C for Beginners

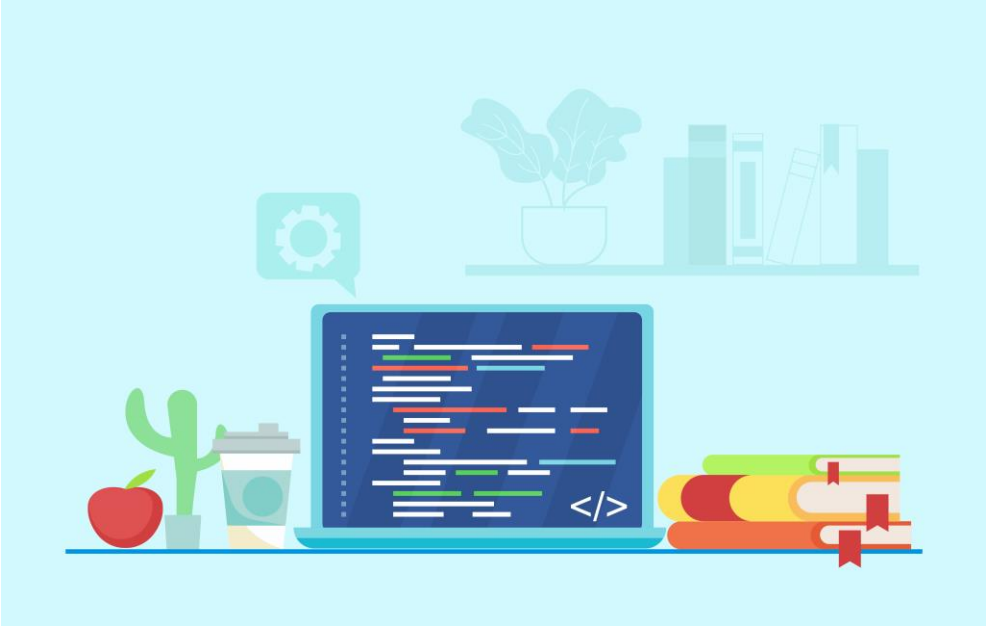
Agenda

- Introduction to C
- Variables in C
- Datatypes in C
- Input / Output in C
- Operators in C
- C Control Statements
- Arrays in C
- Functions in C
- Strings in C
- Structures and Union
- Pointers in C

Introduction to C

Introduction to C

- C was first introduced by Dennis Ritchie.
- C is a procedure-oriented language.
- C is a widely used and popular programming language for developing system application software.
- It can be called as a mid-level programming language.

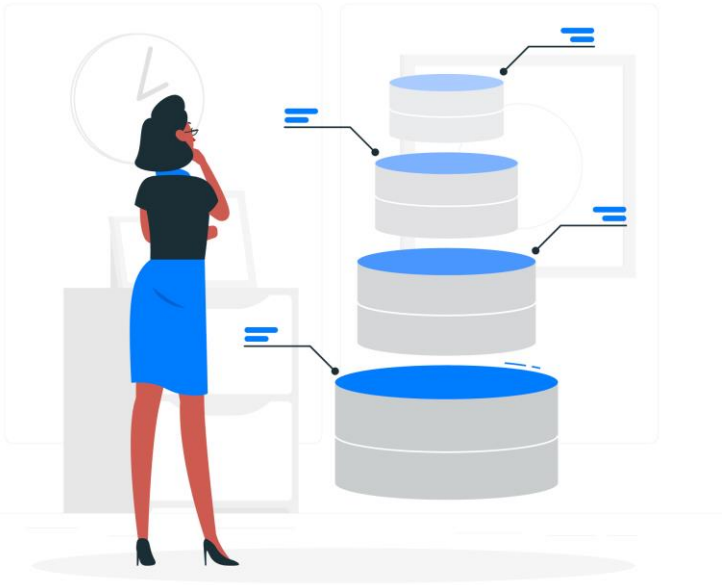


Introduction to C

Features:

