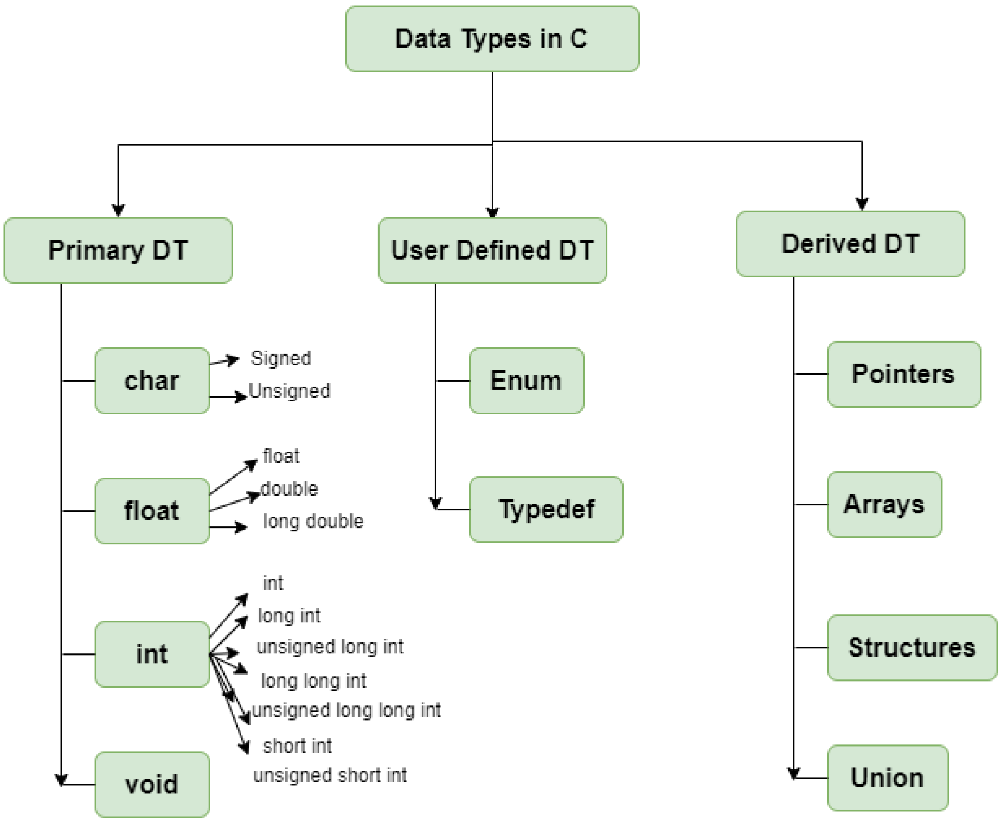
DATA TYPES

Each variable in C has an associated data type.
It specifies the type of data that the variable can store like integer, character, floating, double, etc.
Each data type requires different amounts of memory and has some specific operations which can be performed over it.
The data type is a collection of data with values having fixed values, meaning as well as its characteristics.

The data types in C can be classified as follows:



Primitive Data Types:

Primitive data types are the most basic data types that are used for representing simple values such as integers, float, characters, etc.

User Defined Data Types

The user-defined data types are defined by the user himself.

Derived Data Types

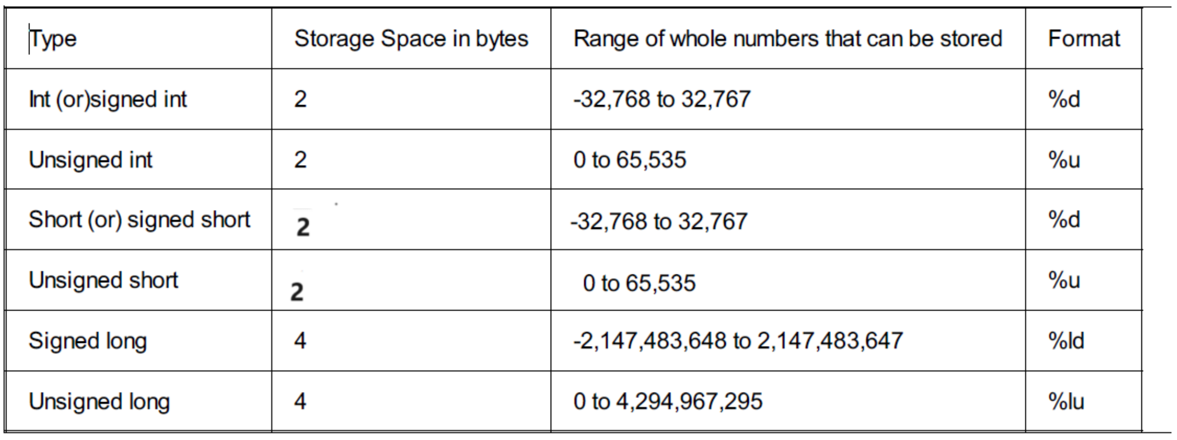
The data types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types.

