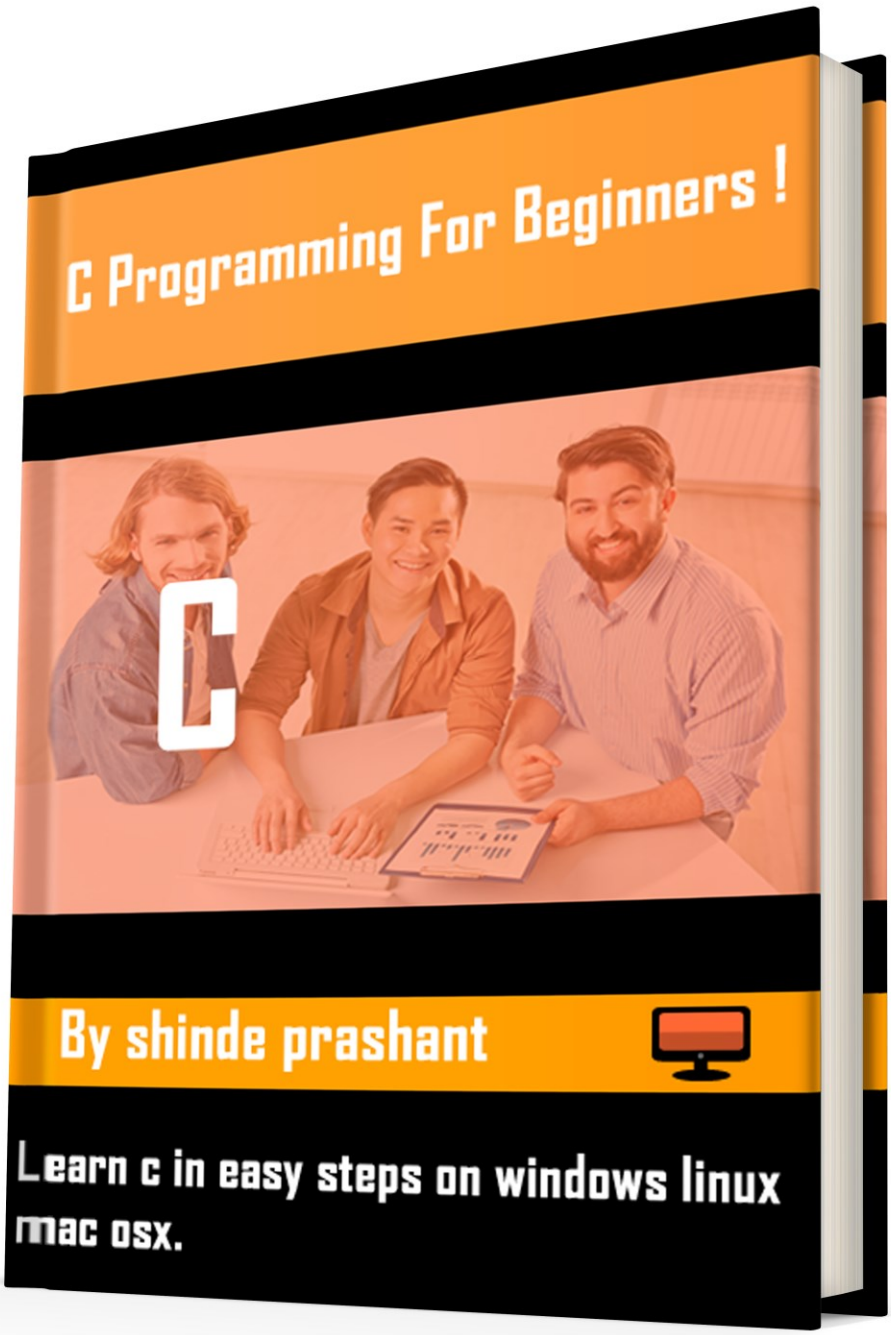


C Programming for Beginners !

A simple & practical guide to learn C Programming within 10 days on Windows/Linux/Mac os .

By Prashant Shinde .

Special thanks to freekpik.com for providing this image license for free .

About the Author

My name is Prashant , I am a passionate computer geek who has mastered different fields of computer studies . My Journey began back in 2009 when I was studying my Bsc II , I joined a computer course where I learned the basics of computers & programming and from there spring greater passion for computers & programming and gained extensive experience in different fields of computer like website design, software development with C & C++, Ethical Hacking, Ubuntu Linux Server Administration.

In the last few years I have taught more than 20000+ students online on Udemy. My Inspiration behind writing this eBook is to give some contribution to the programming world. This eBook will help all the programming lovers to learn the fundamentals of C programming absolutely free. This is the first version of this eBook which will be absolutely free for lifetime. I hope this eBook will help all the geeks to get started in programming.

BRIEF CONTENTS

TABLE OF CONTENTS

1. INTRODUCTION .

HISTORY & IMPORTANCE OF C PROGRAMMING LANGUAGE .

C is the most popular programming language in the world . It was developed in early 1972 at At & T bell laboratory by Dennis Ritchie . Before C programming language there are different programming languages were available in the market like algol , bcpl but these programming languages were having their own limitations like “Algol” was used to design only scientific softwares , BCPL was used to write system related softwares or in system programming . You could not be able to write system softwares with algol or you could not be able to write Scientific softwares with the help of BCPL . So at this time a need of such a programming language is arised which can be used in every field like for writing application software , for writing system softwares or for writing scientific softwares . So due to this need c programming language is developed in 1972 by Dennis Ritchie .

Following is the Evolution History of C Programming Language.

Year	Language	Developer
1960	Algol	International Group .
1967	BCPL	Martin Richards .
1970	B	Ken Thompson.
1972	Traditional C	Dennis Ritchie .
1978	K &Rc	Kernighan & Ritchie .
1989	ANSI C	ANSI Committee .
1990	ANSI	ISO C	ISO Committee .
1999	C99	Standardization Committee .

Importance of C Programming language .

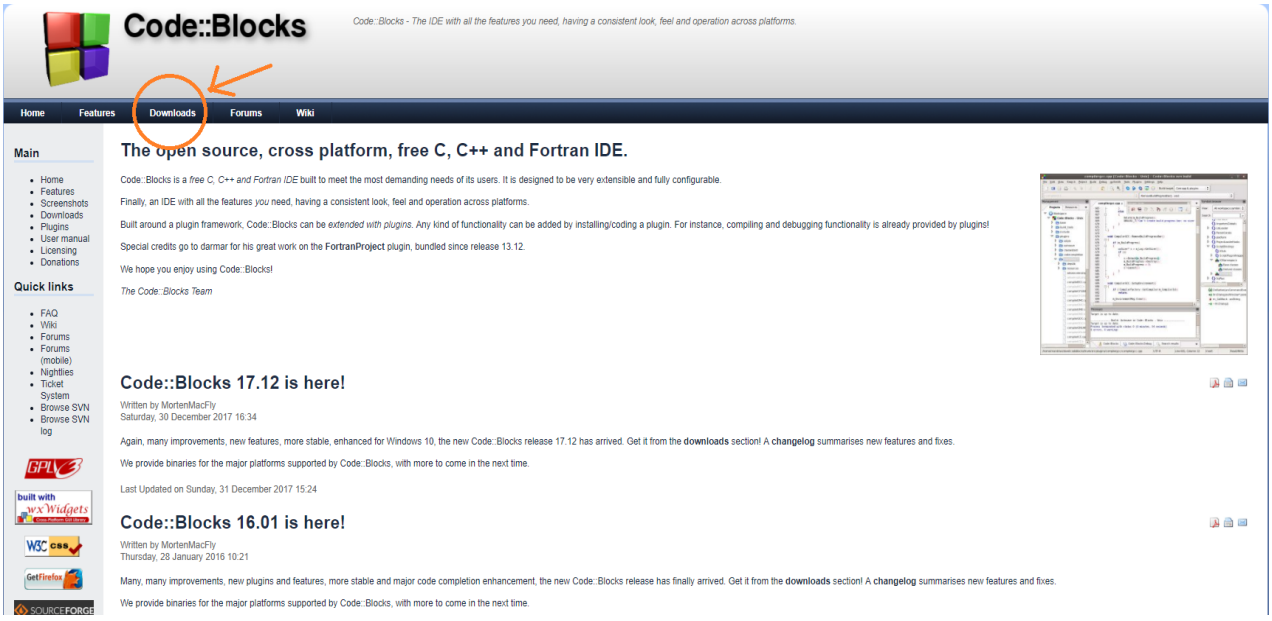
- C is such a flexible programming language which can used to design any type of software or application .
- You can develop operating systems with the help of c programming language .
- You can develop computer games with the help of c programming language .
- You can use it write application softwares like student database management system.
- You can use c programming language in developing embedded systems .

- C can be a best start to learn programming before moving to c++ , java , php .

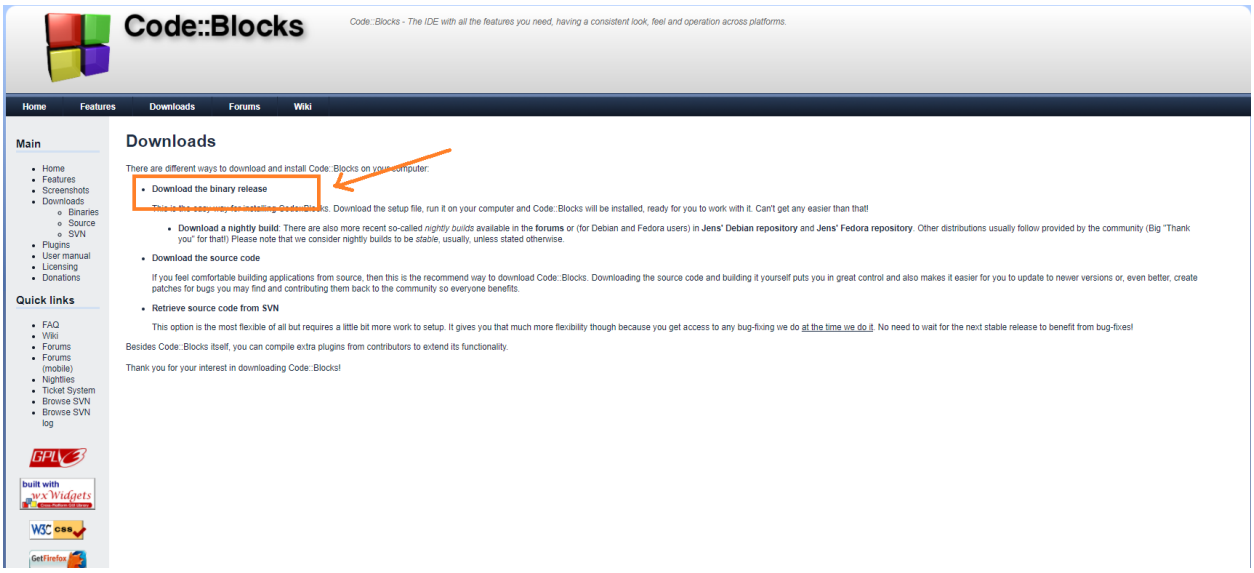
Setting up the Environment .

To Setup the c programming environment you need to download codeblocks IDE to write & execute the c programs . It is a modern IDE to Learn C & C++ Programming language . But it doesn't support all the features of c programming . if you wanting to learn c programming in depth then you would have to go with turboc++ compiler , it is a 16 bit compiler . But it comes with an emulator so you can run it on either 32 or 64 bit environment . So to install the codeblocks IDE on your system just follow the following steps .

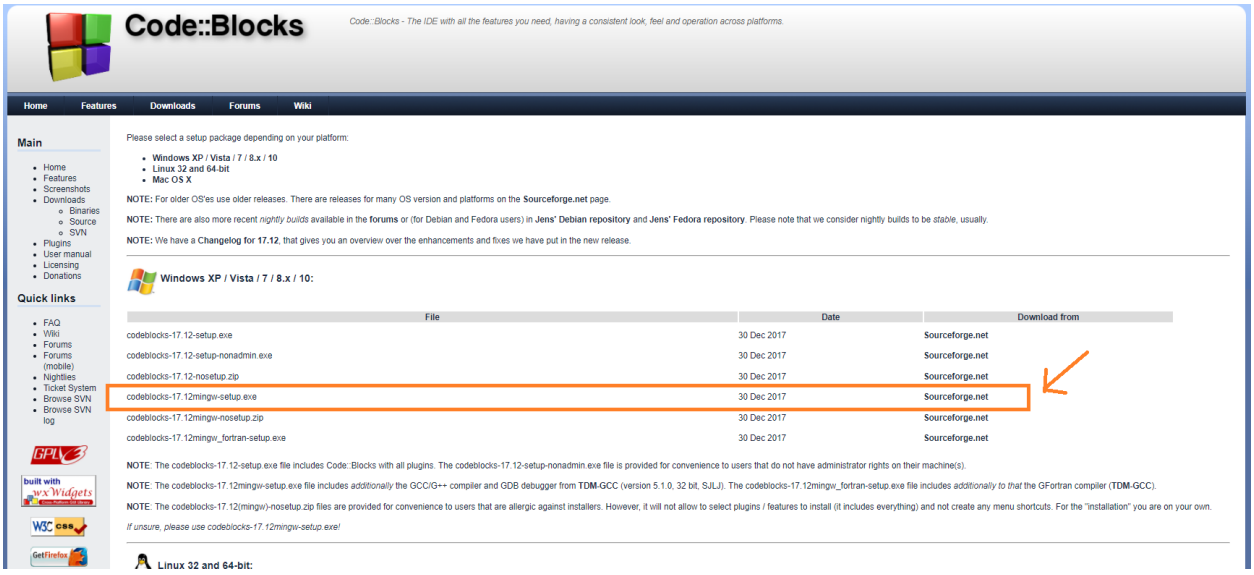
First goto www.codeblocks.com & you will find the following web page there . So click on downloads link as shown in the image below . Please zoom out the page if you can't see it properly .



After clicking on downloads , you will get a new page as shown in the image below . Here you need to click on " Download the binary release " .

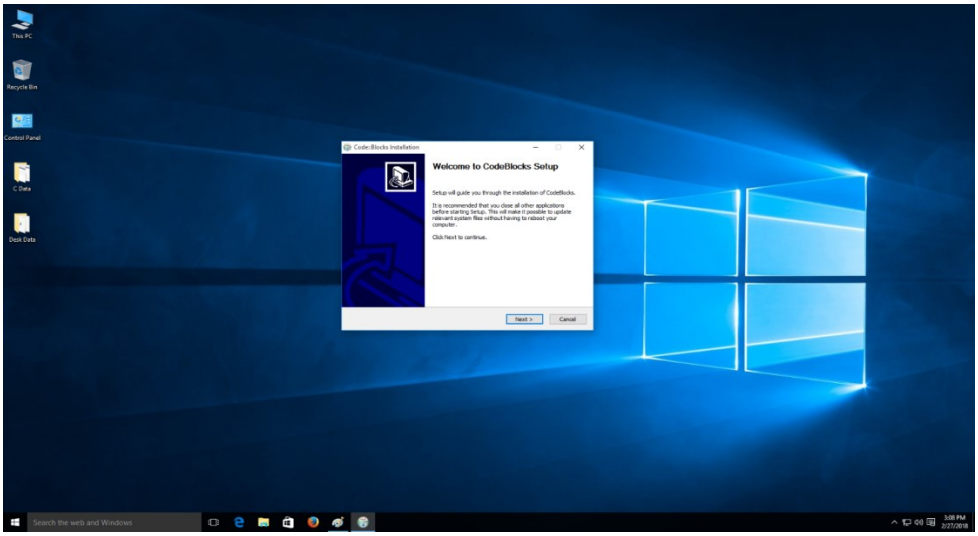


Now after clicking on Download the binary release you will get a new page . Here you will find different download options but you need to download the Option as shown in the image below .

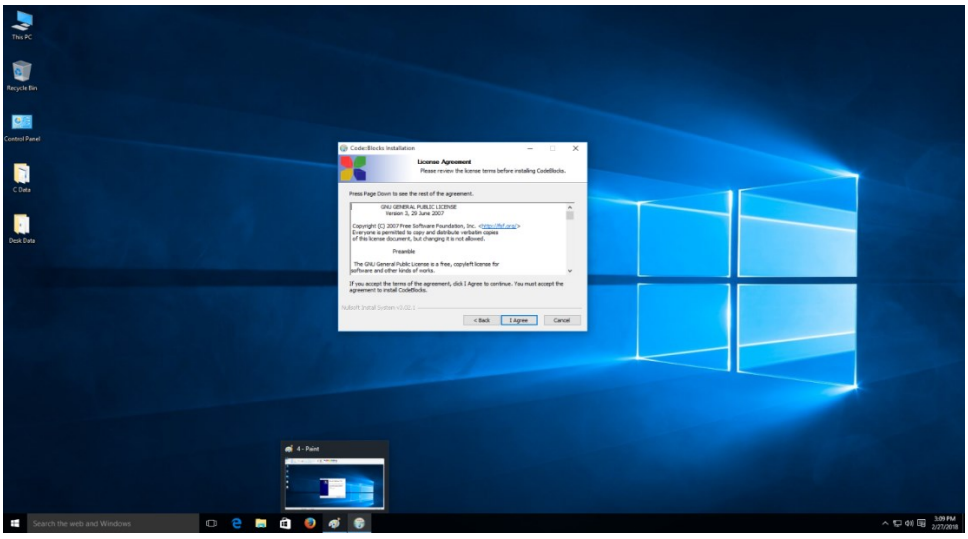


You just need to click on sourceforge.net & you will be redirected to another page & your download will begin automatically . If you don't then you will be able to see restart download option on that new page , so click on restart downloads .

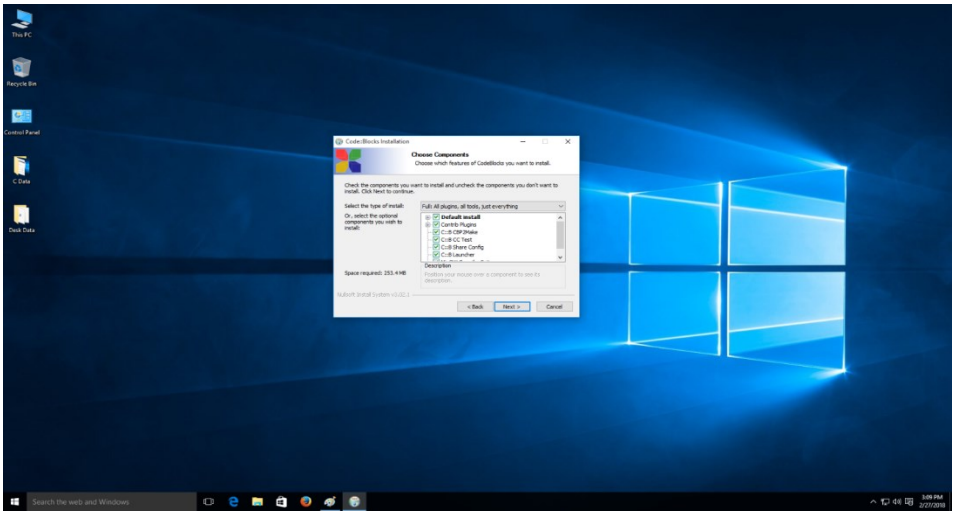
Now after downloading , double click on the executable file , then you will see following window , click on Next .



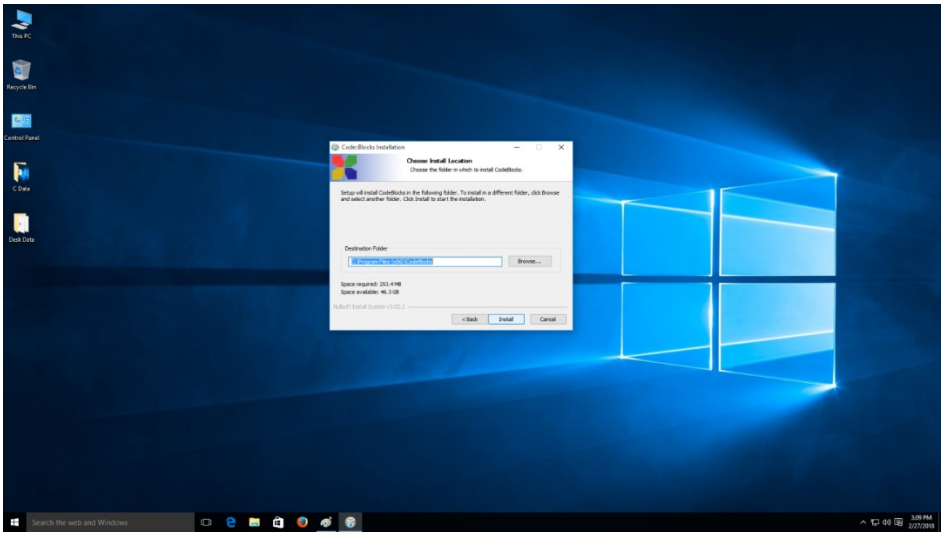
Click on " I Agree " .