Some of the features of C Programming are:-

- Easy to learn
- Structured Programming language
- Machine Independent or Portable
- Powerful
- Used in low- level as well as high level applications



Basic block of a C Program

```
Preproccesor directives

Global declarations;

void main()
{

local declarations;
statement 1;
statement 2;
        :
statement n;
}

User defined functions
```

Comments

```
/* This is my First C Program
 Author name:                */

# include<stdio.h>
#include<conio.h>

void main()
{

printf("C Programming");              // Single line Comment

getch();

}
```

C Tokens

C tokens are smallest individual units in a program.

- Identifiers- user defined names/abbreviations
- Keywords - words which are specific to C language
- Constants - values which don't change
- Strings - sequence of characters
- Operators - act on operands and generates output
- Special Symbols - used for preprocessor directives (#)

Variables in C

Variables in C

What is a Variable?

- It’s a name given to a memory location where the value is not fixed, it can change with the course of time. Ex : int num;

Syntax:

datatype variable_name;

Assignment:

variable_name = value

Variables in C

Scope of a Variable

- Scope generally speaks about the visibility of variables.
  - local variable and global variable

Local

```
void main ()
{

int rollno = 10;

}
```

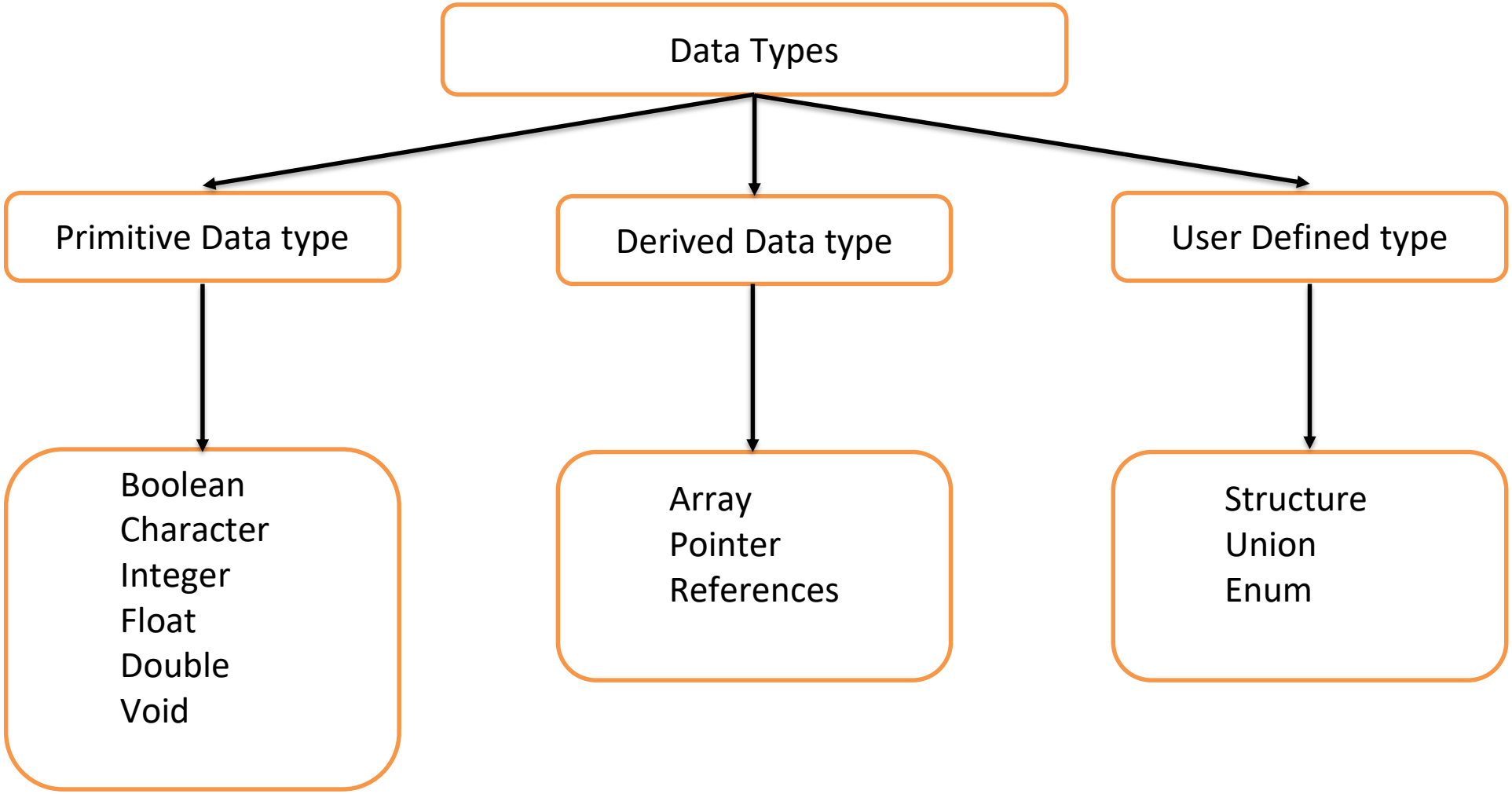
Global

```
int y = 50;

void main ()
{
int x = 100;
}
```

