Table 2.1 *C Character Set*

Letters	*Digits*
Uppercase A.....Z	All decimal digits 0 9
Lowercase a.....z	

Special Characters

, comma	& ampersand
. period	^ caret
; semicolon	* asterisk
: colon	– minus sign
? question mark	+ plus sign
‘ apostrophe	< opening angle bracket (or less than sign)
“ quotation mark	> closing angle bracket (or greater than sign)
! exclamation mark	(left parenthesis
| vertical bar	) right parenthesis
/ slash	[left bracket
\ backslash	] right bracket
~ tilde	{ left brace
_ under score	} right brace
$ dollar sign	# number sign
% percent sign	

White Spaces

Blank space
Horizontal tab
Carriage return
New line
Form feed

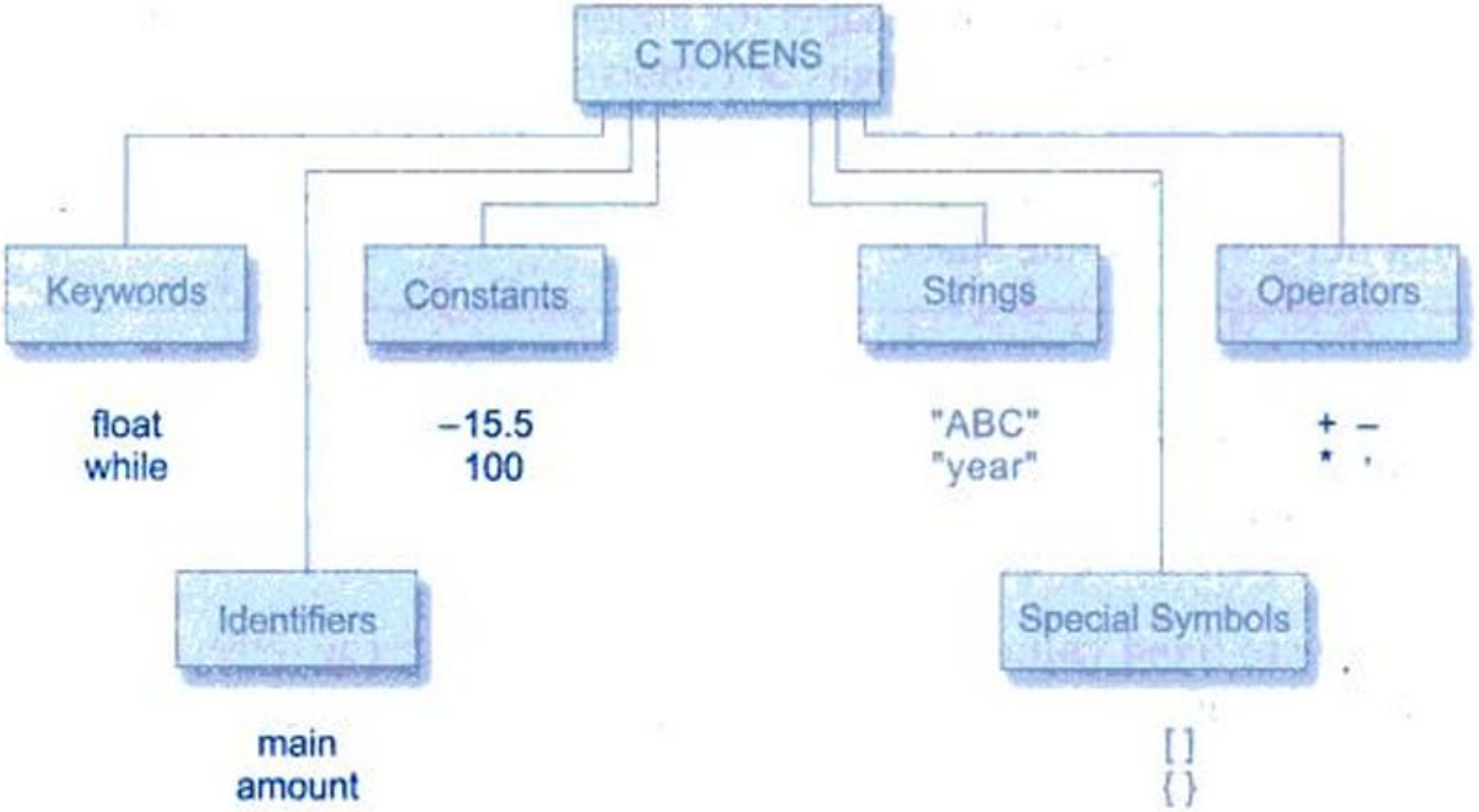


Fig. 2.1 *C tokens and examples*

Table 2.3 *C Keywords*

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

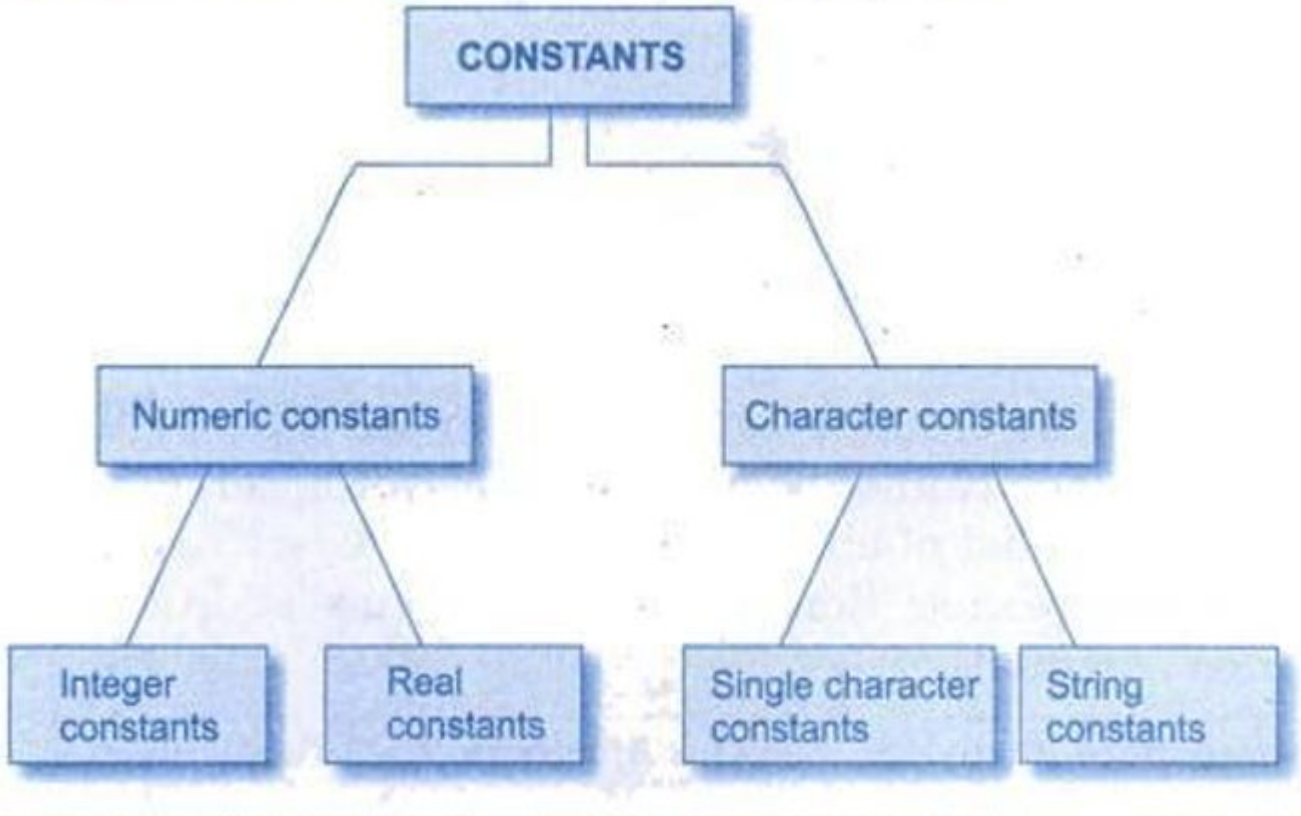
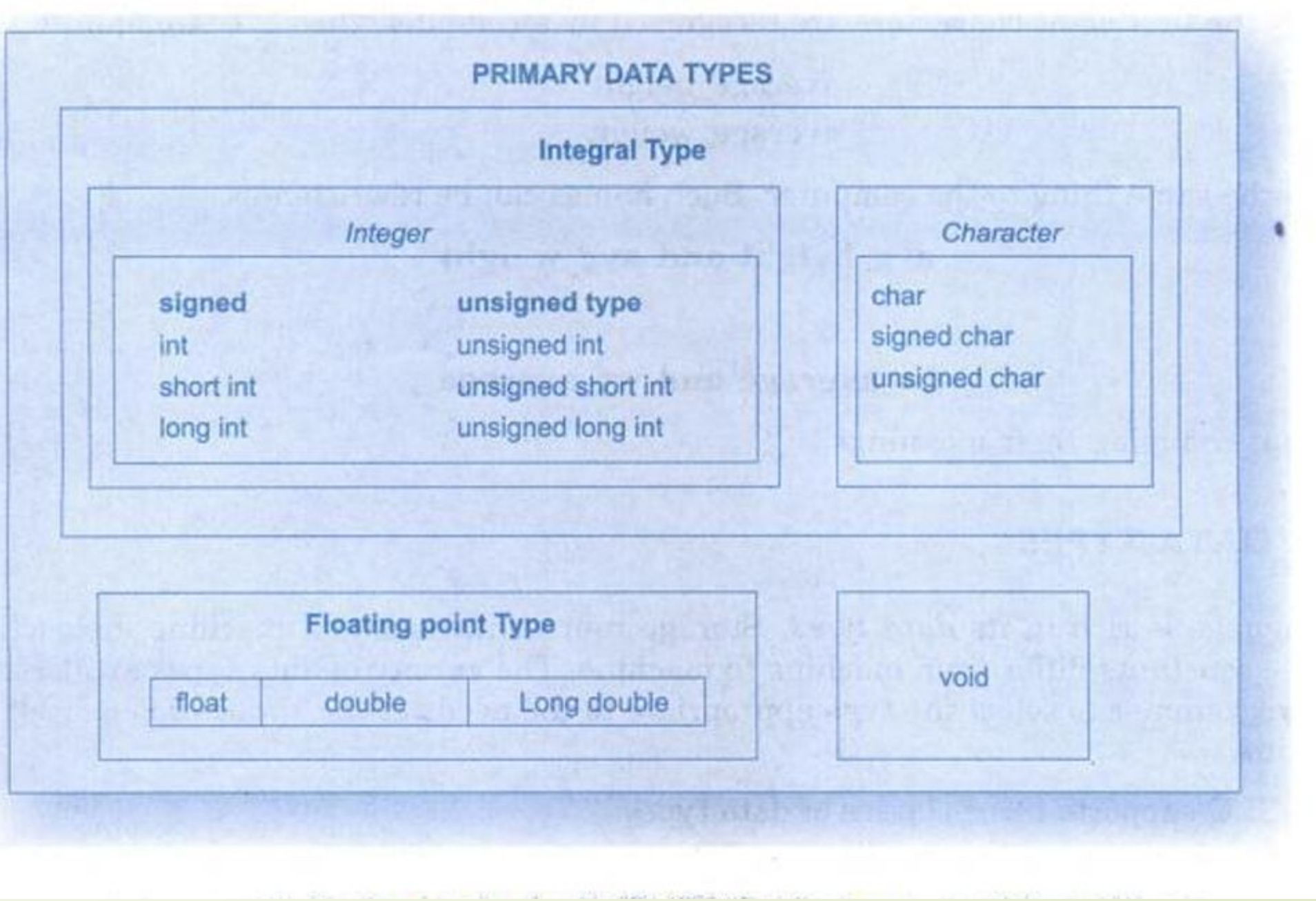