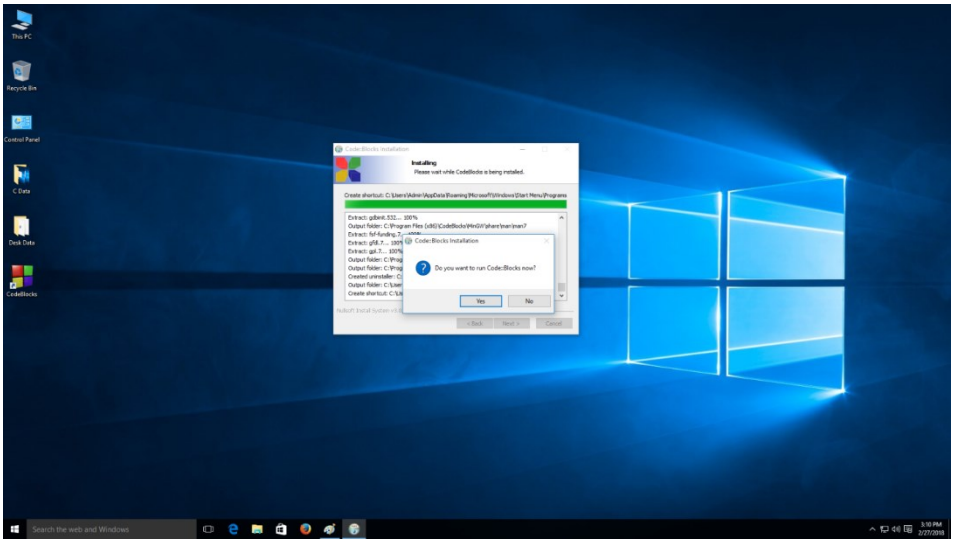
Then click on " next " .



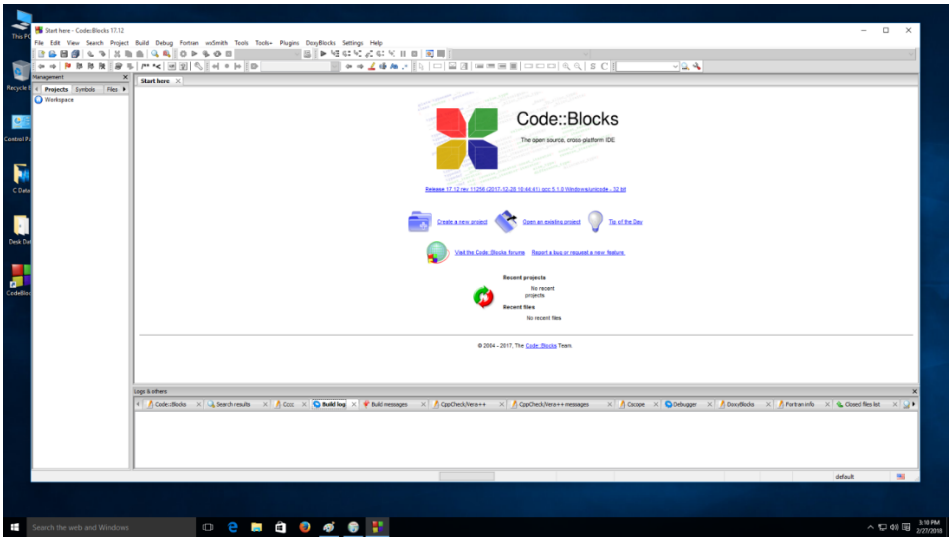
Now click on “ Install ”.



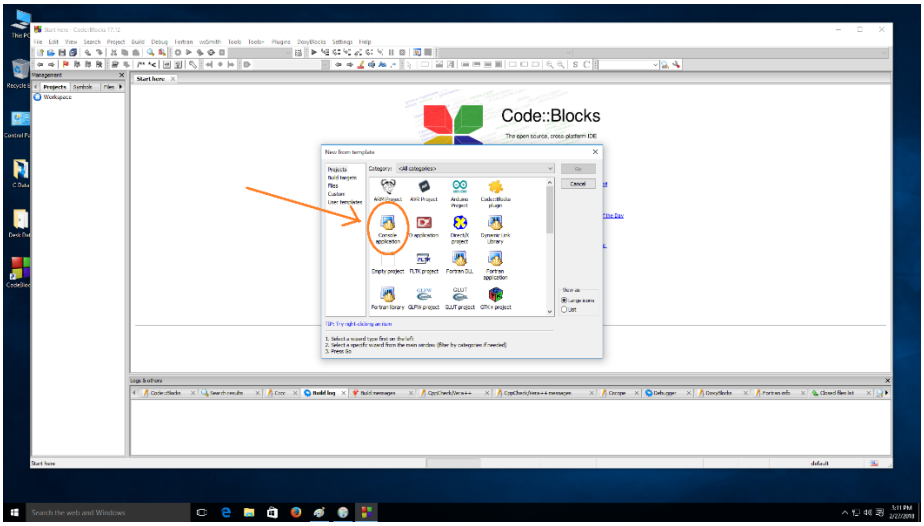
Then After installation it will ask you for do you want to start codeblocks now , then click on Yes .



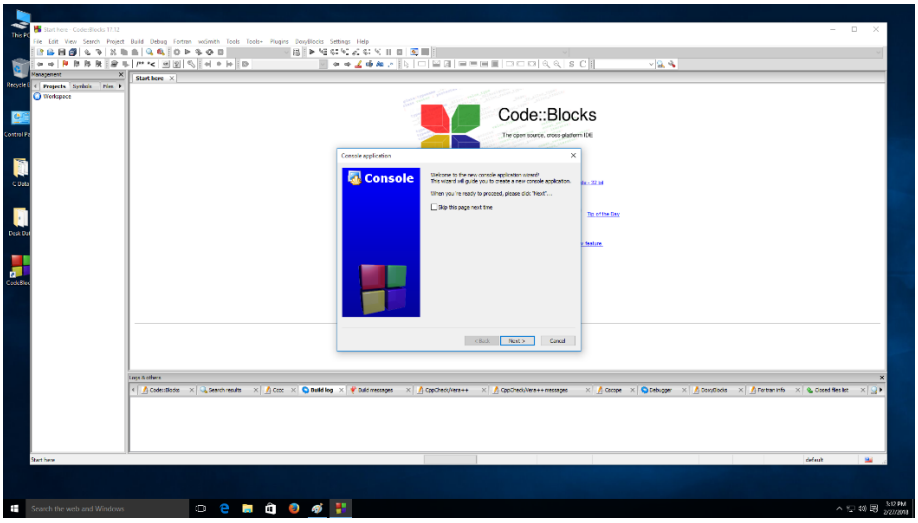
codeblocks will start with this window , here you need to click on “Create new project ” .



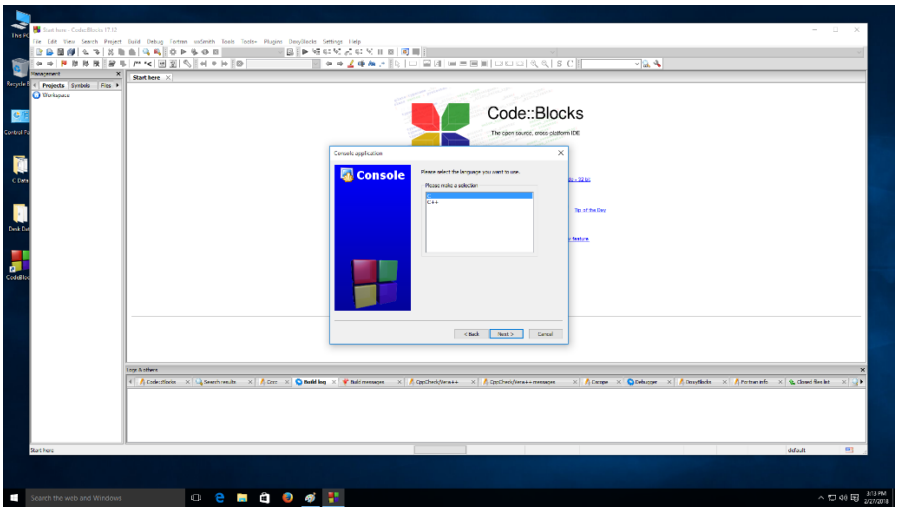
Then after clicking on “ Create new project ” , You will get the following windows , here you need to select “ console application ” as shown in image below .



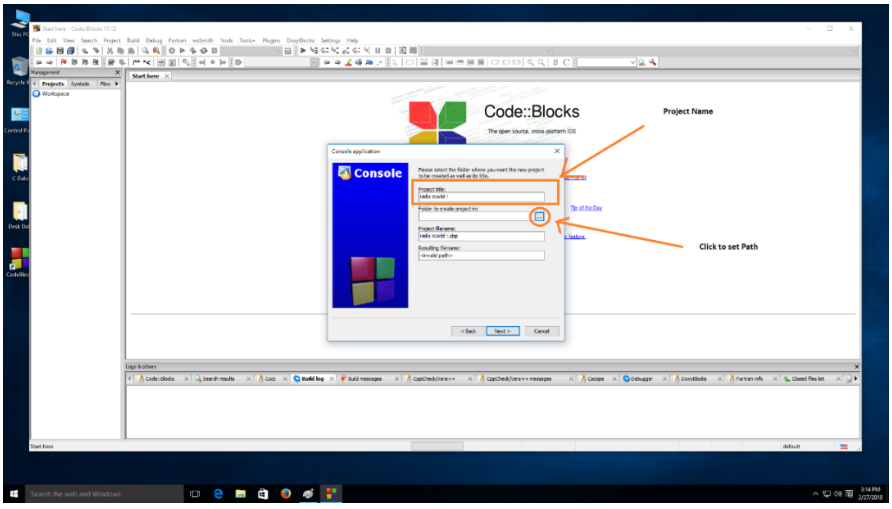
After clicking on console application , you will get following window . Here click on next .



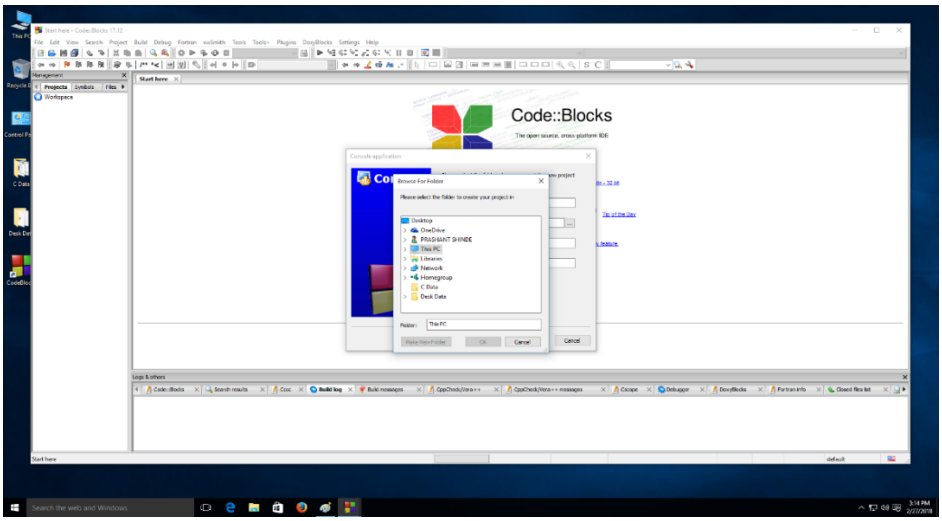
After clicking on next , you will get the below window , here you need to select " C " & click on next .



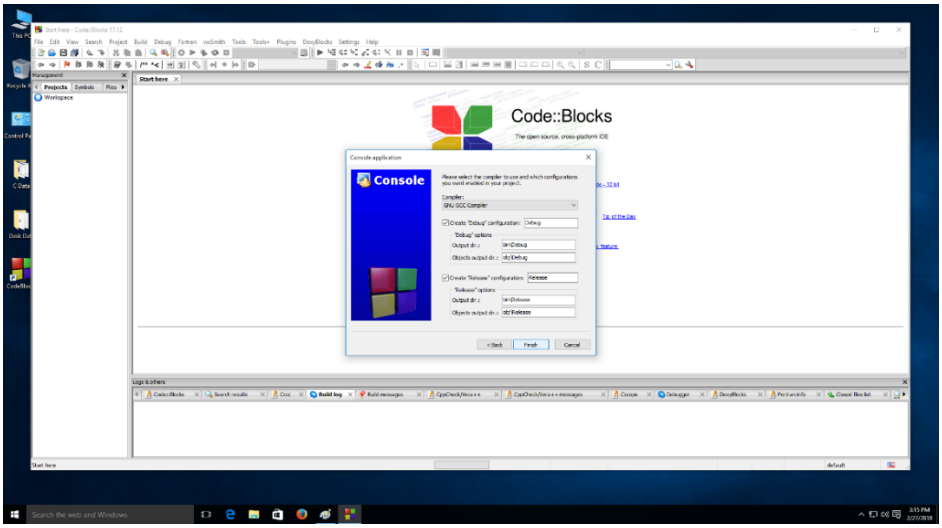
Ok after clicking on next you will get following window , Here You need to add the name for your project & Also specify the path or directory for the project . So as you can see below i have added the Project name as " Hello World ! " . You can specify what ever name you want for your project .



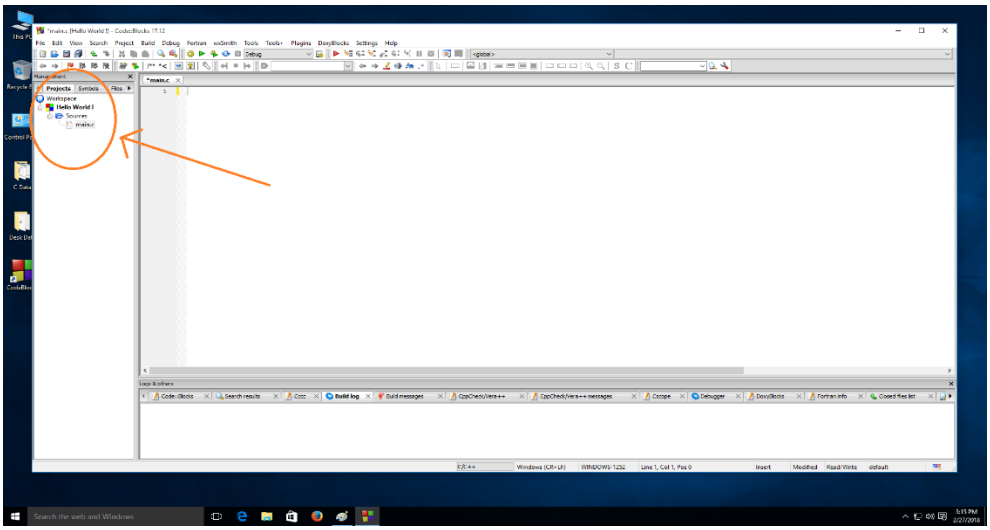
After setting project name , now it's time to set path for your project , means you can save your project in a particular directory , so that you can access you project easily . So as you can see in above image click on the small squar as pointed & you will get the following window .



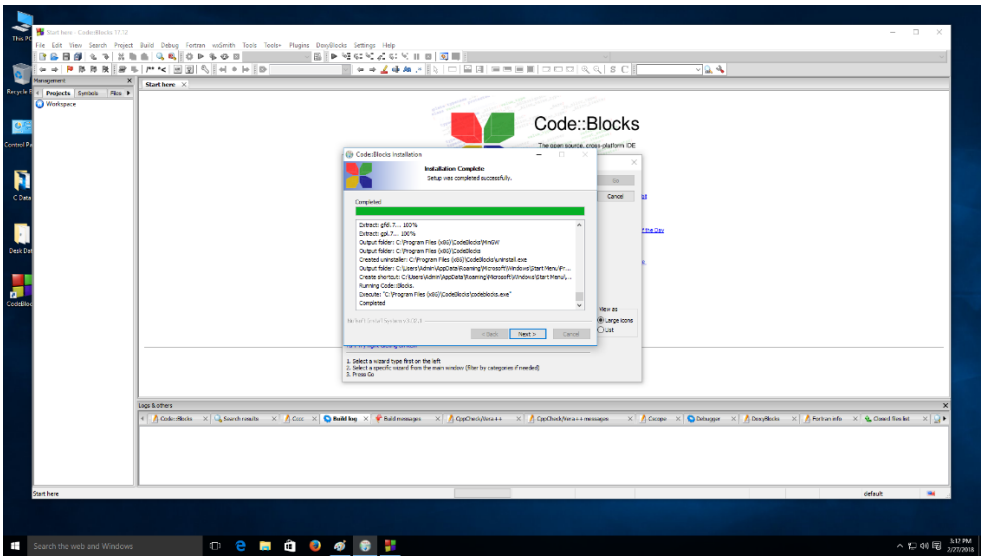
Here you can specify the path for your projects to any directory or you can just simply create a new folder there & save your project . I have selected the folder called " C Data ", which is present on desktop . So after setting folder click on next & then finish as shown below .

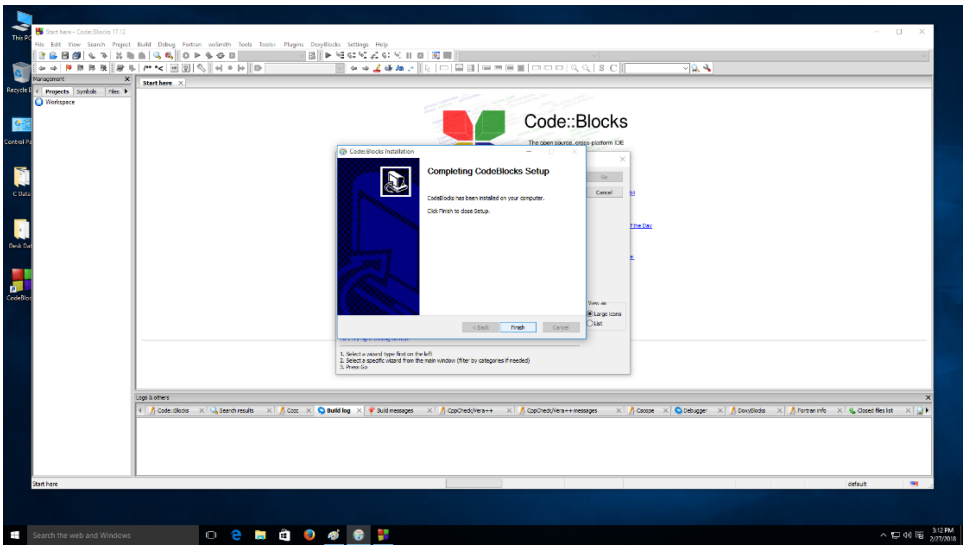


Now after clicking on finish , you will get a window , here you need to click on your project on left side as show in image below & you will get an editor at right side where you can write down your programs . Now our environment is ready here to learn c programming language .



You might also get following window during installation so just click Next & Finish as shown below .





You can also install & use the same compiler on Linux & Mac os machines as well .

Following are the youtube links for Linux & Mac users on " How to install codeblocks on Ubuntu Linux & Mac Machines ". So just copy & pase these links in your browser and follow the instructions .

Please follow the instructions as guided in the video .