Data Types in C

Data Types in C



Data Types in C

Typecasting

- C allows for conversions between the basic data types
  - Implicit and Explicit

```
int num = 5 + 13.75;   // num = 18
```

Implicit conversion

```
float a = 5.25;
int b = (int) a; // b = 5
```

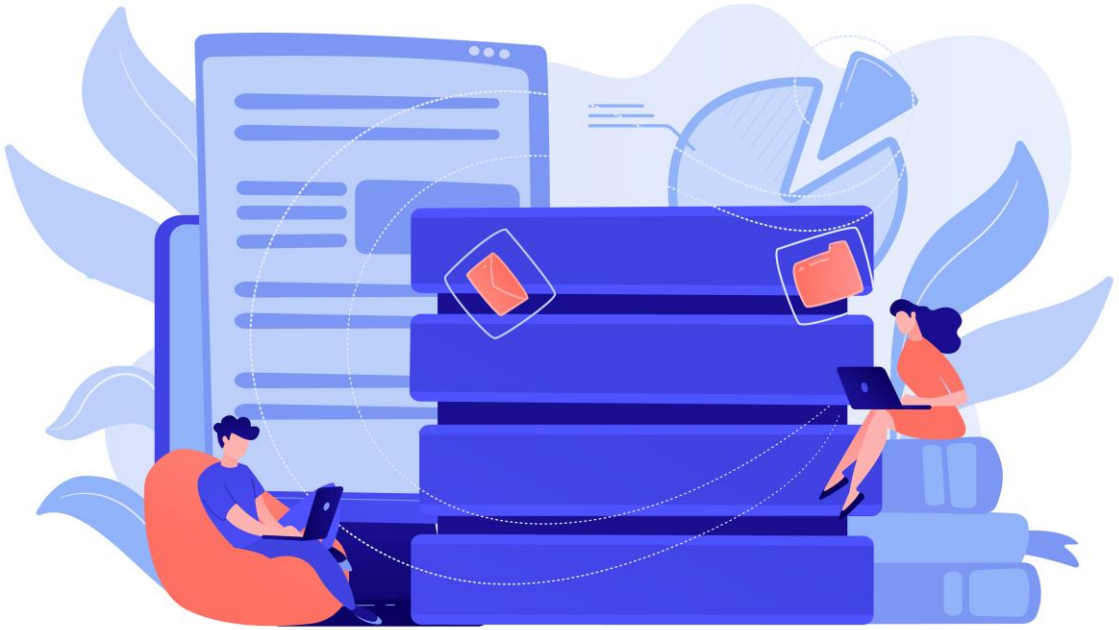
Explicit conversion

Input / Output in C

Input / Output in C

Input and Output functions in C are:-

- printf() and scanf()
- gets() and puts()
- getchar() and putchar()