Fig. 2.2 *Basic types of C constants*

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


PRIMARY DATA TYPES
Integral Type
Integer
signed
int
short int
long int
unsigned type
unsigned int
unsigned short int
unsigned long int
Character
char
signed char
unsigned char
Floating point Type
float
double
Long double
void

Integer datatype

- The integer datatype in C is used to store the integer numbers(any number including positive, negative and zero without decimal part).
- Octal values, hexadecimal values, and decimal values can be stored in int data type in C.
- **Format Specifier:** %d

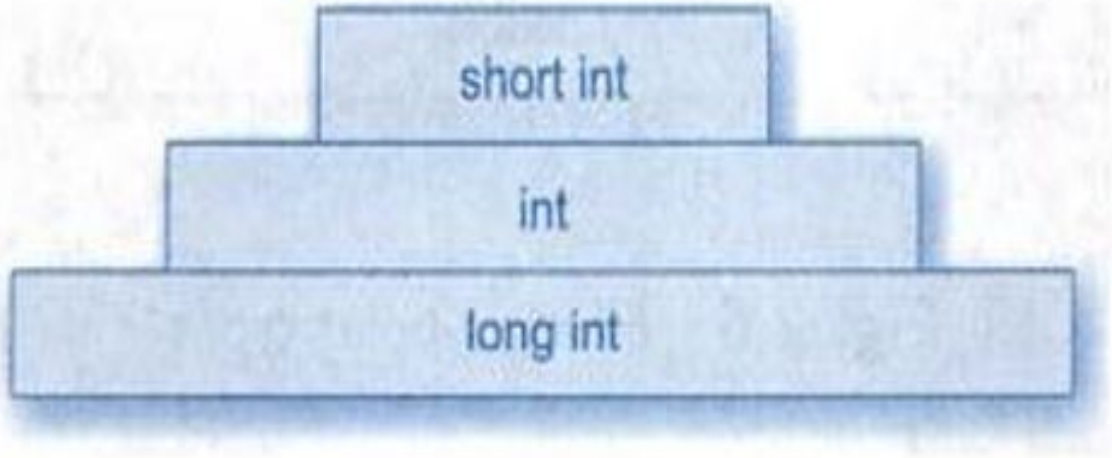


Fig. 2.5 *Integer types*

```
int var_name;
```

```c
#include <stdio.h>
int main()
{
        int a = 9;
       int b = -9;
       long int c = 99998;

     printf("Integer value with positive data: %d\n", a);
     printf("Integer value with negative data: %d\n", b);

     printf("Integer value with an long int data: %ld", c);

     return 0;
}
```

```c
#include <stdio.h>
int main()
{
     int a = 9;
    int b = -9;
    long int c = 99998;

   printf("Integer value with positive data: %d\n", a);
   printf("Integer value with negative data: %d\n", b);

   printf("Integer value with an long int data: %ld", c);

   return 0;
}
```

Output

```
Integer value with positive data: 9
Integer value with negative data: -9
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.
- **Format Specifier:** %c

```
char var_name;
```

```c
#include <stdio.h>

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

    printf("Value of tmp: %c\n",tmp);
     return 0;
}
```

```c
#include <stdio.h>

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

    printf("Value of tmp: %c\n",tmp);
return 0;
}
```

Output

```
Value of tmp: a
```

```
#include <stdio.h>
int main()
{
    char a = 'a’;
    char c;

    printf("Value of a: %c\n", a);
    a=a+1;
    printf("Value of a after increment is: %c\n", a);

    c = 99;
    printf("Value of c: %c", c);

    return 0;
}
```

```
#include <stdio.h>
int main()
{
    char a = 'a’;
    char c;

    printf("Value of a: %c\n", a);
    a=a+1;
    printf("Value of a after increment is: %c\n", a);