1] Installing codeblocks on Ubuntu Linux Machine :- >>

https://www.youtube.com/watch?v=xoK5JCtNH5A

2] Installing codeblocks on mac machine :- >>

https://www.youtube.com/watch?v=_bKLttPVoC8

Basic Structure of a C Program .

In this part we gonna talk about the Basic structure of c program . Structure means how a c program can be constructed or In other words you can say the rules for constructing a c program.

So let's understand this with the help of following program .

```
/* Program to print Hello world Message to computer screen
    Author :- Prashant Shinde */
        # include < stdio.h >
        int main()
        {
        system ( " cls " ) ;
        printf ( " Hello World ! " ) ;
        return 0;
        }
```

Above is a basic c program which will print a message " Hello World ! " to the computers screen . So now let's understand how it is structured . A C Program is nothing but collection of preprocessor directives , Header files , variables ,keywords , different types of functions or in other words you can say a set of instructions . Now let's understand different sections of a c program . Means you can construct a c program in same flow or order .

1] Documentation section :- >> At the top of the program you can see there is text written in between /* & */ . This section is called as documentation section . Where you need to write down the purpose of your program . Suppose you gonna write a program about doing the addition of two numbers then you can simply write the comment in between /* */ as follows

/* C Program to print addition of two numbers */. This part is not compulsory in c programming . Means you can compile & run your c program successfully without writing this part . This thing which has been written in between /* */ is Called as " Comment ". You can write the comments anywhere inside the c program . The purpose of comment is to just to know the purpose of code for others . Means suppose you have written a two thousand lines of code then at such situation it is very important for you to write down the purpose of your code . If you give this code to any other programmer then it will be really easy for him to understand the purpose of your code .

2] Link Section :- >> In above code you can see there is one line of code written after documentation section i.e. #include<stdio.h> . In this line #include<> is a preprocessor directive which is used to link different functions used in your program with corresponding header file . So "stdio.h" is a header file . So what is function & header file , we gonna talk about it later as move forward in this book. But for now just understand that the definitions of functions those we gonna in your code are present in these included header files . There are different header files , which support different functions . So you need to include those header files according to your needs . Suppose you need to perform graphical operations in c then you can simply add #include<graphics.h> in you code & use the corresponding functions . This section is really important part of any c program . Because you can't compile your code successfully without linking the fuctions used in your code to corresponding header files .So Basically the job of this section is to link all the functions used in your program to it's corresponding header files .

3] Definition section :- >> The third section is definition section . There is a preprocessor directive named #define , with the help of this directive you can define symbolic constants . In above code we didn't used any of such code . But if you want to use it , then you need to write it down just exact below the preprocessor directives . Following is a Basic example of #define preprocessor directive .

Example :- >> # define PLUS +

Means whenever you use "PLUS" word in you code , it will be get treated as + symbol . So for now just understand that this section is used to write down symbolic constants . This part is not compulsory , if you want to skip writing down symbolic constants then you can simply skip them .

4] Global Declaration Section :- >> The blank part after Definition section is called as " Global declaration section " . In this part you can declare global variables . What are global variables . Global variables are those which can be accessed anywhere throughout our program . What are variables , global variables , we gonna talk about it as we move forward in this ebook , for now just understand this part is reserved to declare global variables .

5] main ( ) function section :- >> This is also a really important section in c programming . without this you can't write any c program . Means whatever code you gonna write in your program can only be written within this main function . This main() function is self invoked function . You don't need to include any special header file for this . In above program you can see we have used the function as follows .

```
int main ( )
{
system( " cls " ) ;
printf( " Hello World ! " ) ;
}
```

This section can also be subdivided into two parts i.e. declaration section & executable section .

The declaration section is a part which starts just below the opening curly brace { of main function . In this part you can declare the variables that you gonna use in your program . Then after this there is a executable part . In executable part you can write down any type of functions or expressions according your program needs .

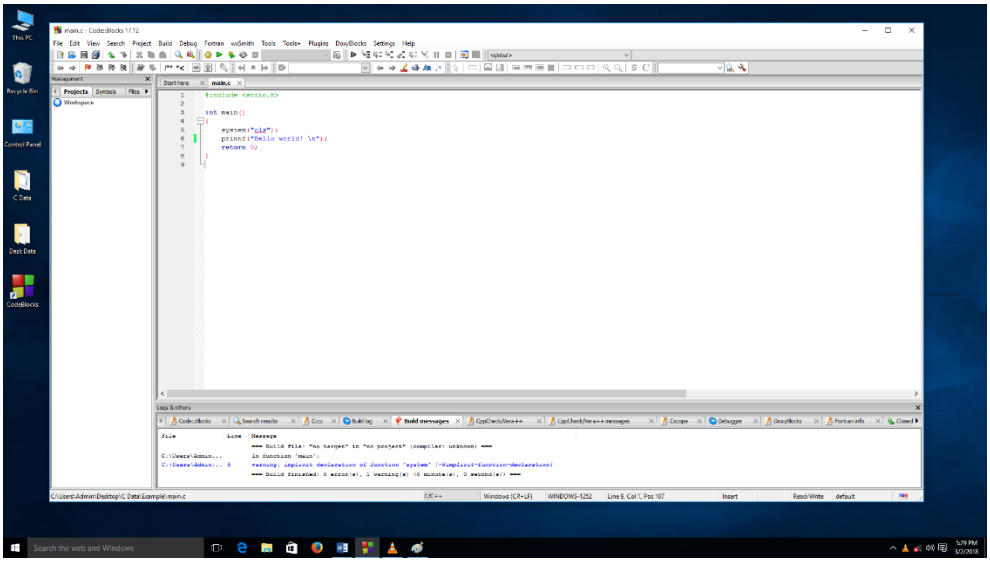
6] Sub-program Section :- >> This is the last section of a c program . we didn't have used this part in above code . This part is reserved for writing down your own function definitions

. We will learn how to write your own function as we move forward in this ebook . For now just understand this part is reserved to write down function definitions .

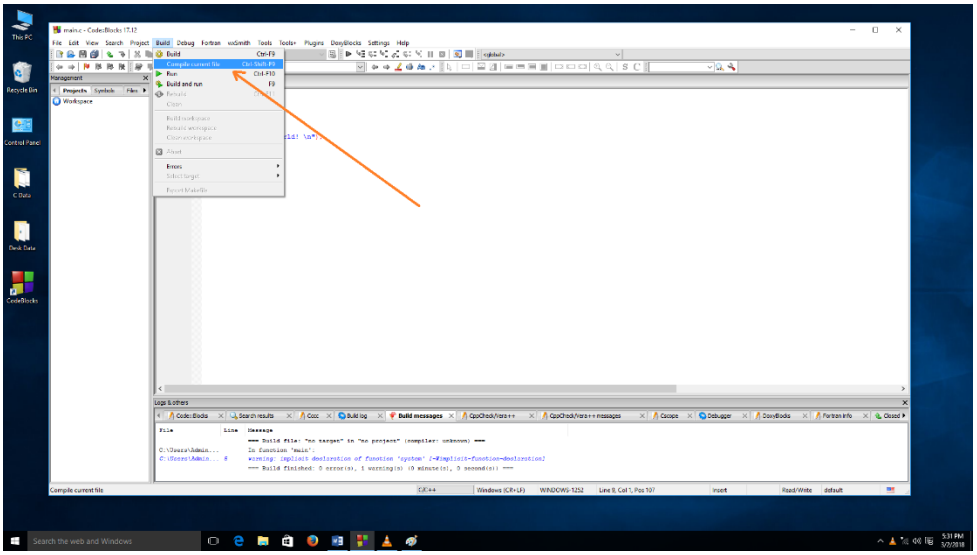
Compile & Execute your first C Program .

Now it's time to compile & execute your first c program . Compiling & executing is the procedure to convert the human understandale code in to machine language & make it executable .

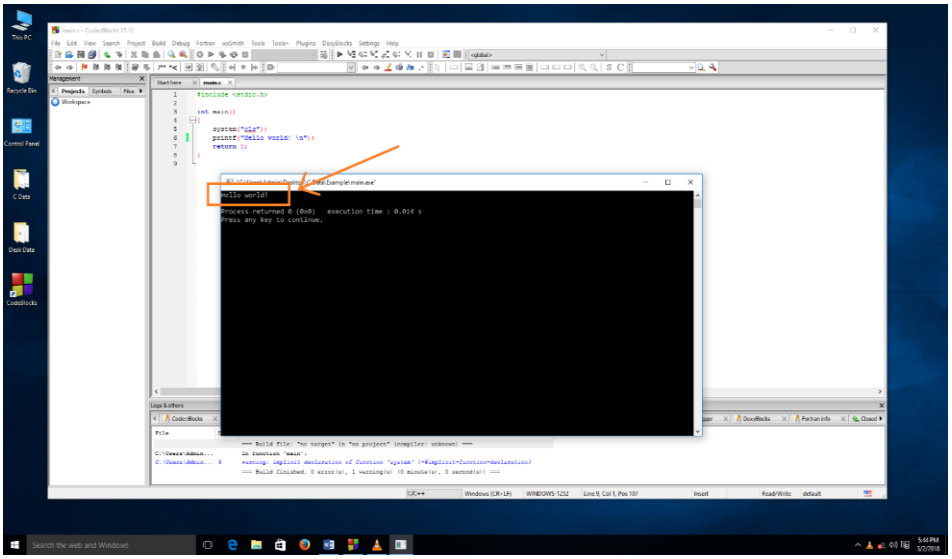
So just open up your codeblocks compiler & write down the above code as shown in the image below .



Now it's time to compile & run . So to compile just go to Build option and click on Compile current file , please see the attached image below for refrence .



If is there any error in your program then you will see those errors & warning messages at the bottom of compiler in Build messages window . There are different types of errors are there in c program . So if any error occurs then you need to solve it first & then your code will get compiled successfully . So after compilation procedure the next step is to run our code so at the bottom of " Compile current file "option you can see there is a option called " Run " . So by simply clicking on this option you can view the output of your code . So to run this just press Run & you will see a simple output as shown in below image .



So now Let's understand how this c program works .

At first we have included the header file with the help of preprocessor directive i.e. #include < stdio.h > . We have included this header file because the definition of printf() function is written inside this header file. The job of printf() function is to print what ever message we write inside it in between double quotes " " to output screen . So when ever you want to use printf() function then what you need to do is you need to include the stdio.h header file . Anewhat ever message you want to print to output you need to include that message within double quotes " " inside printf() .

The another function we have used is system() , with the help of this function we can execute any system command or dos command within our c program . So what ever command you wanting to use inside your code , you just need to add it to the double quotes . So in above program we have used the "cls" command to clear the output screen . so why we need to do that because suppose we have executed any previous program then the output of that program could be there so to print the new output we need to clear the screen . so that's why we have used the system("cls"); function .

Then the third line of code we have used is return 0 . This is a return value to the main function. We need to return 0 to main function because the meaning of it is that we have successfully executed our program . So it is important everytime to provide" return 0 " to main function .

Ok now it's time to know how our c program is get compiled &executed .

So when we press compile current file option , then program control enters into main() function . The blinking cursor that you will be able to see is called as " program control " . The job of this program control is to compile the code one by one.

STEP 1 :- >> So first it enters into main function then at first line it will read system("cls"); function , so the job of system() function is to run system or dos commands . so it will execute cls command & clears the output screen .

STEP 2 :- >> After that it will get to printf ( ) ; function & then check for it's header file at the top . It checks whether it can find the definition of printf() inside above included file . So the program control goes above at #include < stdio.h > and see the definition of printf() inside "stdio.h" header file . It's because it contains the definitions of printf() function . after that the parameters of printf() functions will get passed to the definition and we will see the output to computer screen . The job of printf() is to print any message to computer screen . So that's why we can see " Hello World ! " to the output .

STEP 3 :- >> At last step it will return 0 to main function & we will get out of the program or executable file .

This how any c program is get executed step by step . Following is a basic assignment which will help you to understand how it works , so please practice it as guided with above program .

ASSIGNMENT :- >> Write down your first C Program to print your full name on computer screen ?

KEY POINTS :->>

- C programming is best way to get into programming world .
- You can install codeblocks on any operating system like windows , mac or linux .
- A C Program is nothing but set of instructions & each instruction or statement in C ends with semicolon ; .
- Execution of every C program always start with main ( ) function .
- In Codeblocks Control + F9 is the shortcut key to compile your current code & to execute your code you need to press shortcut key Control + F10 .

2. VARIABLES CONSTANTS DATA TYPES & KEYWORDS IN C .

What is variable ?

A variable is an entity whose value can be changed . In other words , a variable is a reserved space inside computers memory or RAM to store the data .

We can store the data inside computers memory with the help of variables and we can declare the variables with the help of data types . There are different types of variables are there in c .

Let's understand the variable with the help of an examples .

Example :- >> `int a ;`

So above is an example of integer variable . Here int is a data type where as "a" is variable name . so when you declare the integer data type then you can store the decimal values to it .

You can change the value of the variable when ever you want inside your program .

What is constant ?

Constant in an entity whose value can not be changed during program execution . In other words you can say constant is an assigned value to a variable during program execution .

Example :- >> `int a = 15 ;`

In above example I have assigned a value 15 to variable" a", so when you print the value to screen , it will get printed as 15 , this can not be changed until someone interact with the variable . so that's why it is called as " constant ".

Data Types in C .

Now it's time to understand the data types in c . There are many data types are available in C . Each data type is having it's own purpose , for example integer data type is used to store only integer values , float data type is used to store floating point values & so on . When ever you declare any type of data type then it will acquire a space inside computers memory . For

example when you declare integer data type then it will acquire 2 bytes in computers memory . As like memory size , it also has a particular range . Example when you declare integer data type then it can store the values in between a particular range i.e -32768 to +32767 . Means you can not store 34,000 value for an integer data type . So you can use or declare the data type inside your program according your needs .

Following is a common list of data types available in c .

Data Type	Size in Bytes	Range .
int	2 bytes	-32768 to 32767 .
char	1 byte	-128 to 127 .
float	4 bytes	3.4x10-38 to 3.4x10 38
unsigned int	2 bytes	0 to 65535
long int	4 bytes	-2147483648 to 2147483647
double	8 bytes	2.22507e-308 to 1.79769e+308

So above is the list of common data types , their size and range available in c . To use these data types in your program you need to use the format specifiers . There are different types of format specifiers are their according to the data type .

Following is a list of format specifiers according to their data types .

Data Type	Format Specifier .
int	% d
char	% c
float	% f
unsigned int	% u
long int	% ld

double	% df

As like data types , there are also different types of constants based on these data types . which are as follows .

1] integer constant example :- >> int a = 12 ;

2] character constant example :- >> char a = 28 ;

3] Real / float constant example :- >> float a = 23.54 ;

What is keyword ?

Keyword is a special word whose meaning is already explained to the compiler to perform a specific task . There are totally 32 keywords are available in c . Each keyword is having it's own function . For example we have used " int " in above example to declare a variable ,though int is a data type but it is also a keyword in c .

.

Following is the list of 32 keywords available in c .

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

Declaring variables in c .

Until now we have learned the basics of variables , data types , keywords and constants in c , now it's time to learn how to declare a variables in c . So to declare variables whether

they are float char or int , you need to follow some rules . So Please checkout the list of following rules to declare the variables in c .

- A variable name is any combination of alphabet , digit or _ (underscore) .
- No comma or blank space is allowed within a variable name .
- No keyword or any special symbol other than underscore (_) is allowed within a variable name .

Following is the Basic syntax to declare a variable name.

Syntax :- >> Data Type variable name ;

Example :- >> int a , float marks , char c ;

Note :- Declaring variable names according to program requirement is good practice . Suppose you want to write a program to calculate the percentage of students then you can declare the variable as " float percentage " .

ASSIGNMENT :- >> Declare five different integer variables to store data .

KEY POINTS :- >>

- There are total 32 keywords are available in c programming language .
- Whenever you declare any variable in c then it reserves some space inside computers memory & it gets initialized to a garbage value .
- The size of a variable is always depends upon the data type & also on compiler for example :- int data type reservers 2 bytes in computers memory when you use it on Turboc++ 3.0 compiler & when you use it on codeblocks then it reserves 4 bytes .
- Each data type contains a specific format specifier to receive an input .
- You can not declare any keyword as variable name .
- A constant can be an integer constant , character constant or float constant .

3. OPERATORS & EXPRESSIONS IN C .

A C Program is nothing but a set of instructions , each instruction in c programming can be designed with the help of variable , keywords , operators and expressions . So here I would like to introduce you to different types of operators supported in c . So let's see them one by one .

1] Arithmatic Operators :- >> Following is the list of arithmetic operators which can be used to perform arithmatic operations in c .

Operator	Meaning
+	Addition .
-	Substraction .
/	Division .
*	Multiplication .
%	Modulus .

2] Relational Operators :- >> In programming when we need to compare some values to accomplish a specific task then we can use relational operators . following is the list of available relational operators in c .

Operator	Meaning
<	less than .
>	greater than .
<=	less than or equal to .
>=	greater than or equal to .

==	equality to .
!=	not equal to .

3] Logical Operators :- >> There are some conditions when we need to perform some logical operations in c , so to write such code logical operators are used .

Operators	Meaning
&&	And Operator .
||	Or Operator .
!	Not Operator .

4] Assignment Operator :- >> The assignment operator is used to assign the right side expression to left side . it is denoted by " = " .

Example :- >> a = 10 ;

5] Unary Operator :- >> There are two unary operators in c .

a] Increament Operator :- >> It is denoted by " ++ " . The increament operator is used to increase the value of an variable by 1 .

b] Decreament Operator :- >> It is denoted by " - - " . The decreament operator is used to decrease the value of an variable by 1 .

6] Conditional Operator :- >> It is denoted by " ? : " . It can be used in some conditions when you want to check whether a condition is true or false .

What is Expression ?

Expression means any legal combination of symbols that represents a value . In c programming there are different types of expression are there like infix ,prefix and postfix . In programming any expression can be build with the help of operators & operand .

Example :- >> Suppose you want to perform addition of two numbers in c then you can simply write an expression as follows .

```
X = Y + Z ;
```

So in above expression y and z are operands and " + " is an operator. In above expression the operator is present in between two operands , so such expressions are called as " Infix Expressions " . There could be many examples of it like substraction of two numbers , division of two numbers , or addition of five different variables .

There are some expressions where the operator is written before operand such expressions are called as " Prefix Expression " .

Example :- >> ++a , ++z ;

There are some expressions in which the operator is written after operand , such expressions are called as "Postfix Expressions ".

Example :- >> a++, z++;

KEY POINTS :- >>

- Basically there are about 6 types of operators are available in c .
- Each operators is used to perform a specific operation , like arithmetic operators are used to perform arithmatic operations , logical operators are used to perform logical operations & so on .
- Expression is nothing but a combination of operator and operands .

4. PERFORMING INPUT & OUTPUT IN C .

Until now we have learned how to print any message to computer screen with the help of printf() function .Now it's time to learn how to receive values from keyboard and then display them to the output with the help of printf() and scanf() function .

So to understand this please observe the following program carefully .

```
/*   Write a c program to receive two values from keyboard and print their addition to
screen .

   Author :- Prashant Shinde    */

#include<stdio.h>

int main ()

{
int a , b , c ;

system ("cls") ;

printf (" Enter values for a & b respectively") ;

scanf (" %d %d ", & a , & b ) ;

c = a + b ;

printf(" The addition of a & b is %d ", c ) ;

return 0 ;

}
```

Now let's understand how this program works ,

Step 1 :- >> At the beginning we have declared 3 variables , a,b,c . variable a & b are declared for receiving 2 values sequentially . variable c is declared to save the addition of variables a & b and then print it to the computer screen .

Step 2 :- >> Then cleared the screen with system() ; function .

Step 3 :- >> Then at next step we have displayed the message to screen , which will help you to take next step .

Step 4 :- >> After the message , now here is our main function to receive the values . So let's understand how scanf() function works , via scanf() function we need to pass two parameters , the first parameter is format specifier & that format specifier must be enclosed within double quote . After format specifier we need to pass variable names at right side of scanf() function . The another thing is you must add " & " symbol before the variable name in scanf() function . In our program we have 2 variables and those are integer variables that's why we have set two format specifiers inside the double quotes as " %d %d " and we have passed the variable names as " &a , &b " separated by comma . If we had 3 variables then we have been set the scanf() function as follows scanf(" %d %d %d ",&a , &b , &c) . You must add comma operator to separate the format specifiers & variable names . The meaning of " & " in c programming is value at address . so we gonna store values for " a & b " received from keyboard at memory locations of variable a & b respectively .