Input / Output in C

Format Specifiers:-

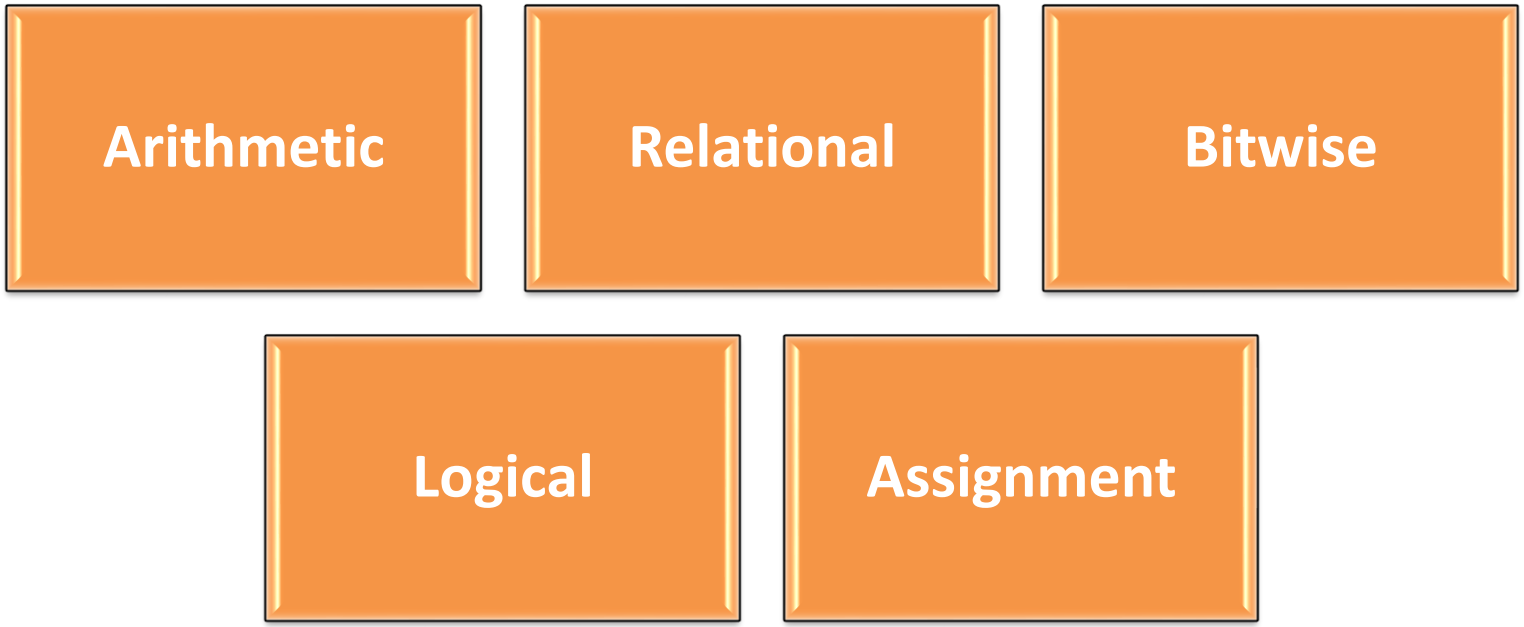
The format specifiers will inform the scanf() function, what kind of input is being fed through input device. It also tells printf() function what type of data has to be printed on the output device.

- %d – Integer
- %c – Character
- %s – String
- %f - Float

Operators in C

Operators in C

- C primarily supports 5 types of operators



Operators in C

Arithmetic Operators:-

- Used for mathematical operations
- We have unary and binary operators
- Unary – (++, --)
- Binary – (+ , - , *, /, %)

Relational Operators:-

- Used to compare the values of two operands
- Some of them are (< , <= , >=, ==)
- We can also check for equality of operands, whether an operand is greater or smaller than the other.