Primitive Data Types:

Integer Data Type

The integer datatype in C is used to store the integer numbers(any number including positive, negative and zero without decimal part).

ranges for different integer datatypes



Type	Storage Space in bytes	Range of whole numbers that can be stored	Format
Int (or)signed int	2	-32,768 to 32,767	%d
Unsigned int	2	0 to 65,535	%u
Short (or) signed short	2	-32,768 to 32,767	%d
Unsigned short	2	0 to 65,535	%u
Signed long	4	-2,147,483,648 to 2,147,483,647	%ld
Unsigned long	4	0 to 4,294,967,295	%lu

Syntax of Integer

We use **int** **keyword** to declare the integer variable:

int ***var_name;***

Example

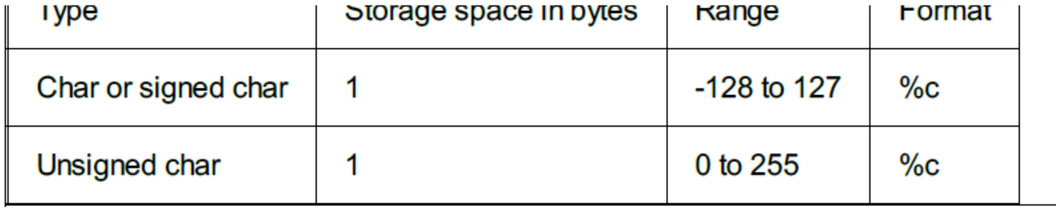
```
// C program to print Integer data types.

#include <stdio.h>

int main()
{
        // Integer value with positive data.
        int a = 9;

        // integer value with negative data.
        int b = -9;

        // U or u is Used for Unsigned int in C.
        int c = 89U;

        // L or l is used for long int in C.
        long int d = 99998L;

        printf("Integer value with positive data: %d\n", a);
        printf("Integer value with negative data: %d\n", b);
        printf("Integer value with an unsigned int data: %u\n",
                c);
        printf("Integer value with an long int data: %ld", d);

        return 0;
}
```

Output

```
Integer value with positive data: 9
Integer value with negative data: -9
Integer value with an unsigned int data: 89
Integer value with an long int data: 99998
```

Character Data Type
Character data type allows its variable to store only a single character. The size of the character is 1 byte. It is the most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.

This datatype is used to store ASCII characters. Examples are a , b , z , 8 , $.
char is supposed to
store characters only not numbers, so why should it holds range?
In the memory characters are stored in their ASCII codes. For example, the character 'B' has the ASCII code
66. In the memory 'B' will not be stored as 'B' but as 66.
We have unsigned char and signed char.We don't have negative characters, then why should we have such
datatypes?
We used signed and unsigned char to ensure portability of programs that store non character data as *char*.



Type	Storage space in bytes	Range	Format
Char or signed char	1	-128 to 127	%c
Unsigned char	1	0 to 255	%c

Syntax of char
The **char keyword** is used to declare the variable of character type:
char *var_name;*

Example of char

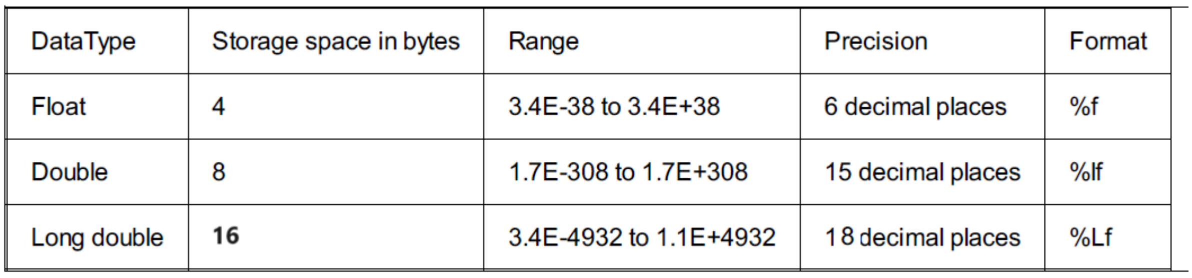
```
// C program to print Integer data types.
#include <stdio.h>

int main()
{
	char a = 'a';
	char c;

	printf("Value of a: %c\n", a);

	a++;
	printf("Value of a after increment is: %c\n", a);

	// c is assigned ASCII values
	// which corresponds to the
	// character 'c'
	// a-->97 b-->98 c-->99
	// here c will be printed
	c = 99;

	printf("Value of c: %c", c);

	return 0;
}
```