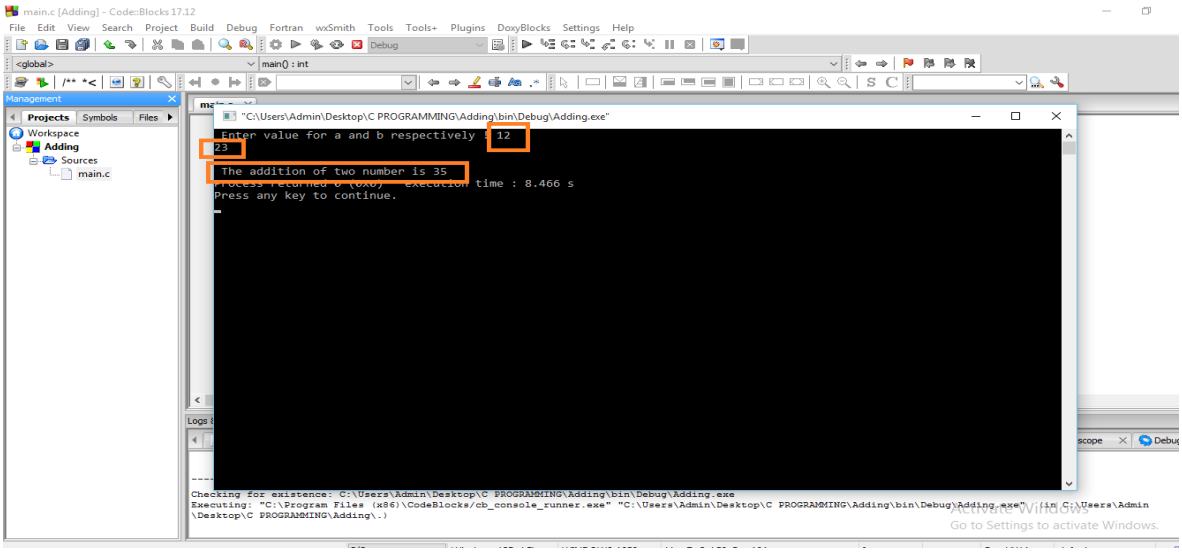
Step 5 :- >> After receiving the values , at next line we have added them with an expression c=a+b ; .

This will add the values of variables a & b and then store the result to variable c .

Step 6 :- >> and finally we have used the printf() function to print that result to computer screen. This is another way of using the printf() function . You can print the value of an variable to computer screen with the help of this format . You just need to pass %d to printf() function and the variable name to right side as shown in the program above . You must use comma operator here to separate the message & variable within printf() function .

Now it's time to compile & run the program . So when you compile and run the program it will display you the message on screen to enter the values for and b respectively . You need to enter those values one by one . Just hit enter sequentially after entering the values and finally it will display the addition of two numbers respectively .

Following is the screenshot where I have added two values 12 & 23 respectively and finally we got the result 35 .



Now it's time to understand the same input concept with another example , following is the program where we gonna calculate the percentage of students after entering total marks obtained by the students . Percentage of student is calculated by dividing by total marks obtained with total number of subjects . So Let's observe the following example carefully.

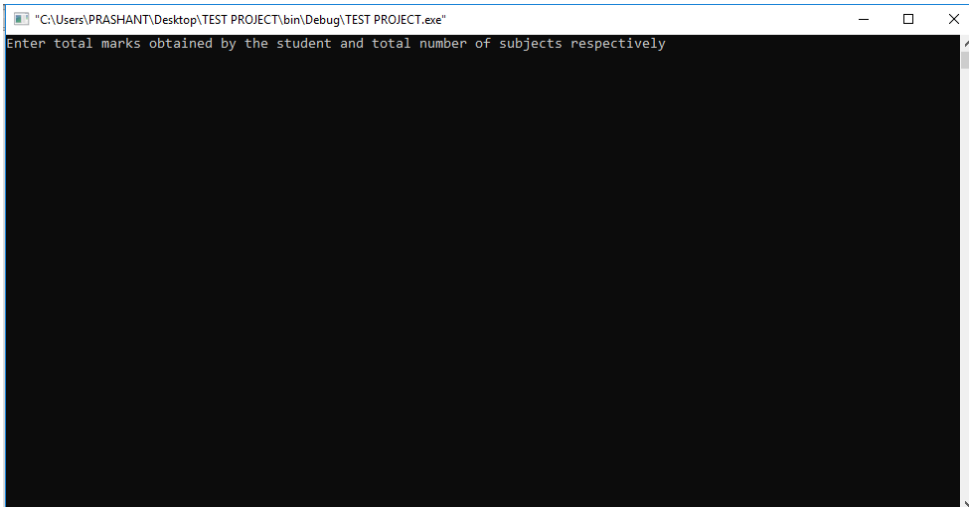
```
/*  Write a c program to calculate percentage of student .

    Author :- Prashant Shinde  */

#include<stdio.h>

#include<conio.h>

int main()

{

/* t_marks is for receiving total marks , sub is for receiving total no. of subjects and per for storing the result and showing it to output . We have declared "per" as float variable because the result could be in floating points  */

int t_marks , sub;

float per;

system ( " cls " );
```

```
printf( "Enter total  marks obtained by student & total number of subjects respectively !") ;

scanf(" %d %d ", &t_marks , &sub ) ;  /*  Here we are receiving total marks & total subjects  */

per = t_marks / sub  ;   /* This line of code is to calculate the percentage of students  */

printf( " The percentage of student is %f",per) ; /* Here we are displaying the result to output  */

return 0;

}
```

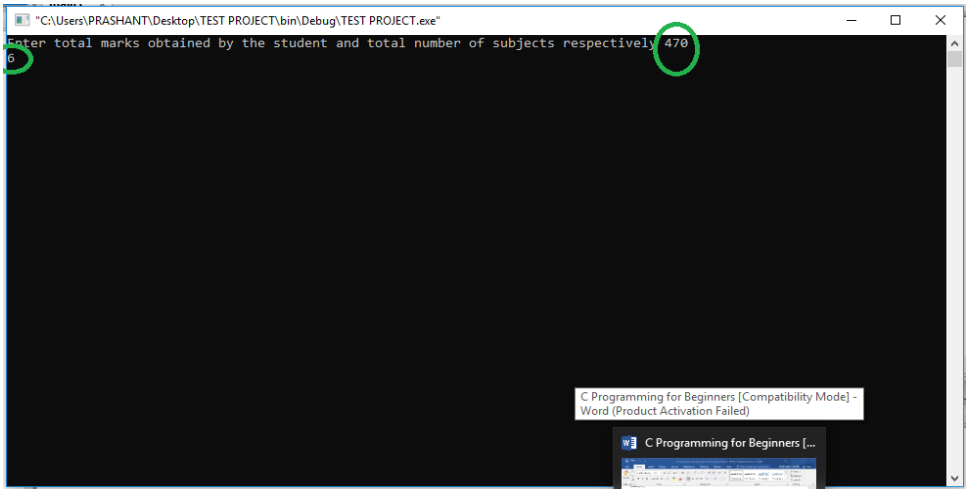
Note : - Please do not directly copy and paste the source code shown in this ebook , it will not work for some reasons . But if you type the same code on your own then it will work definitely .

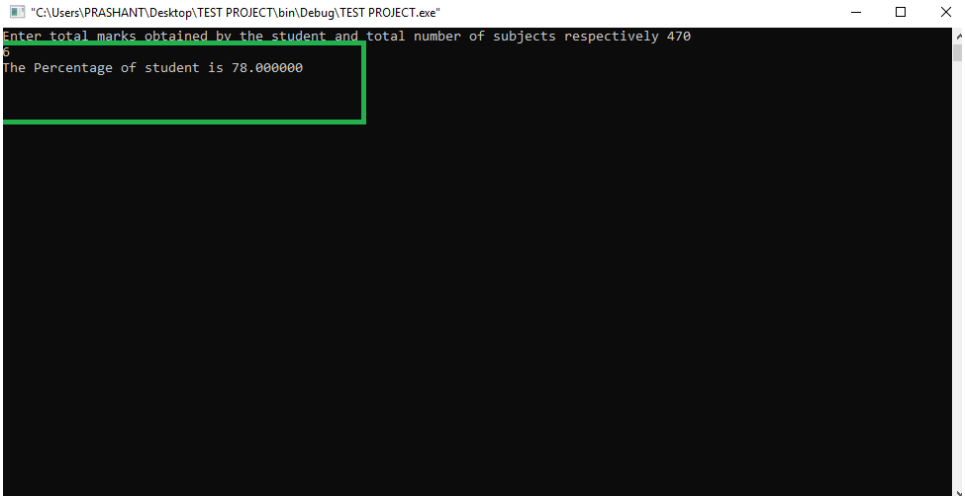
So here by reading the comments , you can easily understand how this program works .

When you execute the code you will see the output as shown in the image below . It will ask you to enter total marks obtained and total number of subjects .



When you enter the total marks obtained and total number of subjects then it will calculate the percentage at the background and show it to you as below in the following output . Entered values are shown with the help of green circles and after that next window shows the calculated result .





ASSIGNMENT :- >> Write a c program to calculate area of square (Area of square=side*side) .

Formatting the output :- >> After understanding how to perform input and output operation in c , now let's understand how to format the output . Formatting the output will make our output looks more cool . So to format the output there are some escape sequence characters introduced in c with the help of which we can format the output in c . Following is the list & use of escape sequence characters supported in c language .

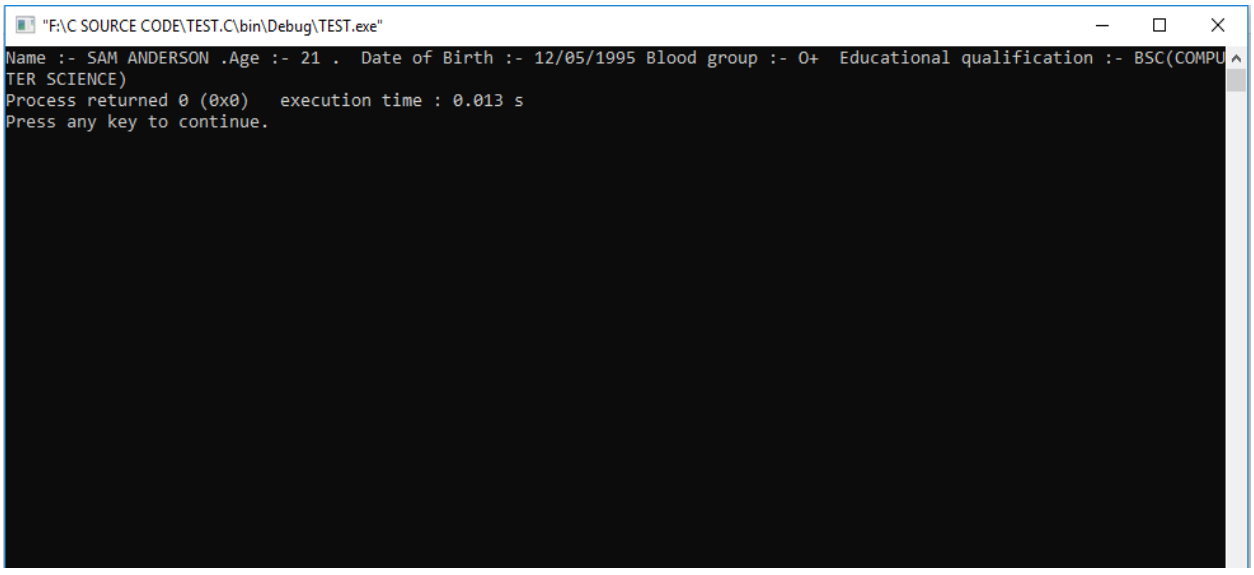
E.S. Character.	Use .
\n	Prints your text to the new line .
\b	Works as back slash . Means deletes the last word in the text where you have used .
\t	Print tab . (Means prints your text with a difference of one tab) .
\r	Return to the first column of current line .
\a	Rings alert bell to the output .
\'	Prints single quote .
\"	Prints double quote .
\f	To form feed.
\\	Prints backslash .
\	Prints escape .

All the above escape sequence characters are generally works with printf() function successfully . So let's see how to implement few of the E.S.Characters practically . Lets print some information about SAM ANDERSON without the use of E.S.Characters . The the information would be his name , age , date of birth , blood group , educational qualification .

```
/*  Write  a c program to print name , age , date of birth , blood group & educational
qualification  of SAM ANDERSON  without the use of any escape sequence character .

    Author :- Prashant Shinde    */
```

```
#include < stdio.h >

#include < conio.h >

#include  < stdlib.h >


int main( )

{

 printf ( " Name :- SAM ANDERSON ." ) ;

 printf ( "Age :- 21 . " ) ;

 printf ( " Date of Birth :- 12/05/1995  " ) ;

 printf ( " Blood group :- O+  " ) ;

 printf ( " Educational qualification :- BSC(COMPUTER SCIENCE) " ) ;

  return 0 ;

}
```

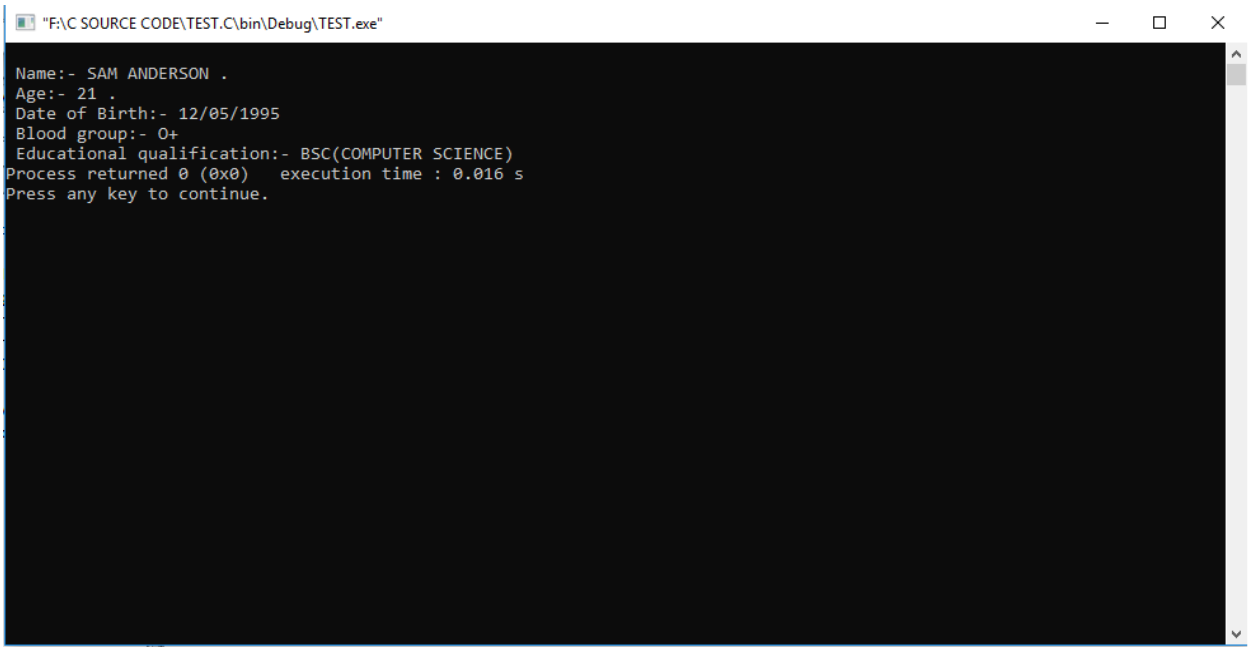
Following is the output of above program so please observe how it looks without use of any escape sequence character . The output looks very ugly because it is printed in one line , so the reader cant observe it clearly .



Now let's write the same program by adding " \n " escape sequence character in printf ( ) function .

```
/*  Write  a C Program to print name , age , date of birth , blood group & educational
qualification  of SAM ANDERSON  By using " \n " escape sequence character .

     Author :-  Prashant Shinde    */

#include < stdio.h >

#include < conio.h >

#include  < stdlib.h >


int main( )

{

 printf ( " \n Name :- SAM ANDERSON ." ) ;

 printf ( " \n Age :- 21 . " ) ;

 printf ( " \n Date of Birth :- 12/05/1995  " ) ;

 printf ( " \n Blood group :- O+  " ) ;

 printf ( " \n Educational qualification  :-  BSC ( COMPUTER SCIENCE ) " ) ;

  return 0 ;

}
```

Now let's observe the following output after adding " \n " escape sequence character in printf() function .It looks so nice & well organized .

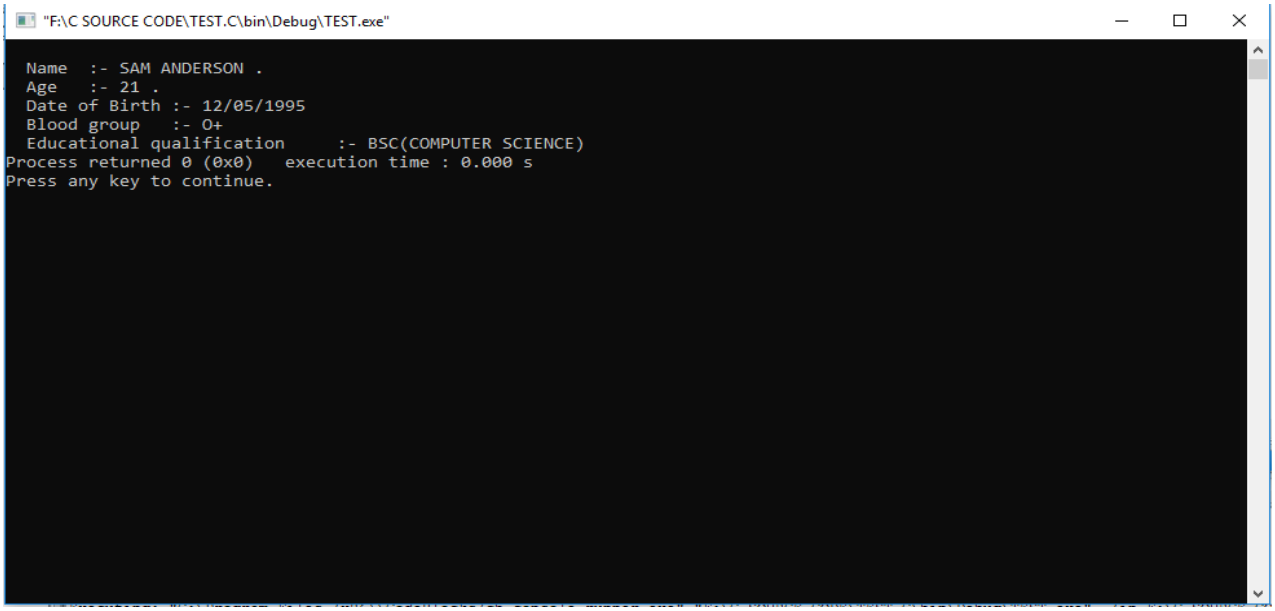


>> Now let's see how to use " \a " & " \t " escape sequence characters . When you add " \a " it will ring a alert bell in the output . It will ring it only one time , no matter how many times you use & the " \t " will add a space in the text in the output . So let's see the same program above after adding " \a " & " \t " .

```
/*  Write a c program to understand the working of  " \a " & " \t " .
    Author :- Prashant Shinde  */
 #include  < stdio.h >
#include < conio.h >
#include  < stdlib.h >

int main( )
{
 printf ( " \n \a Name\t:- SAM ANDERSON . " ) ;
```

```
 printf ( "\n \a Age\t:- 21 . " ) ;

 printf ( "\n \a Date of Birth\t:- 12/05/1995 " ) ;

 printf ( "\n \a Blood group\t:- O+  " ) ;

 printf ( "\n \a Educational qualification\t :- BSC( COMPUTER SCIENCE ) " ) ;

  return 0 ;

}
```

Following output will help you to understand how it works . After adding " \ t " you can see a space in between Name & :- . You can observe a space wherever we used " \t " . You can use these escape sequence characters anywhere in your program with print() function .