Operators in C

Logical Operators:

- To combine one or multiple conditions we can use Logical operators.
- Logical operators always give us a Boolean value, i.e. true or false
- Some of them are (&& - AND, || - OR, ! – NOT)

Bitwise Operators:-

- We can perform bit-level operations on the operands.
- Some of them are (& - bitwise AND, |- bitwise OR)
- It converts them to bit-level first and then performs the operations.

Operators in C

Assignment Operator:-

- It is used to assign a value to a given variable.
- The value on the right side of the assignment operator is assigned to operand at left side.
- We also have some short hand assignment operators (+=, -=, /=).

.

C Control Statements

C Control Statements

Conditional Statements:-

- Also known as selection statements.
- Based on the conditions specified we can make some decisions.
- Anytime we execute conditional statements we get a Boolean value in return.
- If the condition executed returns a true value, a set of statements are executed else another set of statements are executed.

C Control Statements

The if statement:-

- Selection and execution of statements based on a given condition is done by the **if** statement.

Syntax:

```
if (condition)
{
        statement 1;
        statement 2;
}
```

C Control Statements

The if-else statement:-

- Depending on the result of the condition, the **if - else** statement executes one of the two potential statements.

Syntax:

```
if (condition)
{
        statement 1;            // if block
        statement 2;
}
else
{
        statement 3:            // else block
}
```

C Control Statements

The Nested if-else statement:-

- A nested if-else contains one or more if-else statements.

```
if (condition1)
{
        statement 1;
        if(condition2)
                statement 2;
        else
                statement 3;
}
else
{
        statement 3:
}
```

C Control Statements

Loops:-

- Loops are used to re-execute a part of a code a given number of times, depending upon the condition specified.

Entry controlled:-

- The condition is checked each-time before entering the loop. If the condition is satisfied then only the loop body gets executed. The loop body does not get executed if the condition is false in the first iteration.

C Control Statements

Entry controlled loops are:-

1) **For loop**

Syntax:

```
for(i=0 ;i<n; i++){
//do something
}
```

2) **While Loop**

Syntax:

```
while(condition is True){
//do something
}
```

C Control Statements

Switch case:-

- The switch statement works as a multiway branch statement.
- It’s a modified form of if-else.
- It’s a common alternative to the if statement when you want to get multiple results.
- Default condition gets executed when no conditions are met.

C Control Statements

Switch case:-

```
#include<stdio.h>
#include<conio.h>

void main() {
int n= 1;

switch (n)
{
    case 1: Statement 1;
            break;
    case 2: Statement 2;
            break;
     default: Statement;
}
```

Arrays in C

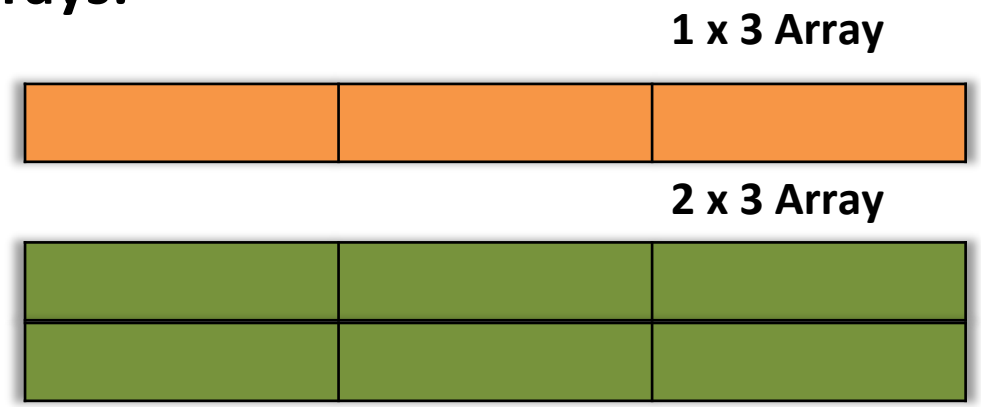
Arrays in C

- An array is a collection of elements where the data types of all the elements are same.
- An array stores all the elements in contiguous memory locations.
- Every element in an array has a specified index.
- Size of an array in C is fixed at the time of definition

We have two types in Arrays:

Single dimensional

Multi-dimensional



Arrays in C

Definition:

An array can be defined in the following manner in C:

Data_type name_of_array>[<size_of_array>]

Note: Size of an array should always be an integer it cannot be a real value.