Output:

```
Value of a: a
Value of a after increment is: b
Value of c: c
```

Float Data Type

In C programming float data type is used to store floating-point values. Float in C is used to store decimal and exponential values. It is used to store decimal numbers (numbers with floating point values) with single precision.



DataType	Storage space in bytes	Range	Precision	Format
Float	4	3.4E-38 to 3.4E+38	6 decimal places	%f
Double	8	1.7E-308 to 1.7E+308	15 decimal places	%lf
Long double	**16**	3.4E-4932 to 1.1E+4932	18 decimal places	%Lf

Syntax of float
The **float keyword** is used to declare the variable as a floating point:
float *var_name;*

Example of Float

```
// C Program to demonstrate use
// of Floating types
#include <stdio.h>

int main()
{
        float a = 9.0f;
        float b = 2.5f;

        // 2x10^-4
        float c = 2E-4f;
        printf("%f\n", a);
        printf("%f\n", b);
        printf("%f", c);

        return 0;
}
```

Output

```
9.000000
2.500000
0.000200
```

Syntax of Double
The variable can be declared as double precision floating point using the **double keyword:**
double *var_name;*

Example of Double

```
// C Program to demonstrate
// use of double data type
#include <stdio.h>

int main()
{
	double a = 123123123.00;
	double b = 12.293123;
	double c = 2312312312.123123;

	printf("%lf\n", a);

	printf("%lf\n", b);

	printf("%lf", c);

	return 0;
}
```

Output

```
123123123.000000
12.293123
2312312312.123123
```

Void Data Type

The void data type in C is used to specify that no value is present. It does not provide a result value to its caller. It has no values and no operations. It is used to represent nothing. Void is used in multiple ways as function return type, function arguments as void, and pointers to void.

Void data type variables doesn't have a value of any type. Void means no value. Theoretically the void type should use no **manipulable memory** on stack when defined. That is to say, a
variable declared with a "void" type shouldn't be capable of storing any data, and therefore has a ***size of zero***.
Basically void data type are used in some special places and some of them are following :
To specify return type of a function(when function returns no value).
To specify the parameters of a function(when the function accepts no argument from the caller)
To create generic pointers.

Syntax:

```
// function return type void
void exit(int check);
// Function without any parameter can accept void.
int print(void);
// memory allocation function which
// returns a pointer to void.
void *malloc (size_t size);
```

Example of Void

```
// C program to demonstrate
// use of void pointers
#include <stdio.h>

int main()
{
	int val = 30;
	void* ptr = &val;
	printf("%d", *(int*)ptr);
	return 0;
}
```

Output

```
30
```

Size of Data Types in C

The size of the data types in C is dependent on the size of the architecture, so we cannot define the universal size of the data types. For that, the C language provides the sizeof() operator to check the size of the data types.

Example

```
// C Program to print size of
// different data type in C
#include <stdio.h>

int main()
{
	int size_of_int = sizeof(int);
	int size_of_char = sizeof(char);
	int size_of_float = sizeof(float);
	int size_of_double = sizeof(double);

	printf("The size of int data type : %d\n", size_of_int);
	printf("The size of char data type : %d\n",
		size_of_char);
	printf("The size of float data type : %d\n",
		size_of_float);
	printf("The size of double data type : %d",
		size_of_double);

	return 0;
}
```
Output

```
The size of int data type : 4
The size of char data type : 1
The size of float data type : 4
The size of double data type : 8
```

Data Type	Size (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d

Data Type	Size (bytes)	Range	Format Specifier
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%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	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	1.2E-38 to 3.4E+38	%f
double	8	1.7E-308 to 1.7E+308	%lf
long double	16	3.4E-4932 to 1.1E+4932	%Lf