ASSIGNMENT :- >> Modify the above program & insert your address & print it to the output screen .

KEY POINTS :- >>

- To receive input from keyboard scanf ( ) function is generally used in c language , there are many functions out there but scanf ( ) is most common and easy to understand function .
- To print any message on computer screen there are many functions are available in c language but printf ( ) is a commonly used and easy to understand function in C .
- Printf ( ) function is not only used to print the message on computer screen but you can also print the value of an variable or memory address of an variable .
- Escape sequence characters helps you to format your output so that it looks clean & simple .

5. DECISION MAKING IN C .

In our normal life we make decisions on the basis of our knowledge and understanding , our likes and dislikes . In the same way we can implement some code in c which will make computer to take decisions on it's own . There is feature called as "statements" , introduced in c which helps computers to take decisions . There are different statements in c . Each statement has it's own benefits and purpose . So let's get started with if statement .

i] if statement :- >> if statement is declared with the help of keyword " if " followed by opening and closing parenthesis ( ) . what ever condition(expression) you put inside the opening and closing parenthesis that condition will get check and if the condition becomes true then it will execute the code within if block and if condition becomes false then all the code within if block will be excluded .following is the basic syntax to declare if statement .

Syntax :- >>

```
if ( condition )

code to be executed;

OR

if ( condition )

{

code to be executed ;

}
```

Above is the basic syntax to declare if statement . if you want to execute a single line of code after a particular condition become true then you can you the " if statement " without use of opening and closing curly brackets { } and when you have multiple lines of code and you wanting to execute that code after a particular condition becomes true then you can use the opening and closing curly brackets { } after setting if condition and write that code inside those opening and closing curly brackets { } as shown in above syntax . The condition within the if block can be prepared with the combination of variables & operators .

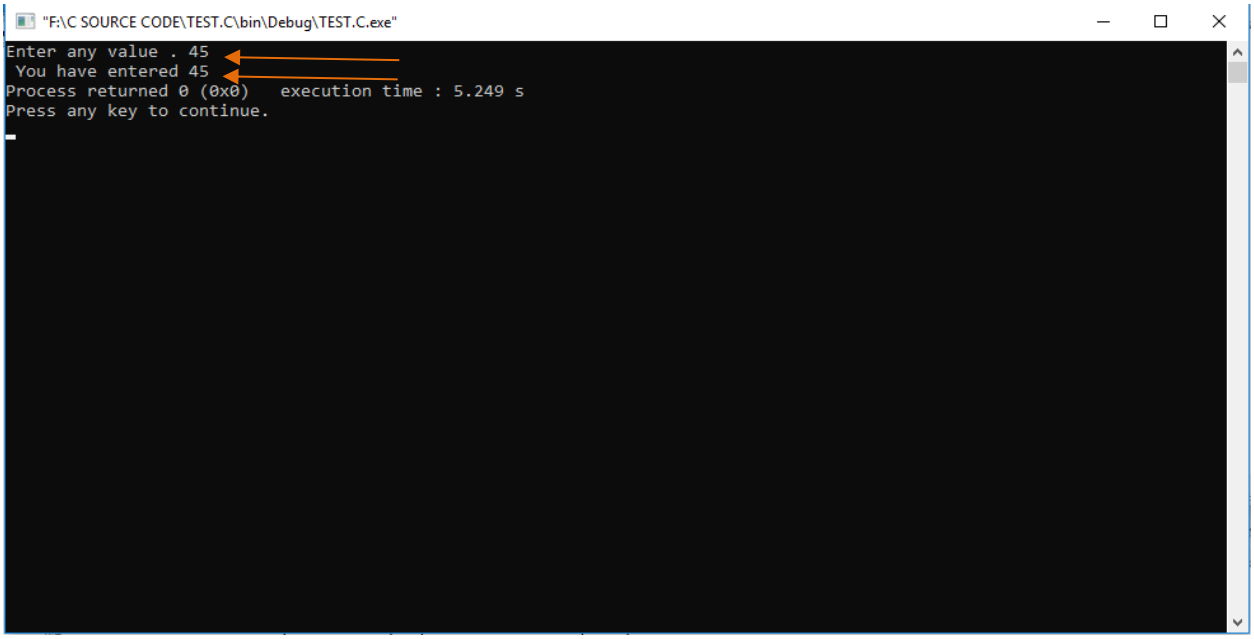
Now let's see how if statement can be implemented practically . Let's write a c program which will receive a value from keyboard and display it to the screen only when

the value is less than 50 . So let's see how to implement the logic for it with the help of if statement .

```
/*  Write a c program to implement if statement

   Author :-  Prashant Shinde  */

#include < stdio.h >

#include < conio.h >

#include < stdlib.h >

int main ()

{

 int value  ;  /* This variable is to receive the value from keyboard */

system( " cls " ) ;

printf( " Enter any value  " ) ;

scanf( " %d " , & value ) ;

if ( value < 50 )  /* This is a condition which has been set to check the entered value*/

printf ( " The value entered is %d " , value ) ;

getch ( ) ;

}
```

Now let's understand how the logic of if statement works . First of all the program control enters into main function & creates an integer variable value . After that clears computer screen due to system(" cls ") ; function .After that it will show a message on computer screen that " Enter any value " . Then next we will enter any value from keyboard . now our entered value will be stored to variable " value " . Now at next line our main code of if statement will get execute . In this if statement we have set a condition to check the entered value is less than 50 or not . we have used variable "value" & less than " < " operator to set this condition . so here if the condition become true then it will show the message on screen as " The entered value is " along with entered value . Otherwise it will skip this part and exit the code . This is how the if statement works .

Following is the screenshot which is displaying the value entered . The arrows are showing the value entered via keyboard and it's result to output .



Now let's modify the same program to understand how multiple lines of code can be set and executed inside the if block with the help of opening and closing curly brackets { } .

Suppose now I want to display a message along with entered value to the computer screen . Then simply implement the opening and closing curly brackets { } below if block and put you code inside of it .

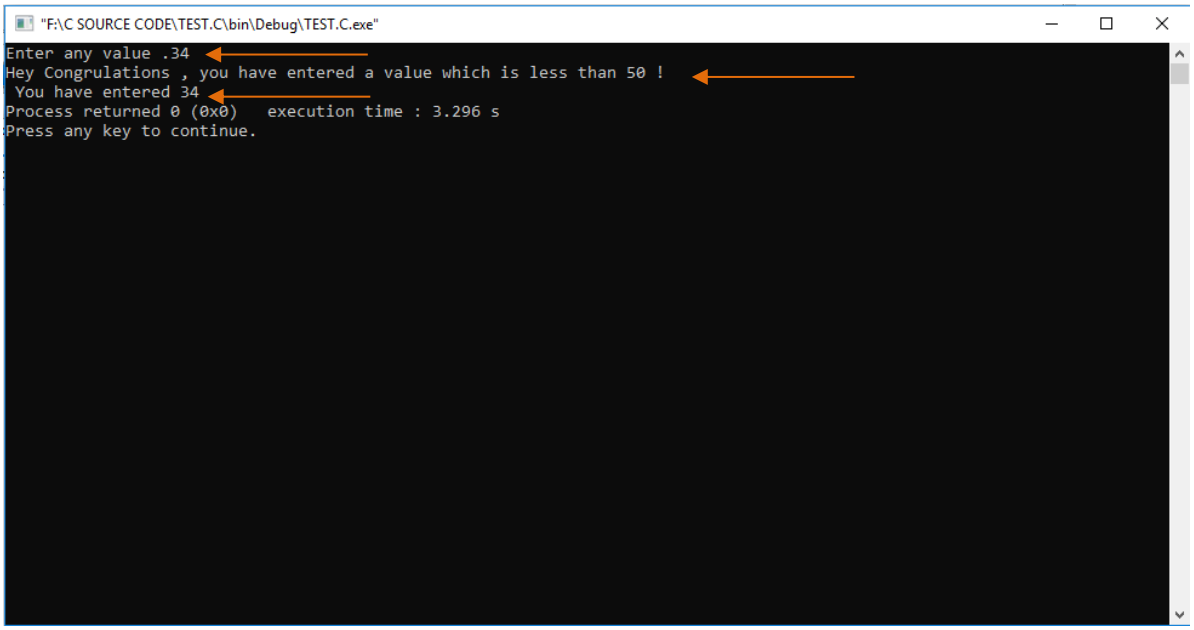
/* Write a c program to determine whether the entered value is less than 50 or not with the help of opening & closing curly brackets .

Author :- Prashant Shinde */

```
#include < stdio.h >

#include < conio.h >

#include < stdlib.h >

int main ()
```

```
{
 int value ; /* This variable is to receive the value from keyboard */
system ( " cls " ) ;
printf ( " Enter any value " ) ;
scanf ( " %d " , & value ) ;
if ( value < 50 )   /* This is a condition which has been set to check the entered value */
{   /* This is a opening curly bracket of if block */
printf( " Hey  Congrulations , you have entered a value which is less than 50 ! " ) ;
printf( " The value entered is %d " , value ) ;
}   /* This is a closing curly bracket to close the if block */
getch ( ) ;
}
```

Following is the screenshot which is displaying the value entered and the message inside the if block along with entered value .



ASSIGNMENT :- >> Write a C Program to receive the value greater than 50 with the help of if statement .

ii] If else statement :- >> After learning if statement now it's time to learn if else statement . If else statement is implemented with the help of keywords if and else . In if else statement the " if " always contains opening and closing parenthesis () but else don't require any parenthesis . There are some situations in programming where you need to implement such code in such a way that if the condition of " if " block fails then you need to execute the code within else block . The else block get executed only when the condition of " if " block fails . Following is the basic syntax to set if else statement .

Syntax :- >>

```
if ( condition )
code to  execute ;
else
code to execute ;
Or
if ( condition )
{
code to execute ;
}
else
{
code to execute ;
}
```

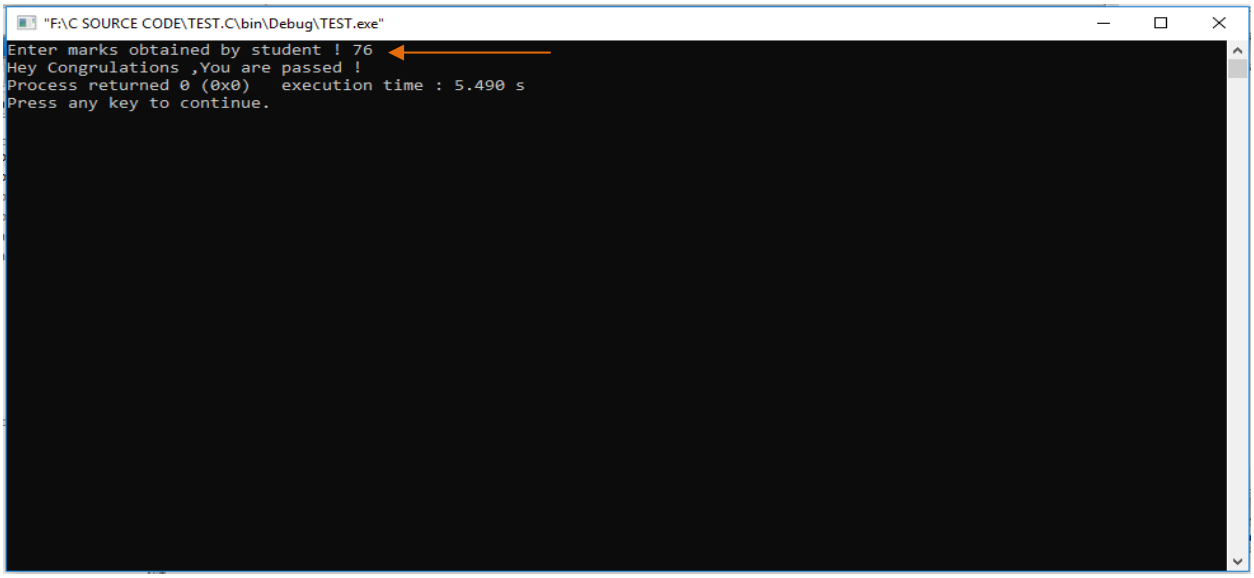
Now let's see how if else statement can be implemented practically . Let's write a c program which will determine whether a particular student is passed or failed ,after entering marks via keyboard . So the condition for this program is that when a student gets marks below 40 then he is failed and if he contains marks above 40 then he is passed .

/* Write a C Program to check whether the student is passed or failed with the help of if else statement .

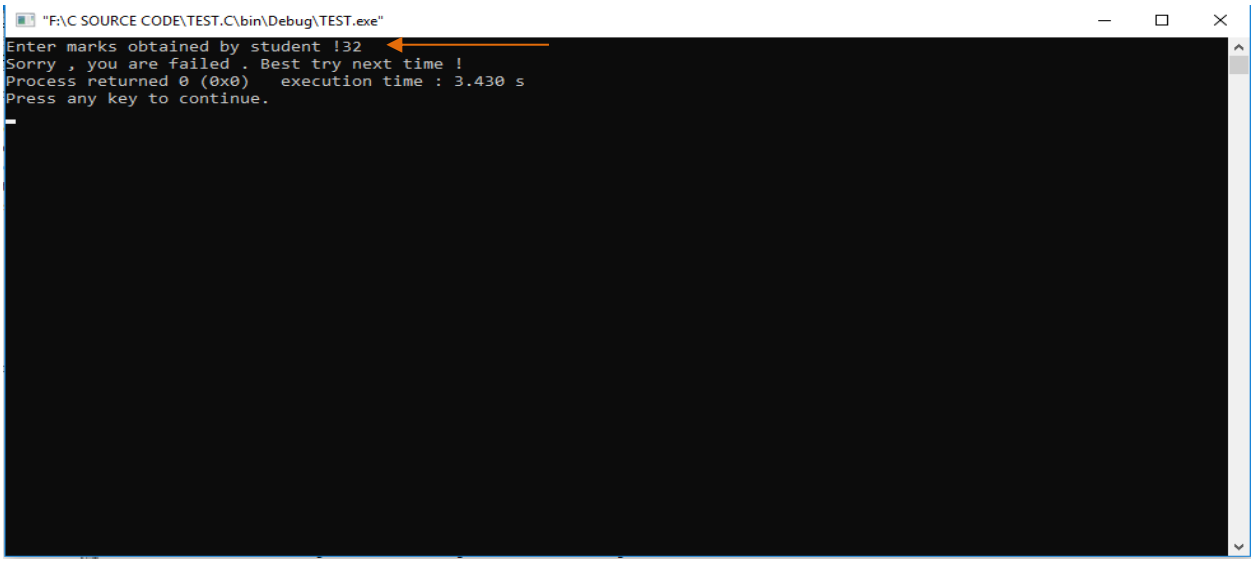
Author :- Prashant Shinde */

```
#include < stdio.h >

#include < conio.h >

#include < stdlib.h >


int main ()

{

   int marks ;

   system ( " cls " ) ;

  printf ( " Enter marks obtained by student ! " ) ;

  scanf ( " %d " , & marks ) ;

  if ( marks >= 40 )

  {

   printf( " Hey Congrulations , You are passed ! " ) ;

  }

  else {

     printf( " Sorry , you are failed .Best try next time ! " ) ;

  }

  return 0;

}
```

Following is the output shown when 76 percentage is entered via keyboard . Because 76 is greater than 76 so that's why the code within if { } block is get executed .



Now following is the output when I entered 32 which is less than 40 . Because 32 is less than 40 so condition of if {} block get failed and code inside the else { } block is get executed .



So now I think you have got the basic understanding of if else statement and it's working . The code inside the else block is get executed only when the condition of if block fails .

ASSIGNMENT :- >> Modify the above code to show the student is passed only when he will have marks above 30 otherwise he is failed .

iii] Switch statement :- >> Switch statement is the next level of decision making introduced in c . In switch statement you can create different branches to perform different operations . In other words you can write menu driven multiple choice programs with the help of switch statement in c . A switch statement is consisting of four keywords & those are " switch ,case ,break , default " . following is the basic syntax to declare switch statement .

Syntax :- >>

```
switch ( condition / variable/ value )
{              /* start of switch statement */
case constant :
        code to execute ;
            break ;
case constant  :
            code to execute ;
                break ;
default :
            code to execute ;
                break;
}       /* end of switch statement*/
```

keyword case

case constant with colon :

Now let's see how switch statement works , first we need to set up a condition(expression) inside the switch statement .We can pass a variable or expression or a constant value via switch statement . Suppose we have passed constant value 2 via switch statement then that value is get tested with each case constant present inside the

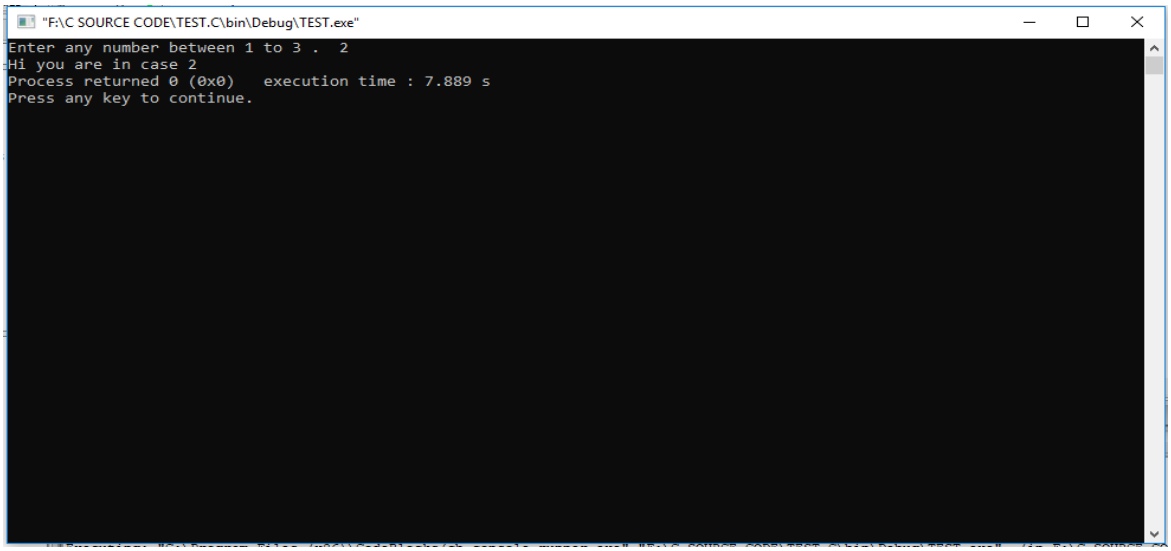
switch block until it matches with the same value and when it founds the same value then it executes the code within that case constant . After executing the code due to break keyword the program control comes out of that switch block .If the value doesn't matches with any of the case constants which are present inside the switch statement then the code within the default statement get executed by default .You can set an operator or character as case constant , but when you use them then you need to put them in single quote like ' + ' ,' * ' , 'A ' , ' B ' . Whenever you use single character or any operator as case constants then you need to use character variable to receive values from keyboard and then test them with each case constants present inside switch block .when you test case constants with character variable then they actually matches the ascii value of entered text from keyboard with case constant declared inside the single quote ' ' .

Now let's see how switch statement works practically , following is the basic example which will receive a value from keyboard . It will display the result according to your choice . You need to enter the value from 1 to 3 . It will display the result based on your choice . If you enter 1 then the code inside case constant 1 will get execute . If you enter 2 then code inside case constant 2 will get execute . If you enter any other value other than 1/2/3 then the code within default case will get execute . whenever you use values as case constant then you don't need to include single quote around it like this ' 1 ' . But when you use any operator or character as case constant then it is compulsory to use single quote around it and in this situation you need to use character variable to receive values from keyboard .