Example:

```
int a[10];          //single dimensional array
float b[3][3];      //float - dimensional array
```

Arrays in C

Array initialization:

1) Static Initialization

```
int a[10]={0,1,2,3,4,5,6}

char c[5]={‘h’,’e’,’l’,’l’,’o’}
```

2) Dynamic Initialization

```
int a[10];              //array is created with garbage value
for (int i=0 ;i<10; i++)
scanf(“%d”,&a[i]);
```

Functions in C

Functions in C

- Functions are blocks of code which are used to perform specific tasks .
- In C a function needs to be declared before it’s used.
- Functions have a function definition , function body and a return type.
- Functions with return type except void needs to return a value at the end.
- Functions with return type void do not return any value.
- A recursive function can call itself during the course of execution of a program.

Functions in C

A Function example:

```
int sum (int a, int b)
{
        return a+b;
}

void main()
{
        printf(" %d", sum(5,10));
}
```

Strings in C

Strings in C

- Strings are used to store sequence of characters as information.
- string.h header file provides some built-in string manipulation functions.

Strings can be initialized in following ways:-

```
char a[5] = {‘H’,’o’,’l’,’a’};

char a [ ] = “Hola”;
```

Strings in C

String functions are accessible by importing **<string.h>**.

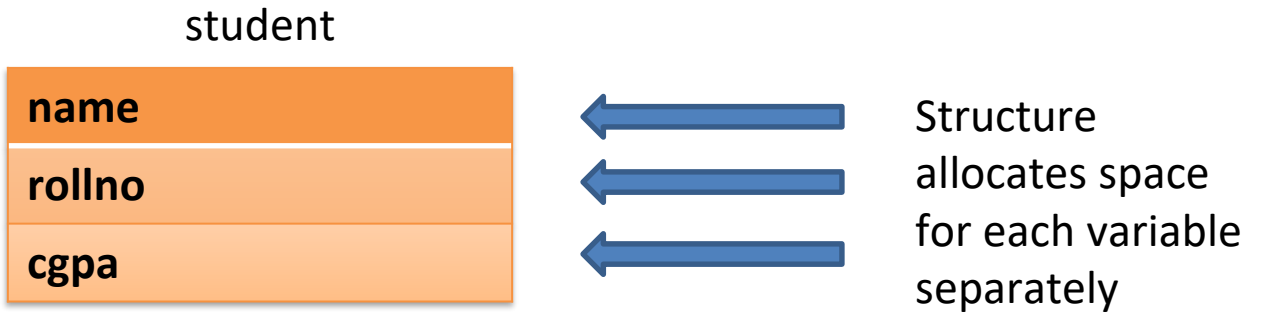
- strlen() Find length of string
- strcpy() Copy one string to other
- strcat() Join two strings
- strcmp() Compare two strings
- strlwr() Converts string to lowercase
- strupr() Converts string to uppercase

Structures and Union

Structures and Union

- Structures provides a way for us to create our own data type called " user-defined data type"
- Assume that we have to store the details of a student, so we can use structure to store it as below:

