=127

double (8 byte / 64 bit)
1) Signed bit 1
2) Exponent bit 11
3) Mantisa bit 52

number will be added with ex bit = 2^exbit-1 = 2^11-1 = 1024 - 1 = 1023

Long double (12 byte / 96 bit)
1) Signed bit 1
2) Exponent bit 15.
3) Mantisa bit 80.

number will be added with ex bit = 2^exbit-1 = 2^15 - 1 = 16384 - 1 = 16383

Enum datatype :-

enum datatype create a sequence set of integer constant.

Ex :- 1

```
main()
{
enum {a, b, c, d, e};
Printf {"%d %d %d %d %d", a, b, c, d, e};
}
```

ex :- 2

```
main()
{
enum {a = 100, b, c, d, e};
Printf {"%d %d %d %d %d", a, b, c, d, e};
}
```

Ex :- 3

```
main()
{
enum {a = 100.5, b, c, d, e};
Printf ("%d %d %d %d %d", a, b, c, d, e};
}
```

// error : enumeration value for 'a' is not an integer constant

Void datatype :-

It is an empty data type associated with Func & pointer.

use :-

→ Generic pointer
→ No return type of a Func^n
→ No argument of a Func^n

Ex :- 1

```
int cte(void)
{
Printf ("I am a Func^ with no argument \n");
}
```

```
void bbsrc (int x)
{
Printf ("i am a func^ having no return type\n");
        Printf ("%d\n", x);
}

main ()
{
int m = 10;
void *p ; //generic pointer
ctc ();
bbsrc (m);
}
```