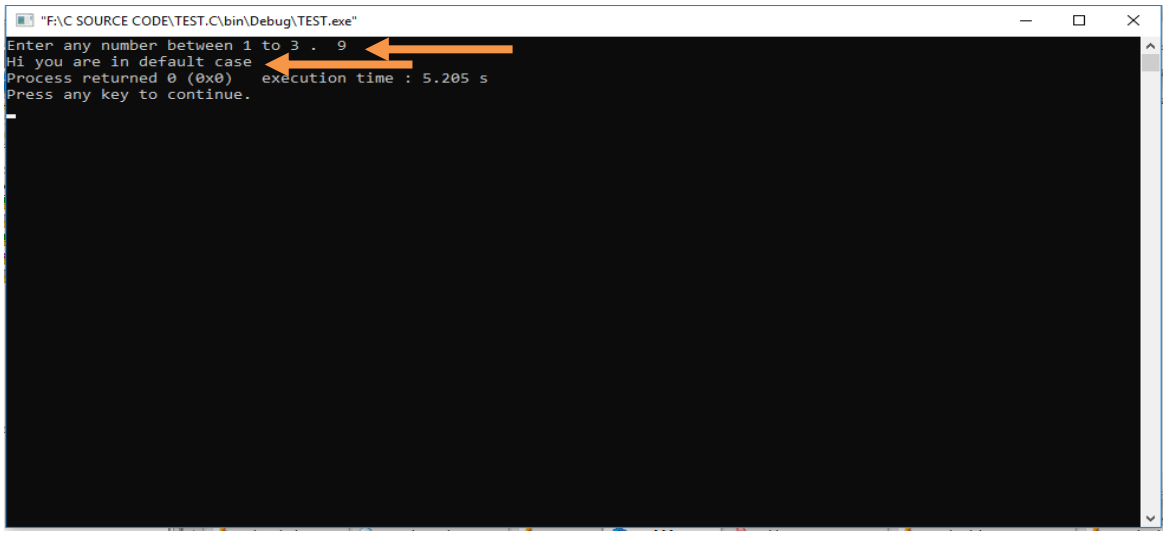
```
/*  Write a c program to display the working of switch statement .
    Author :- Prashant Shinde .  */
#include < stdio.h >
#include < conio.h >
#include  < stdlib.h >
int main ( )
{
  int  a ;
  system ( " cls " ) ;
  printf ( " Enter any number between 1 to 3 ." ) ;
```

```
 scanf ( " %d ", &a ) ;


 switch( a )
 {
case 1 :
    printf( " Hi you are in case 1 " ) ;
    break ;
case 2  :
   printf ( " Hi you are in case 2 " ) ;
    break ;
case 3 :
   printf ( " Hi you are in case 3 " ) ;
    break ;
default :
   printf( " Hi you are in default case " ) ;
   break ;


 }
 return 0 ;
}
```

So when you compile and execute the above program you will find the output as below . In the following output I have entered 2 so that's why it is showing output as " Hi you are in case 2 " . Means code inside the case constant 2 is executed here .



Following is the output when I entered 9 . Because 9 is a value which is other than declared case constants , that's why it is showing the code within default case . So whenever you enter any other value which is other than each case constant present inside the switch block then by default the code inside the default case is get executed .



Let's see another example with character constant and operators . In following example we have used [a] as character variable to receive a character from keyboard . In case first we have used alphabet ' A' . When someone enters 'A' from keyboard then by default the code inside the case constant 'A' will get execute . In case number 2 and 3 we have used

operators -,* . When someone enters these characters from keyboard then by default the code inside these case constants will get execute and if some one enters any other character which is different than case constants then by default the code inside default will get executed .

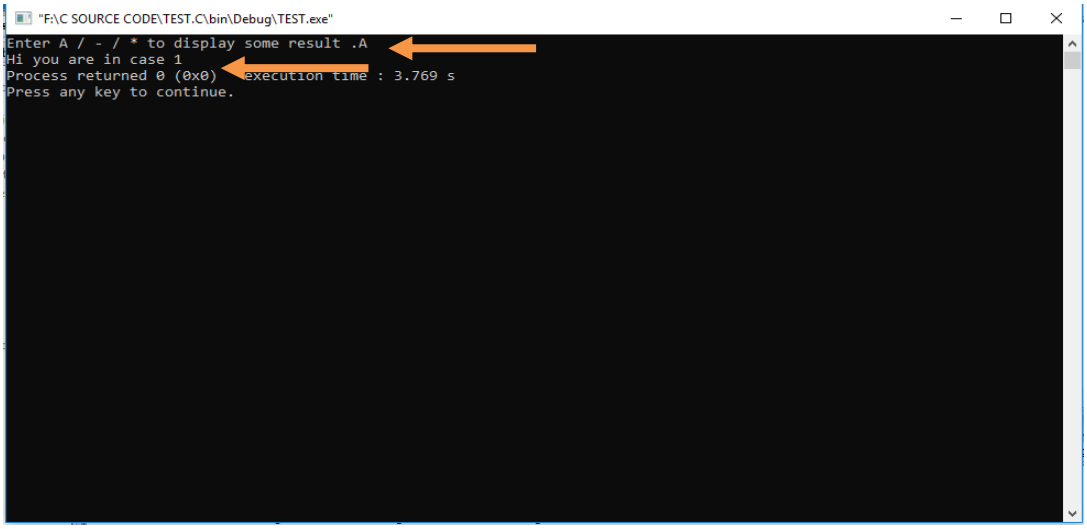
/* Following example show use of switch statement with operators and character constant .

Author :- Prashant Shinde . */

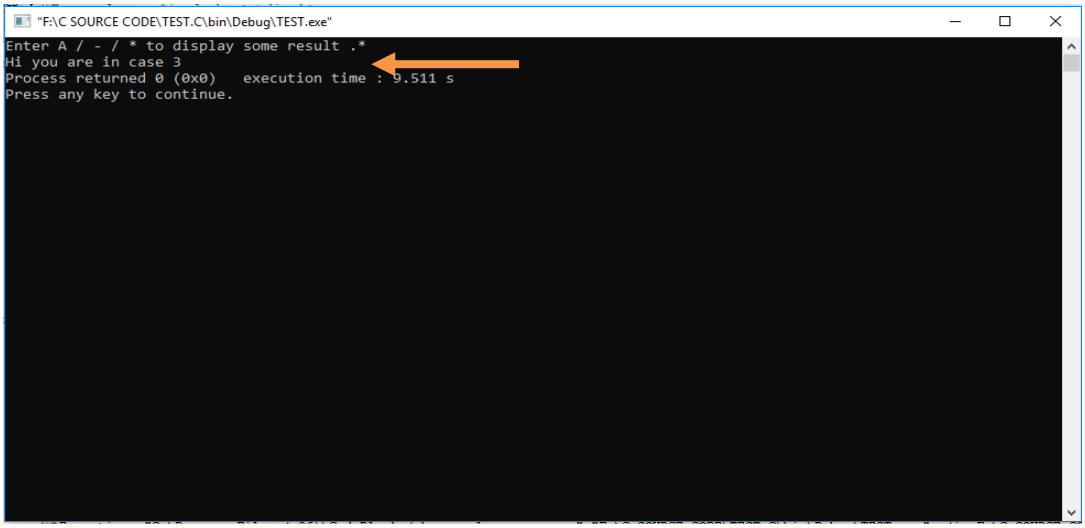
```
#include < stdio.h >

#include < conio.h >

#include < stdlib.h >

int main ()

{

  char  a ;

  system (" cls ") ;

  printf (" Enter  A / - /* to display some result .") ;

  scanf (" %c ", & a ) ;

  switch ( a )

  {

      case 'A' :

     printf (" Hi you are in case 1 ") ;

     break ;

 case '-' :

    printf (" Hi you are in case 2 ") ;

     break ;

 case '*' :

    printf (" Hi you are in case 3 ") ;
```

```
    break ;
 default :
    printf ( " Hi you are in default case " ) ;
    break ;
  }
  return 0 ;
}
```

Following is the output when I had entered character A from keyboard . Please don't put any space before A , otherwise it will not work properly. Because a single space is get stored as single character in programming .



Following is the output when I had entered multiplication operator * .



So now I think you have got the basic understanding of how switch statement works . So solve the following assignment to get the better understanding of it .

ASSIGNMENT :- >> Write a c program which will display a city name to output screen on the basis of users choice . In this program user needs to input characters from A to D . If user enters any other character or value then the program will display a message "You are lost !" .
If user enters A then it should display a message " Hi You are in America " .
If user enters B then it should display a message " Hi You are in Bangladesh " .
If user enters C then it should display a message " Hi You are in China ".
If user enters D then it should display a message " Hi You are in Denmark ".

iv] Conditional Operator :- >> Conditional operator is also called as " ternary operator " . It is denoted by two operators i.e. number one is " question mark " and second one is " colon " as ? : . Normally it works like if else statement . It takes 3 arguments . It displays the result on the basis of the result of first argument . If the expression or condition in first argument becomes true then it will automatically execute the second argument and if expression or condition in first argument becomes fail then it will automatically execute third argument as result .

Syntax :- >>

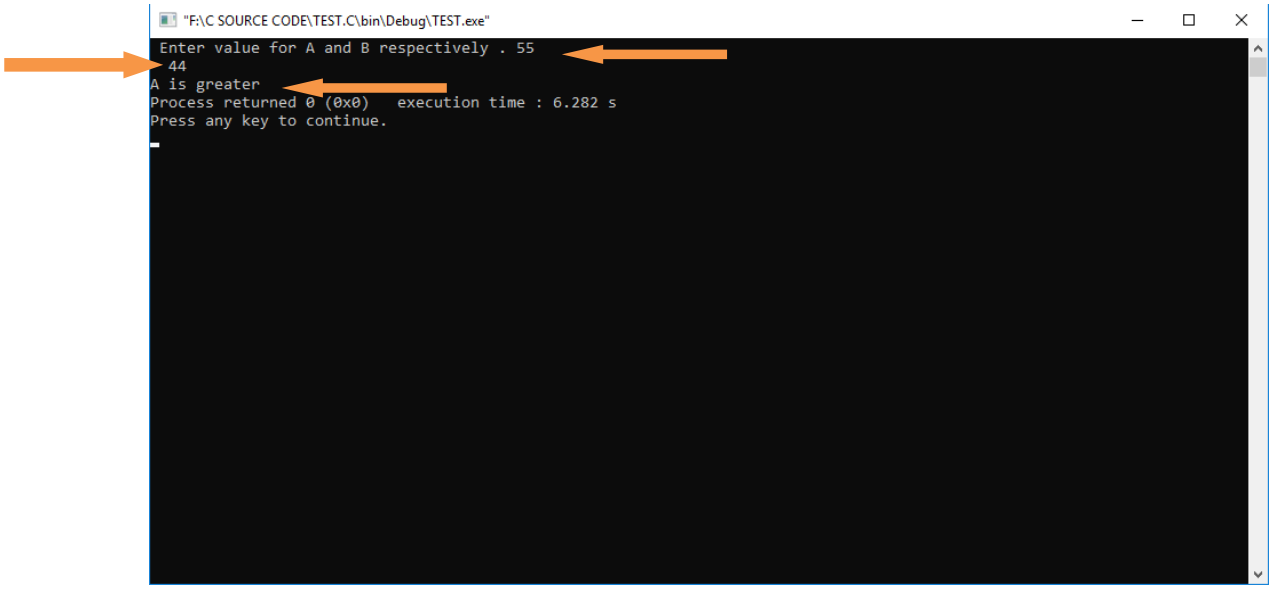
Expression ? argument 2 : argument 3 .

Let's understand the working of conditional operator ? : with practical example . So let's write a program which display the greatest values in between two values entered via keyboard by using conditional operator .

```
/* Write a c program to show the working of conditional operator
    Author :- Prashant Shinde   */
#include < stdio.h >
#include < conio.h >
#include  < stdlib.h >
int main ()
{
  int a , b ;
  system ( " cls " ) ;
  printf ( " Enter value for A and B respectively .");
  scanf ( " %d%d " , & a , & b ) ;
  ( a > b ) ? printf ( " A is greater  " ) : printf ( " B is greater ! " ) ;
  return 0 ;
}
```

In the above program we have received two values for a and b respectively . Then tested them by setting condition as (a > b) and after that we have used conditional operator to display the result . We have set the message in printf() on the basis of working of conditional operator .Because when condition (a > b) becomes true then it will execute second argument i.e. printf(" A is greater ! ") ; and if condition becomes false then it will execute third argument printf (" B is greater ! ") ; .

Following is the screenshot showing output of above program where I had entered two values 55 for a and 44 for b .



ASSIGNMENT :- >> Write a c program with the help of conditional operator which will display the smallest value in between two entered values via keyboard for a and b respectively .

KEY POINTS :- >>

- In decision making & branching you need to use logical operators or relational operators to create any type of logic depending upon your programming needs .
- Whether it is if statement or if else statement you don't need to use the semi colon .
- If you have only single line of code then you can use if statement or if else statement without the use of opening & closing curly brackets { } .
- In programming , at some particular points zero means false and one means true , when you pass zero via if or if else statement then the if condition will become automatically false and if you pass one via if or if else statement then the condition of if or if else statement will become automatically true .
- Switch statement is consisting of four different keywords in c such as switch , break , case , default .
- Conditional operator is basically acts like an if else statement in c .

6. LOOPING IN C .

What is looping :- >> Looping means executing the same code or particular block of code again and again until a particular condition .There are some situations in programming where we need to execute same code again and again to accomplish a particular task then at such situations looping is helpful .There are mainly 3 types of loops are there . So let's see them one by one .

i] While loop :- >> While loop is declared with the help of keyword while followed by opening and closing parenthesis .Inside the parenthesis you need to set a particular condition to execute the while block again and again .Following is the Basic syntax to declare the while loop .

Syntax :- >>

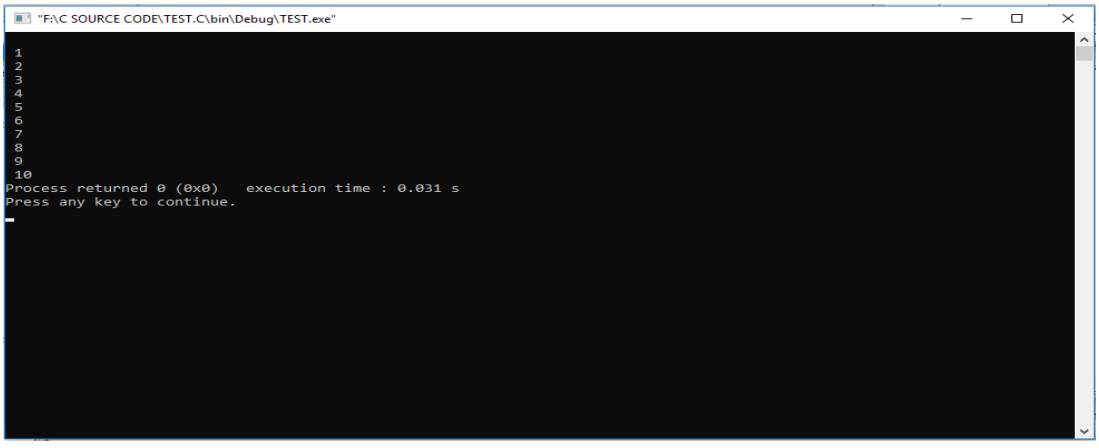
```
while ( condition )
{
 code to execute ;
}
```

Above is the basic syntax to declare the while loop . The while executes the code inside the while block until the condition becomes false . When the condition becomes false then the program control comes out of that while block and excutes next line of code whatever it is . So let's see the working of while loop with a practical example . Following is the program which will print the values from 1 to 10 with the help of while loop .

```
/* Write a c program to print the values from 1 to 10 with the help of while loop .
   Author :- Prashant Shinde . */
#include < stdio.h >
#include < conio.h >
int main ()
{
  int x = 1 ;
```

```
system ( " cls " ) ;
while ( x < = 10 )
{
  printf( " \n %d ", x ) ;
  x++ ;
        }
return 0 ;     }
```

Now let's see how this program works .When the program control enters in to main () function variable x is get created and get initialized to 1 . Then due to system(" cls ") ; the previous output is get cleared . After that the program control checks the while condition , now the value of x is 1 so the condition of block (x<=10) i.e. x is less than or equal to 10 becomes true so that's why program control enters into while block . After that it will print the current value of x i.e. 1 with the help of printf() function . After that the program control reaches to x++ , here ++ is an increament operator which increase the value of x by 1 so now the value of variable x becomes 2 , so after here the program control reaches to while loop to test the condition and again prints the new values of x and after that value of x get increased due to x++ and again it reaches to test the while loop condition . So it's gonna print the value of x until condition of while block becomes false . so when the value of x becomes 11 at this time the condition of while loop becomes fase and the program control comes out out that loop . So this is how while loop works . It will print the while block until the condition of while becomes false . Following is the output of the above program where the while loop has printed values from 1 to 10 .



So now I think you have got the basic understanding of how loop works . So solve the following assignment on your own .

ASSIGNMENT :- >> Write a c program with the help of while loop to print values from 10 to 1 . Means you need to reverse the order of numbers displayed in above program . (Hint :- use decreament operator --)

ii] Do while loop :- >> It is next type of loop introduced in c . In this loop the code written within do block is get executed first and then the condition of while loop get tested . It executes the do block until the condition of while statement becomes false . Following is the basic syntax to declare do while loop .

Syntax :- >>

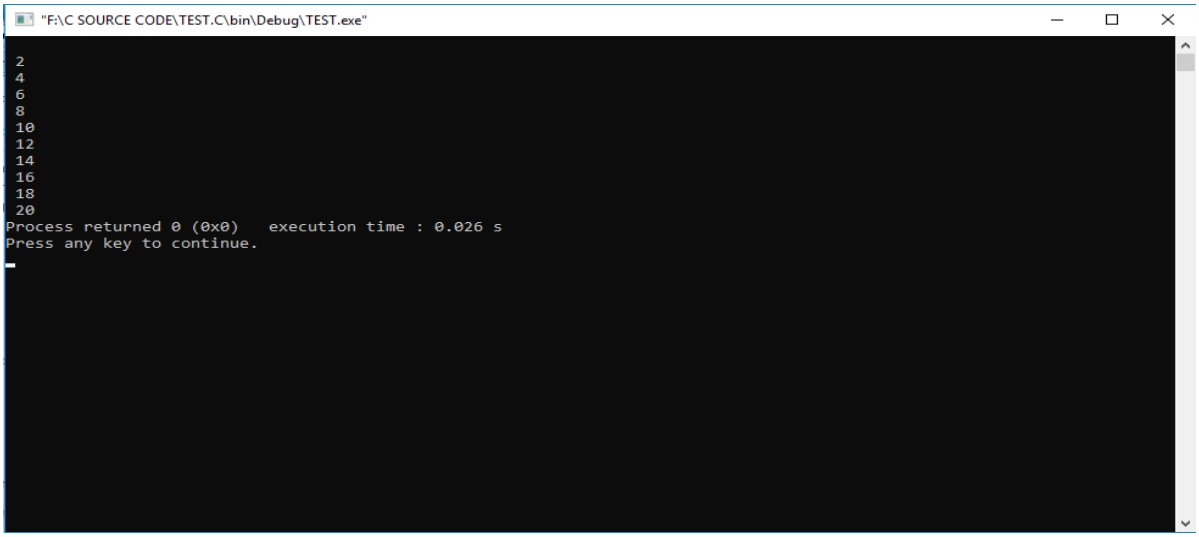
```
do {
    code to execute ;
} while( condition ) ;
```

So in do while condition the do block will get execute until condition of while becomes false .First the program control enters in do block executes the code present inside it & then it goes to while statement and test the condition and if the condition is true then it again goes to do block to execute the same code . Here you need to end the while loop with semicolon ; . So let's see how do while statement works practically with an example . Here we gonna write a program to print table to 2 with the help of do while loop .

```
/* Write a c program to print the table of 2 with the help of do while loop .
   Author :- Prashant Shinde . */
#include < stdio.h >
#include < conio.h >
#include < stdlib.h >
int main ()
```

```
{
  int x = 1 ;
  system( " cls " ) ;
  do
  {
    printf(" \n %d ", x * 2 ) ;
     x++ ;
  } while( x < = 10 ) ;
  return 0 ;
}
```

Now here , let's understand how this program works . first of all the program control enters in main function . It creates an integer variable x and assign a value 1 to it , then due to system("cls"); function it clears the previous output . After that it enters into do block and prints the value of x to screen by multiplying it with 2 .It's because we have set the condition inside printf() as [x*2] . After that it reaches to x++ and here due to increament operator value of x now becomes 2 , then after it checks while condition , now value of x becomes 2 , So 2 is less than 10 so condition of while statement becomes true and thus it again reaches to do block to execute the same block . Now at this time value of x is 2 so 2*2 becomes 4 ,so that this result is get printed on screen .After that again value of x increase to 3 due to x++ and again while condition is get tested and again it will reach to do block because condition become true . So it will print the do block until while condition becomes false . So this is how we can get a table of 2 by applying the simple logic of operators . following is the output of the above program .