     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
```

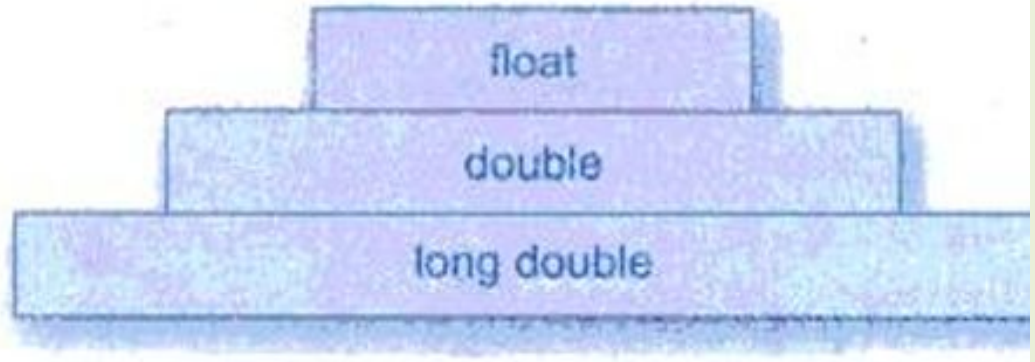


Fig. 2.6 *Floating-point types*

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.

- Format Specifier: %f

```
float var_name;
```

```
#include <stdio.h>

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

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

    return 0;
}
```

```
#include <stdio.h>

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

    // 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
```

Double Data Type

- A Double data type in C is used to store decimal numbers (numbers with floating point values) with double precision.
- **Format Specifier:** %lf

```
double var_name;
```

```c
#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;
}
```

```c
#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

```
examples
// function return type void
void exit(int check);

// Function without any parameter can accept void.
int print(void);
```

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
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu

char	1	-128 to 127	%c
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

- *The **long, short, signed and unsigned** are datatype modifier that can be used with some primitive data types to change the size or length of the datatype.*

Enumeration (or enum)

is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

To create an enum, use the enum keyword, followed by the name of the enum, and separate the enum items with a comma:

```
enum Level {  LOW,  MEDIUM,  HIGH};
```

To access the enum, you must create a variable of it.

```
enum Level myVar;
```

myVar should be assigned value which must be one of the items inside the enum (LOW, MEDIUM or HIGH):

```
enum Level myVar = MEDIUM;
```

By default, the first item (LOW) has the value 0, the second (MEDIUM) has the value 1, etc.

If you now try to print myVar, it will output 1, which represents MEDIUM:

- Enum default values can be changed

```
enum Level {
  LOW = 25,
  MEDIUM = 50,
  HIGH = 75
};
```

- if you assign a value to one specific item, the next items will update their numbers accordingly:

```
enum Level {
  LOW = 5,
  MEDIUM, // Now 6
  HIGH // Now 7
};
```

typedef

The C typedef keyword is used to redefine the name of already existing data types.

typedef existing_name alias_name;

```
typedef  int unit;
```

Now, we can create the variables of type **int** by writing the following statement:

```
unit a, b;
```

instead of writing:

```
int a, b;
```

```
/*..........DECLARATIONS.............................*/
    float     x,p ;
    double    y,q ;
    unsigned k ;
/*..........DECLARATIONS AND ASSIGNMENTS............*/
    int       m = 54321 ;
    long int n = 1234567890 ;
/*..........ASSIGNMENTS.............................*/
    x = 1.234567890000 ;
    y = 9.87654321 ;
    k = 54321 ;
    p = q = 1.0 ;
```

```
/* Example of storage classes */
int m;
main()
{
   int i;
   float balance;
   ....
   ....
   function1();
}

function1()
{
   int i;
   float sum;
   ....
   ....
}
```

m is external or global variable

i, balance, sum are local variables

Table 2.10 *Storage Classes and Their Meaning*

Storage class	*Meaning*
auto	Local variable known only to the function in which it is declared. *Default is auto.*
static	Local variable which exists and retains its value even after the control is transferred to the calling function.
extern	Global variable known to all functions in the file.
register	Local variable which is stored in the register.

Static and external (**extern**) variables are automatically initialized to zero. Automatic (**auto**) variables contain undefined values (known as 'garbage') unless they are initialized explicitly.

Declaring a variable as a constant

As the name suggests, a constant in C is a variable that cannot be modified once it is initialized in the program.

```
const data_type var_name = value
```

```
const int a=10;
```

Defining Symbolic Constants

```
#define PI 3.14159
```

```
#define PASS_MARK 50
```