**Type Casting:**

Type Casting in C is used to convert a variable from one data type to another data type, and after type casting compiler treats the variable as of the new data type.

Syntax:

(type\_name) expression

## Without Type Casting

Example:

#include <stdio.h>

main ()

{

int a;

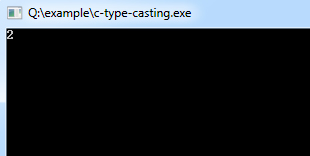
a = 15/6;

printf("%d",a);

}

Program Output:

In the above C program, 15/6 alone will produce integer value as 2.



## After Type Casting

#include <stdio.h>

main ()

{

float a;

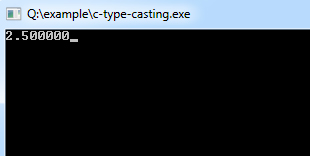
a = (float) 15/6;

printf("%f",a);

}

Program Output:

After type cast is done before division to retain float value 2.500000.



It's best practice to convert lower data type to higher data type to avoid data loss.

**Arrays:**

The array is a data structure in C programming, which can store a fixed-size sequential collection of elements of the same data type.

For example, if you want to store ten numbers then instead of defining ten variables, it's easy to define an array of 10 lengths.

In the C programming language, an array can be*One-Dimensional*, *Two-Dimensional* and *Multidimensional*.

Define an Array in C

Syntax:

type arrayName [ size ];

This is called a one-dimensional array. An array type can be any valid C data types, and array size must be an integer constant greater than zero.

Example:

double amount[5];

Initialize an Array in C

Arrays can be initialized at declaration time:

int age[5]={22,25,30,32,35};

Initializing each element separately in a loop:

int myArray[5];

int n = 0;

// Initializing elements of array seperately

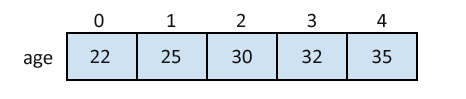
for(n=0;n<sizeof(myArray);n++)

{

myArray[n] = n;

}

A Pictorial Representation of the Array:



Accessing Array Elements in C

Example:

int myArray[5];

int n = 0;

// Initializing elements of array seperately

for(n=0;n<sizeof(myArray);n++)

{

myArray[n] = n;

}

int a = myArray[3]; // Assigning 3rd element of array value to integer 'a'.

**String:**

In C programming, the one-dimensional array of characters are called strings, which is terminated by a null character '\0'.

Strings Declaration in C

There are two ways to declare a string in C programming:

Example:

Through an array of characters.

char name[6];

Through pointers.

char \*name;

Strings Initialization in C

Example:

char name[6] = {'C', 'l', 'o', 'u', 'd', '

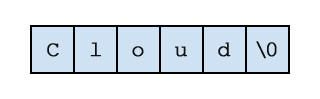
char name[6] = {'C', 'l', 'o', 'u', 'd', '\0'};

'};

or

char name[] = "Cloud";

Memory Representation of Above Defined String in C



Example:

#include<stdio.h>

int main ()

{

char name[6] = {'C', 'l', 'o', 'u', 'd', '

#include<stdio.h>

int main ()

{

char name[6] = {'C', 'l', 'o', 'u', 'd', '\0'};

printf("Tutorials%s\n", name );

return 0;

}

'};

printf("Tutorials%s\n", name );

return 0;

}

Program Output:

TutorialsCloud

**Pointers:**

A pointer is a variable in C, and pointers value is the address of a memory location.

## Pointer Definition in C

Syntax:

type \*variable\_name;

Example:

int  \*width;

char  \*letter;

## Benefits of using Pointers in C

* Pointers allow passing of arrays and strings to functions more efficiently.
* Pointers make it possible to return more than one value from the function.
* Pointers reduce the length and complexity of a program.
* Pointers increase the processing speed.
* Pointers save the memory.

## How to use Pointers in C

Example:

#include<stdio.h>

int main ()

{

int n = 20, \*pntr;  /\* actual and pointer variable declaration \*/

pntr = &n;  /\* store address of n in pointer variable\*/

printf("Address of n variable: %x\n", &n );

/\* address stored in pointer variable \*/ printf("Address stored in pntr variable: %x\n", pntr );

/\* access the value using the pointer \*/ printf("Value of \*pntr variable: %d\n", \*pntr );

return 0;

}

Address of n variable: 2cb60f04

Address stored in pntr variable: 2cb60f04

Value of \*pntr variable: 20

**Structure:**

The structure is a user-defined data type in C, which is used to store a collection of different kinds of data.

* The structure is something similar to an array; the only difference is array is used to store the same data types.
* *struct* keyword is used to declare the structure in C.
* Variables inside the structure are called *members of the structure*.

Defining a Structure in C

Syntax:

struct structureName

{

//member definitions

};

Example:

struct Courses

{

char WebSite[50];

char Subject[50];

int Price;

};

Accessing Structure Members in C

Example:

#include<stdio.h>

#include<string.h>

struct Courses

{

char WebSite[50];

char Subject[50];

int Price;

};

void main( )

{

struct Courses C;

//Initialization

strcpy( C.WebSite, "cools.in");

strcpy( C.Subject, "The C Programming Language");

C.Price = 0;

//Print

printf( "WebSite : %s\n", C.WebSite);

printf( "Book Author : %s\n", C.Subject);

printf( "Book Price : %d\n", C.Price);

}

Program Output:

WebSite : cools.in

Book Author: The C Programming Language

Book Price : 0