ASSIGNMENT :- >> Write a c program to print the table to 2 in reverse order . Means you need to print the values form 20 to 2 . (Hint :- use decreament operator) .

iii] for loop :- >> for loop is the next loop which is introduced in c language . for loop is declared with the help of keyword "for" followed by opening and closing parenthesis () . For loop is consisting of 3 different sections and those are 1] initialization 2] condition testing 3] increament / decreament . The first and section section is always separated by a semicolon ; . Following is the basic syntax to declare a for loop .

Syntax :- >>

```
for ( initialization ; condition testing ; increament or decreament )
        {
            code to execute ;
        }
```

Above is the basic syntax to declare for loop in c . Now let's see how it works . First of all program control enters into for loop and checks whether the for loop is initialized or not , if it is initialized then it will check the for condition , if condition becomes true then it will goes

to for loop block { } and executes the code inside the block .After executing the code inside the block the program control goes to increament / decreament section and increament or decreaments the variable and again checks the condition of for loop if condition is true then it goes to the block and executes the same code again . The process repeats itself until the condition of for loop becomes false and when condition becomes false , the program control comes out of for loop and execute next statements which are present after for loop . So this is how the for loop works .

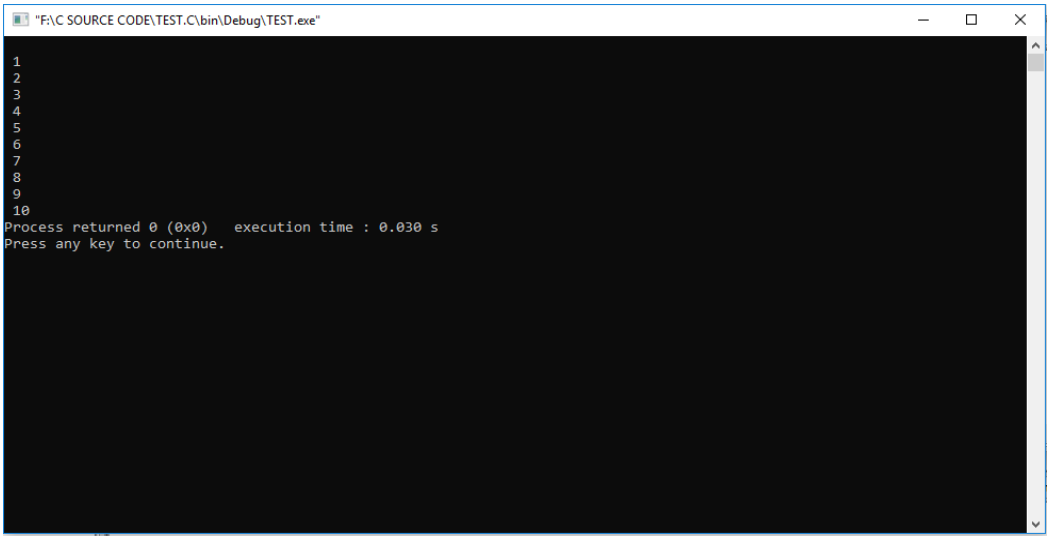
Let's understand the working of for loop with the help of an practical example .

```
/*  Write a c program to print the values from 1 to 10 with the help of for loop .
    Author :- Prashant Shinde .  */
  #include  < stdio.h >
 #include < conio.h >
#include < stdlib.h >
 int main ()
   {
    int x ;
   system( " cls " ) ;
   for ( x = 1 ; x < = 10 ; x++ )
  {
      printf( " \n %d ", x ) ;
   }
      return 0 ;
     }
```

Now let's understand how this program works . first of all program control enters into main function and creates variable x . after that clears the computer screen due to system("cls"); function . after that it enters into for loop , and initialize the x to 1 . This is the initialization part of the for loop , now value of x becomes 1 , now it goes to testing section where it test whether x is less than or equal to 10 , at this time condition becomes true so now the

program control enters into for block { } and prints the value of x due to printf() . after that it goes to increament section and increases the value of x due to increament ++ operator . After increament the value by 1 now x becomes 2 , now it goes to testing condtion ,so becase 2 is less than 10 , so the condition becomes true and program control goest to for block { } prints new value of x i.e. 2 . So this is how this procedure repeats it self until the condition of for block { } becomes false and as soon as the condition becomes false the program control comes out of for loop . so this is how it prints values from 1 to 10 because of the condtion we have set i.e. x<=10 .

Following is the output of the above program which shows values from 1 to 10 .



ASSIGNMENT :- >> Write a c program with the help of for loop to print the values from 10 to 1 .

iv] Nested for loop :- >> When we use one for loop within another for loop then it is called as " Nested for loop " . There are some situations where we need to use nested for loop in c programming . So that's why this feature has been added in c where we can add one for loop within another for loop . following is the basic syntax to declare nested for loop .

Syntax :- >>

```
for ( initialization  ;  condition testing  ;  increament or decreament )
{                                                    /* start of outer for loop */
 for( initialization ; condition testing ; increament or decreament )
      {
                                                /* start of inner for loop */
            code to execute ;


      }                    /* end of inner for loop */
 }                                  /* end of outer for loop */
```

Above is the basic syntax to declare nested for loop. Now let's understand how it works . first program control initializes the outer for loop and checks the condition if condition becomes true then it enters into inner for loop & if condition becomes false then it will skip the whole for loop . After entering into outer for loop now the inner for loop is get initialized and here it checks for the condition if condition becomes true then it executes the code inside the inner for loop block { } . When the condition of inner for loop becomes false then the program control goes to outer for loop block and goes to increament or decreament section and after that it checks for the condition in outer for loop block , if the condition becomes true then it again goes to inner for loop block and execute that inner for loop block until the condition becomes false inside inner for loop block . This cycle repeats it self until the condition of outer for loop block becomes false . so this is how this nested for loop works . In short you can also say that the nested for loop executes the inner for loop until the condition of outer for loop becomes false .

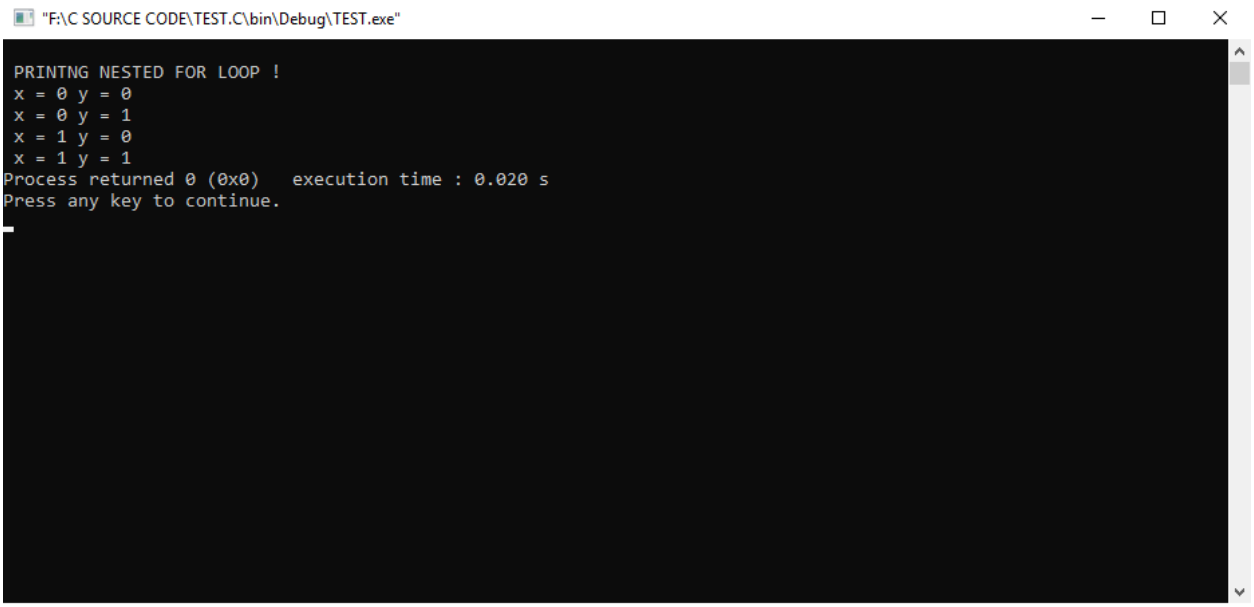
Now let's understand the working of nested for loop with the help of following practical example .

```
/*  Write a C Program to understand the working of nested for loop
```

```
   Author :- Prashant Shinde .  * /


 #include  < stdio.h >

#include < conio.h >

#include  < stdlib.h >

int main ()

{

      int x , y ;

    system ( " cls " ) ;

  printf ( " \n PRINTNG NESTED FOR LOOP ! " ) ;

   for ( x = 0 ; x < 2 ; x++ )

   {

         for ( y = 0 ; y < 2 ; y ++ )

     {

         printf( " \n x = %d y = %d " , x , y ) ;

       }

  }

  return 0;

}
```

Now let's understand how the above program works . first of all the " program control" enters into main function and creates integer variable x,y respectively . after that clears computer screen due to system (" cls ") ; . Then it prints the message on screen " PRINTING NESTED FOR LOOP ! " . After that it enters into outer for loop and initializes it " x = 0 " and then checks the condition x< 2 , because the current value of x is 0 so that the condition becomes true and it enters into block of outer for loop and then goes to inner for loop . Here it initialized y=0 and checks the condition y< 2 , because the current value of y is 0 so that the condition becomes true and it enters into inner for loop block { } and prints the current

values of x & y respectively i.e. x = 0 and y = 0 . After that it goes to increament section of inner for loop and checks the condition , now this time value of y becomes 1 , so condition y<2 becomes true and it again goes to inner for loop block and prints the current values of x and y respectively i.e. x =0 and y=1 , after that it again goes to increament section and increases the value of y , now y becomes 2 , now this time the condition y<2 becomes false , because 2 is not less than 2 . So this time the program control reaches to increament section of outer for loop and now value of x becomes 1 . after that it checks the condition x< 2 , condition becomes true and it again goes to inner for loop and prints the current values of x and y respectively i.e . x = 1 , y = 0 . So this is how our loop continue to print the values of x and y until the condition of outer for loop becomes false . So this is how nested for loop works . I hope all of you have get the basic understanding of how nested for loop works . Following is the output of above program where you can observe the values of x and y respectively .



ASSIGNMENT :- >> Write a c program with the help of nested for loop to print the values of x & y where the value of x is 5 and y is 5 and write down it's working on paper .

KEY POINTS :- >>

- Loops are used to print the same line of code again & again for a particular reason .
- Also in case of loops , zero means false and one means true , that means when you pass one then loop will be executed infinite times & when you pass zero then loop condition will become automatically false and it will skip the code within the loop block .
- In for loop you need to pass zero or one three times separated by commas in order to work it correctly .
- Except do while loop you don't need to use semicolon ; .
- You can also use any of the loop without the use of opening & closing curly brackets { } if you have single line of code to execute .

7. ARRAY IN C

Until now we have learned how to create variables to store data . Suppose you want to store name of two students then you can simply create two variables like student1,student2 . But what if you want to store name of such 100 students . Then declaring 100 variables and then assigning names to them looks hectic task right . So to overcome such problems the concept of array is introduced in c . So by declaring array we can overcome such problems very easily .

What is an array ?

An array is a collection of similar type of data element which stores similar types of data element in sequential order or Array is a particular type of data structure which is used to store the similar type of data elements .

Ok, now let's see how to declare and access array ?

There are two steps to use array feature . The first one is declaration of an array and second one is accessing the array . So lets see how to declare an array .

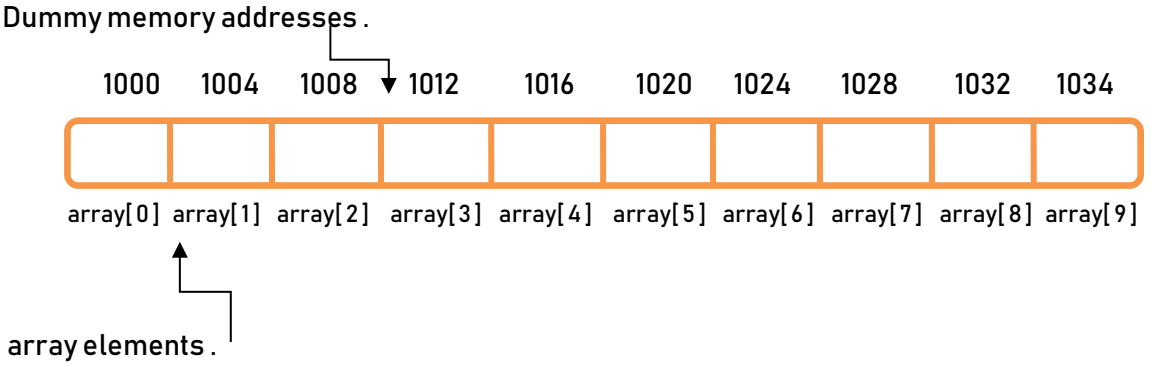
i] Declaring an Array :- >> To declare an array you need to first declare data type then array name & then inside opening and closing square brackets [ ] you need to set the size of array and end the array with semicolon ; . Following is the Basic syntax to declare an array .

Syntax :- >> Data_Type array_name [array size] ;

Example :- >> int x [10] ;

In the above example we have declared an array of integer type which can store data of 10 integer variables because of size 10 . The array has name " array " , you can set any name you want . So let's understand how this works . When we declare an array in c then the compiler allocates a continuous blocks of memory locations . It allocates the array as array [0] , array [1] , array [2] , array [3] , array [4] , array [5] , array [6] , array [7] , array [8] , array [9] . You can store 10 different names to each array element here . All these variables from array [0] to arrya[1] is called as " Array element ". The array elements are always starts with zero and end with less than array size by 1 . So that's why

our element is started with array [0] and end with array [9] . Following graphics shows how array elements are stored in sequential order in computers memory . All the memory addresses are shown below are dummy memory addresses .

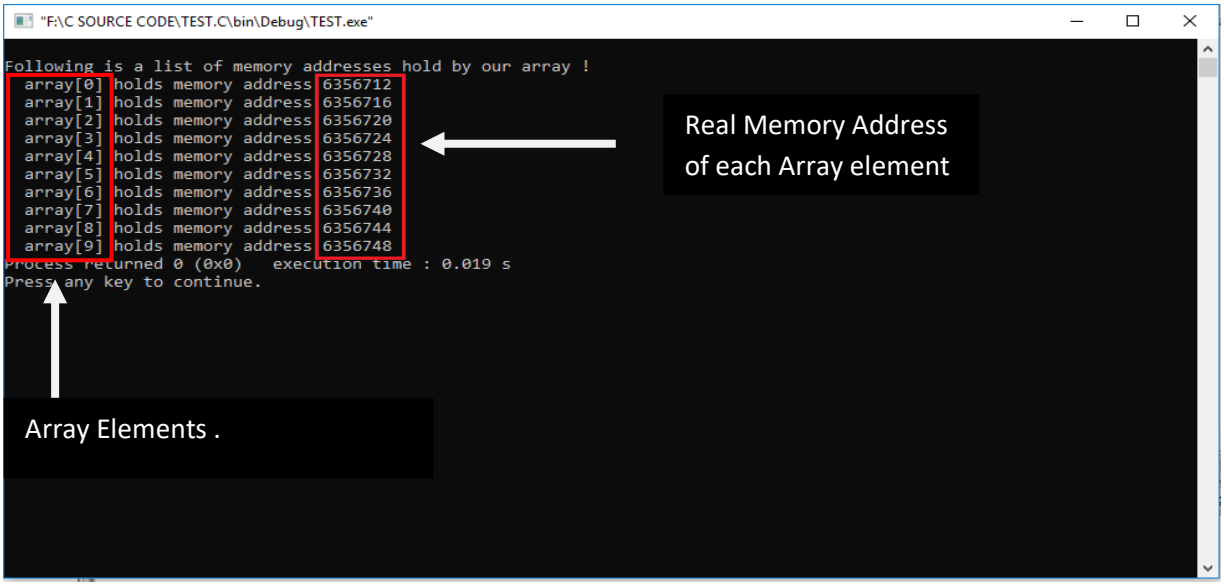


Now let's see a practical example to understand array elements allocation practically .

Following program will show you how our array allocates a continuous memory locations in c with real memory addresses .

```
/*  Write a c program to understand memory allocation in array

    Author :- Prashant Shinde  */

#include < conio.h >

#include < stdlib.h >


int main ()

{

  int array [ 10 ] ;

  system ( " cls " ) ;

  printf ( " \n Following is a list of memory addresses hold by our array ! " ) ;

  printf ( " \n  array [ 0 ] holds memory address %u " , & array [ 0 ] ) ;

  printf ( " \n  array [ 1 ] holds memory address %u ", & array [ 1 ] ) ;

  printf ( " \n  array [  2  ] holds memory address  %u " , & array [ 2 ] ) ;

  printf ( " \n  array [  3  ]  holds memory address %u ", & array [ 3 ] ) ;
```

```
    printf ("\n array [4] holds memory address %u ", &array[4]);
    printf ("\n array [5] holds memory address %u ", &array[5]);
    printf ("\n array [6] holds memory address %u ", &array[6]);
    printf ("\n array [7] holds memory address %u ", &array[7]);
    printf ("\n array [8] holds memory address %u ", &array[8]);
    printf ("\n array [9] holds memory address %u ", &array[9]);
    return 0;
}
```



By observing the above output of the program you can easily understand how array allocates a continuous blocks of memory locations . It always start with zero and ends with less than 1 by array size . One another important point , the size of each memory location is always depends upon the data type that you have used in array . We have used integer data type to declare our array so that's why it is showing a difference of 4 bytes with each memory location . Because code blocks is a 32-bit compiler so that's why it is showing the size of integer variable 4 bytes , but if you execute the same code on Turbo c++ 3.0 compiler which is 16-bit compiler then you will see the size of 2 bytes . Because size of integer variable in Turboc++ 3.0 is 2 bytes . So just observe the real addresses of each memory

locations above ,and specially observe the last two digits , you will automatically find the difference of 4 between each number like 12, 16, 20 .

ii] Accessing Array : ->> above we have learned how to declare an array , now it's time to learn how access that array . How to access it to insert and display the data . so the array is normally accessed with the help of an for loop . so let's see how to access above array with the help of for loop . Following is the basic example which will show you how to access array with the help of for loop .

```
/* Write a c program to access array element
   Author :- Prashant Shinde . */
#include < stdio.h >
#include < conio.h >
#include < stdlib.h >
int main ()
{
  int array [ 5 ] , i ;
  system ( " cls " ) ;
  for ( i = 0 ; i < 5 ; i ++ )
  {
  printf ( " \n Enter 5 values one by one ! " ) ;
  scanf( " %d ", & array [ i ] ) ;
  }
printf( " You have Entered following values ! " ) ;
   for ( i=0 ; i < 5 ; i++ )
  {
```

```
  printf( " \n %d ", array [ i ] ) ;
  }
  return  0 ;
}
```

Above program creates an array of 5 integers and access them with the help of for loop and print those values to computer screen with the help of for loop and printf(). So now let's understand how it works step by step .