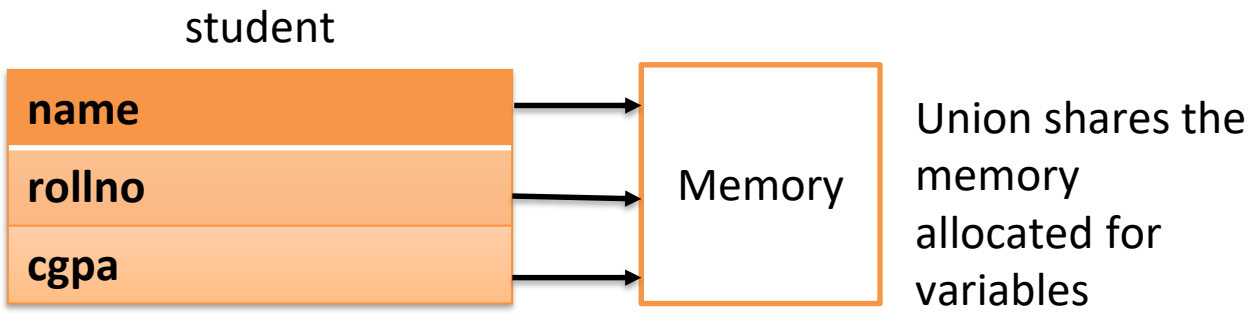
```
struct student
{
        char name[10];
        int rollno;
        float cgpa;
} ;
```



Structures and Union

- Union also provides a way for us to create our own data type called “ user-defined data type”.
- Here each variable shares the memory collectively.

```
union student
{
        char name[10];
        int rollno;
        float cgpa;
} ;
```



Pointers in C

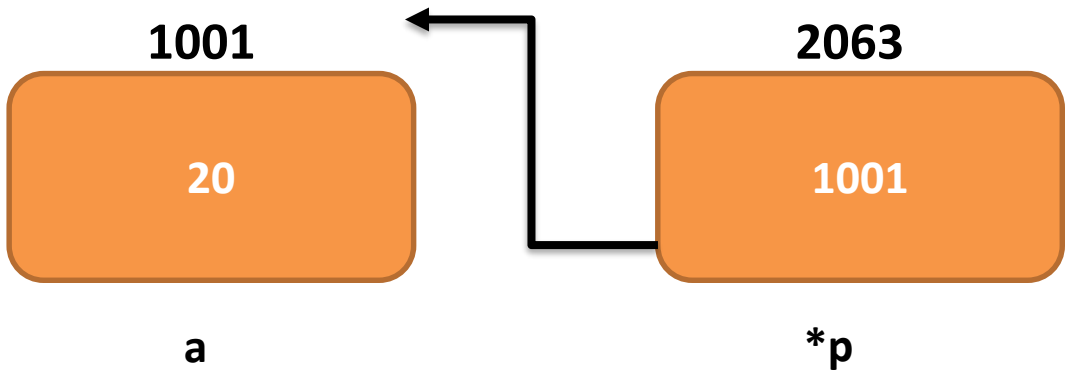
Pointers in C

- Pointers are variables which store the address of a variable.

- They have data type just like variables, for example an integer type pointer can hold the address of an integer variable and an character type pointer can hold the address of char variable.

Example:

```
int a =20;
int *p=&a;
```



Pointers in C

Example of Pointer:

```
#include <stdio.h>
void  main ()
{

int y = 5;
int *p;
p = &y;
printf("Address of y : %x\n", &y );
printf("Content of p: %x\n", p );
printf("Content of *p: %d\n", *p );

}
```

Summary

- We saw an introduction to C, features available in C and tokens available in C.
- Then, we started with variables, how to declare and use them.
- Next, we came across, what are the data types which we can use here in C language.
- Then we went through the input and output functions, how can we process the input and display results.
- Operators available in C through which we can perform various operations on our data.
- Various controls statements such as if, if..else, loops and switch case.
- Next, we introduced arrays in C, how to declare and use them with examples.
- Next up, we saw what functions are, how to use them in the program and recursion.
- Later we came across Strings, how to declare and built-in functions available in string.h.
- We introduced Structures and Union with an example for each.
- A brief introduction to pointers, how to declare and use them with example.