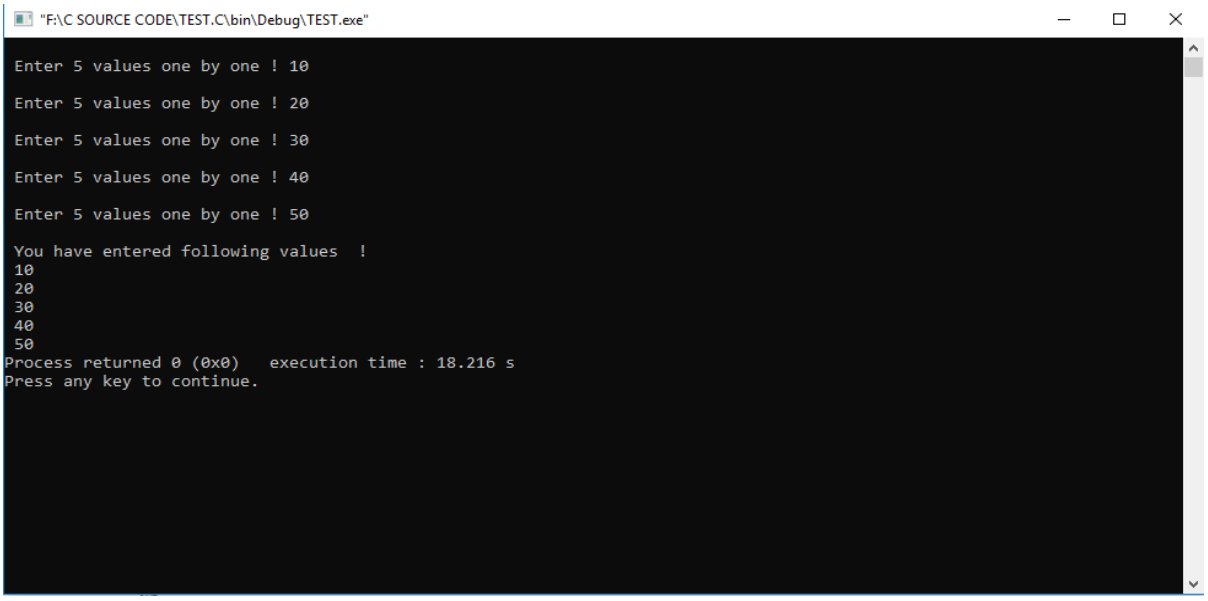
Step i :- >> In first step we have declared an array of size 5 and a variable " i " to work with for loop . so array [5] will reserve 5 continueous memory locations of 4 bytes each . after that a variable " i " will get create and reserves 4 bytes in computers memory .

Step ii :- >> Due to system("cls"); the computer screen will get cleared and the program control will get entered into for loop , inside the for loop the variable " i " will get initialized to 0 after that condition will get check , (i < 5) this condition will become true because current value of " i " is 0 & and then program control will enter into for loop block { } . As we have learned above that each array element always starts with zero and ends with less than 1 by array size . After that we will be able to see the message on screen that " Enter 5 values one by one ! " and then we will be into scanf() function . In the scanf( ) function we have declared " &array [i] " because the current value of " i " is 0 so that's why we can access the first array element array [0] to insert the data . Then due to for loop the value of " i " becomes 1 so now we can access the next element array [1] so in such way we can access each array element to store the data due to for loop .

Step iii :- >> So after inserting the data one by one , the program control will come out of for loop and it will print the message on computer screen " You have entered following values !" .

Step iv :- >> After that due to next for loop & printf( ) function , we can print those entered values on screen . So in this for loop the variable " i " will get initialized to 0 again and then condition will get checked and due to printf() it will print the value of array element array [0] to computer screen . After that value of " i " will become 1 and then it will print the value of array element array [1] . So in this fashion it will print the values of each array element one by one .

Following is the output of above program which show which values are entered and then shows those entered values to screen .



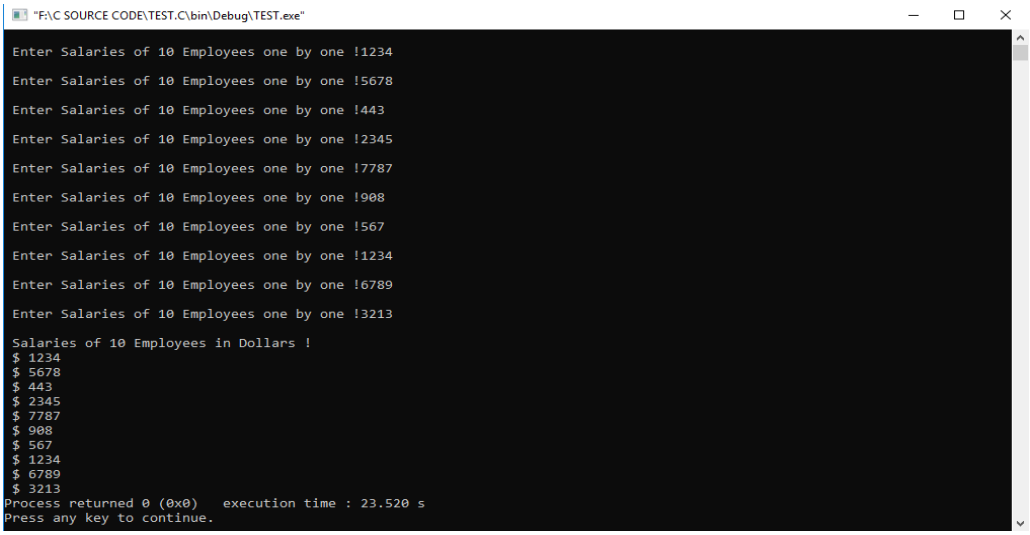
Types of Array :- >> Above we have learned the basics of how to declare and access an array . Now let's see different types of array . On the basics of dimensions array is devided in to 3 types such as 1] One dimensional array 2] Two dimensional array 3] Multidimensional array .So here we gonna learn about one dimensional and two dimensional array .

A] One Dimensional Array :- >> The array which contains only one subscript or dimension is called as " One dimensional Array " . The example that we saw above is an good example of one dimensional array . The one dimensional array is declared with the same syntax that we have used above i.e. Data_type array_name [size] ; and it is accessed with the help of for loop as shown above .So let's write a c program to understand One dimensional array . Following program will receive and print salaries of 10 employees in dollars .

```
/*  Write a c program to understand One dimensional array

    Author :- Prashant Shinde .  */

#include < stdio.h >
```

```
#include < conio.h >

#include  < stdlib.h >

int main ( )

{

  Int  salary [ 10 ] , i ;

  system("cls");

  for(i=0;i<10;i++)

  {

  printf("\n Enter Salaries of 10 Employees one by one  !");

  scanf("%d",&salary[i]);

  }

  printf("\n Salaries of 10 Employees in Dollars !");

   for(i=0;i<10;i++)

  {

  printf(" \n $ %d",salary[i] );

  }

  return 0;

}
```

Following is the output of the above program . You can easily understand the working of program by observing it's output .



Now let's understand how this program works . First the program control enters into main function and creates an array of 10 integers with name salary and also creates a variable " i " to use with our for loop . In computers memory 40 bytes are get reserved due to integer type array from salary[0] to salary[9] and another 4 bytes due to variable "i" . Then after that due to system("cls"); our screen get cleared . After that our program control enters into our for loop and initialize variable "i" to 0 , then check condition " i < 10 " , here condition becomes true because current value of variable "i" is 0 , then after it enters into our loop block and prints message on computer screen " Enter salaries of 10 Employees one by one . " after than scanf() will receive salary of first employee due to salary[i] . After that program control goes to i++ and now value of "i" become 1 now it again checks condition "i < 10 " and condition becomes true . Now this time it receives salary of next employee i.e. salary [1] . So this is how it receives salaries of 10 employees until the condition of loop becomes false . After this the program control comes out of for loop and goes to next line of code and prints message to screen " Salaries of 10 Employees in dollars ! " . After this it goes to next for loop , this for loop reads all the salaries above entered and prints it to screen with the help of printf() function . I hope all of you have get the basic understanding of one dimensional array and it's working .

ASSIGNMENT :- >> Write a c program with the help of one dimensional array to receive and print the marks obtained by 5 students in computer subject .

Initialization of one dimensional array : - >> Instead of receiving values from keyboard we can directly initialize the values to our array where it is declared . So initializing the array where it is declared is called as " array initialization " . One dimensional array is initialized with following syntax .

```
data_type  array_name [ 5 ] = { value1 , value2 , value3 , value4 , value5  } ;
```

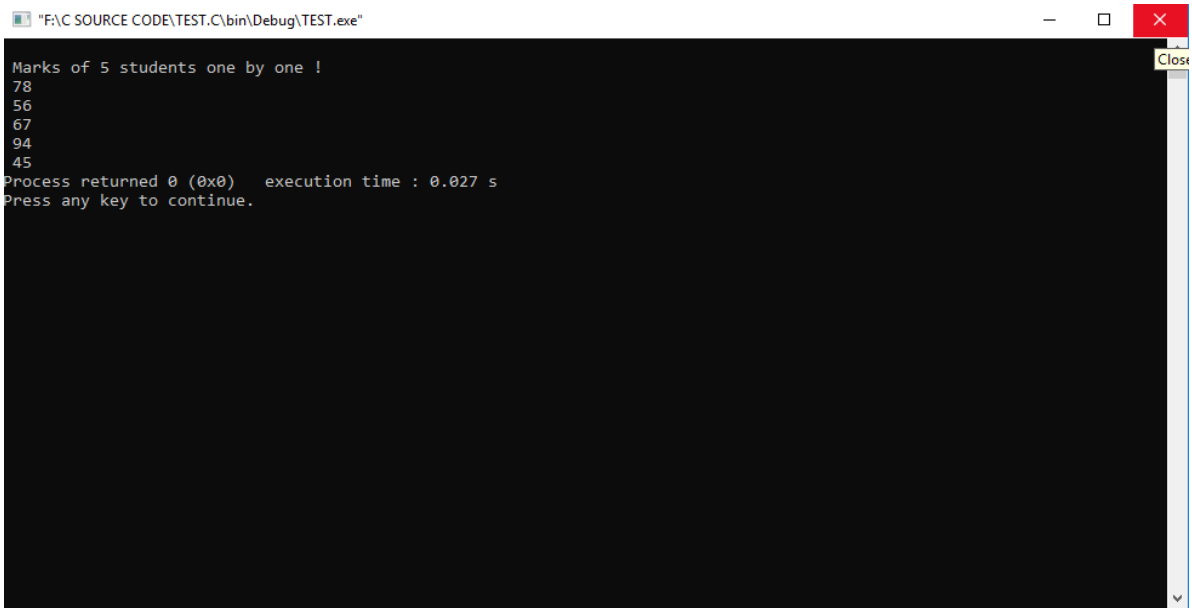
You need to use assignment operator ,exactly after declaring array and then need to use opening & closing curly brackets to initialize values . Each value mush be separated by comma and the closing curly bracket should always ends with semicolon ; .

So let's understand how one dimensional array initialization works with the help of following program .

```
/*  Write a C Program to initialize one dimensional array .
    Author :- Prashant Shinde .  */
#include  < stdio.h >
#include < conio.h >
#include  < stdlib.h >
int main ()
{
  int marks [ 5 ] = { 78 , 56 , 67 , 94 , 45 } ;
  int i ;
  system( " cls " ) ;
  printf( " \n Marks of 5 students one by one ! " ) ;
   for ( i = 0 ; i < 5 ; i++ )
  {
  printf ( " \n %d " , marks [ i ] ) ;
  }
```

```
return 0 ;
}
```

Following is the output of above program where it shows the initialized marks of 5 students one by one .



Now let's understand how this program works . First the program control enters into main function and creates an array of 5 integers with name "marks" as marks[5] . So here 20 bytes are reserved in computers memory . after that due to assignment operator " = " all the values that we have set inside opening and closing curly brackets assigns those values to reserved array elements or reserved memory locations one by one sequentially . after that it goes to next line and creats a variable "i" and then clears computer screen due to system("cls") ; . Then it shows the message on computer screen " Marks of 5 students one by one ! " due to printf() function . After that the program control enters into for loop , initialize variable "i" to 0 , checks condition i<5 and enters into for loop block and reads first value at " marks[0] " and prints it to screen , then goes to i++ and again test condition and again enters into for loop block and performs the same operation until condition of loop becomes false and this is how we can see all the values to output . I hope all you have get the basic understanding of how one dimensional array initialization works . To understand this more perfectly solve the following assignment .

ASSIGNMENT :- >> Write a c program to print the age of each family member sequentially with the help of array initialization .

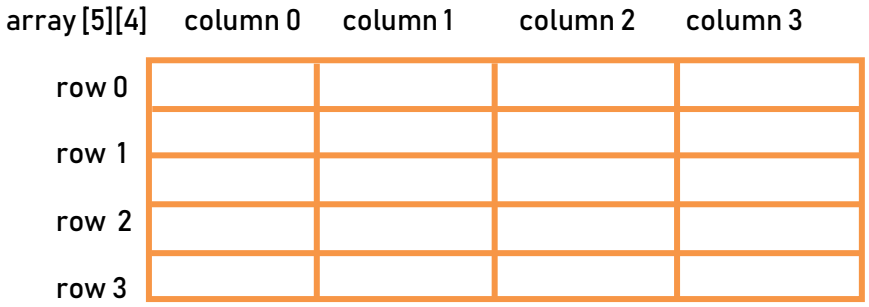
B] Two Dimensional Array :->> The array which contains two subscripts or dimensions is called as " Two Dimensional Array " . The two dimensional array is called as " Matrix " because it stores the data in matrix form (row x column) . The two dimensional array is declared with the help of following syntax .

Syntax :- >> Data_type array_name [size 1] [size2] ;

In two dimensional array size 1 is called as row and size 2 is called as column .Because it stores the data in row x column format .

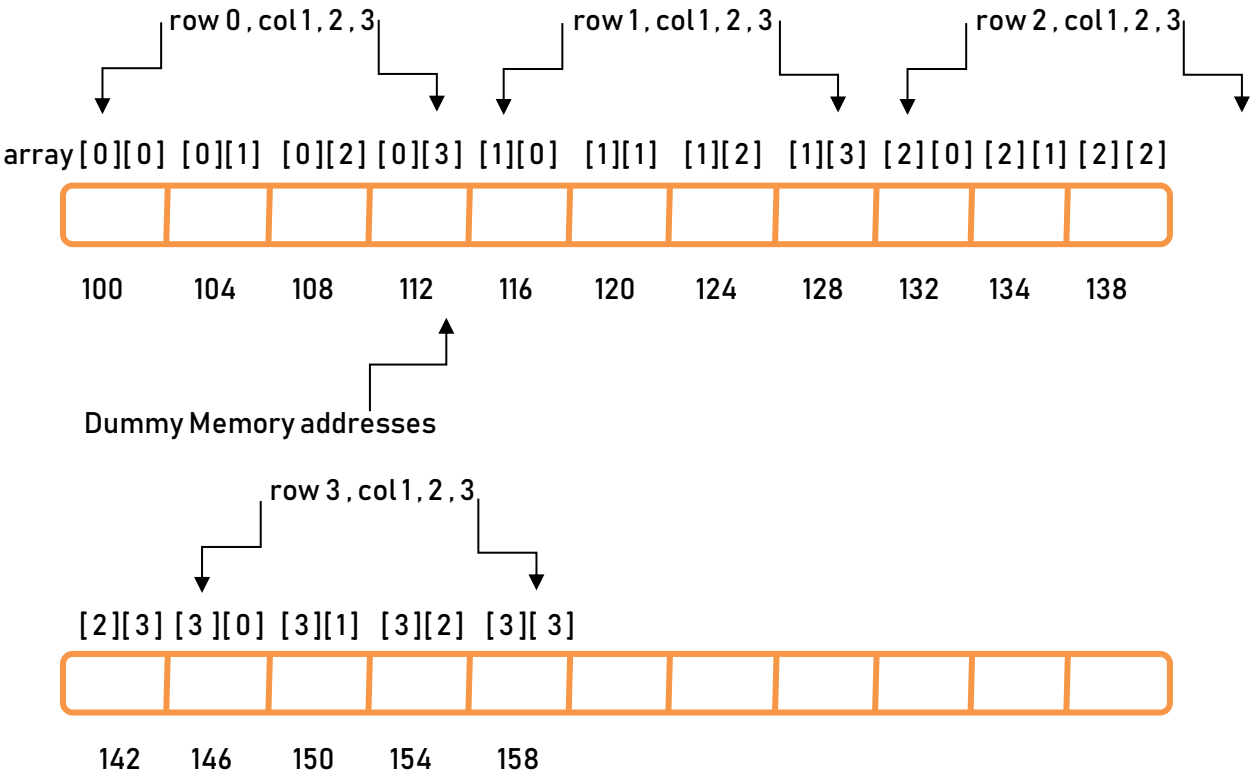
Example :- >> int array [5] [4] ;

So when we declare the above array then it creats an array of name array of size of 5 rows and 4 columns in computers memory . Means in total it can store up to 20 values . Because 5 x 4 = 20 . following graphical representation will help you to understand how two dimensional array is get created in computers memory for the above example .



2-Dimensional array elements are stored as row x column format inside the computers memory . It stores the array elements as array [row 0] [col 0] , array [row 0] [col 1] , array [row 0][col 2] , array [row 0] [col 3] , afte this next row and same columns as array [row 1] [col 0] , array [row 1] [col 1] , array [row 1] [col 2] , array [row 1] [col 3] , after this the next row and same columns as array [row 2] [col 0] , array [row 2] [col 1] ,array [row 2] [col 2] , array [row 2] [col 3] and after this the last row and same columns as array [row 3] [col 0] , array [row 3] [col 1] , array [row 3] [col 2] , array [row 3] [col 3] .

Following is the memory map of 2 D array, which show how array elements are stored inside the computers memory as row x column format with continuous memory addresses .



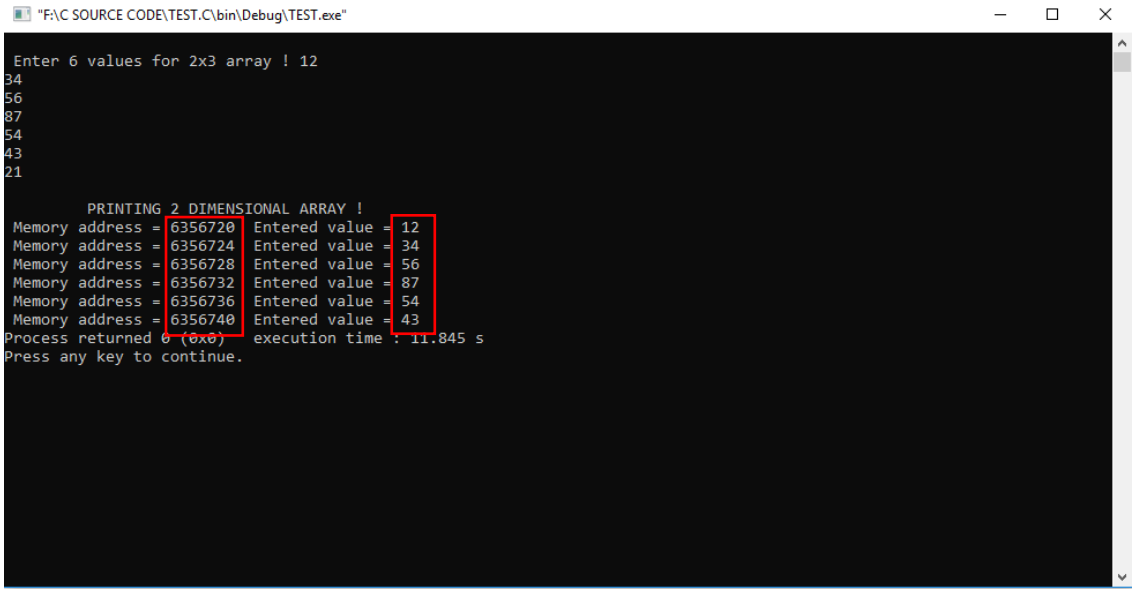
I hope all of you have get the better understanding of how 2 Dimensional array elements are stored inside the computers memory in continues memory locations in row x column format .

Accessing 2 Dimensional array : - >> after understanding how array elements are stored , now let's understand how to access them . We use for loop to access one dimensional array , in the same way to access 2 dimensional array nested for loop is used . So now let's understand how an 2 Dimensional array is declard and accessed with the help of an practical example .

```
/* Write a c program to understand 2 Dimensional array
   Author :- Prashant Shinde */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main ()
{
  int array [2] [3];
  int x , y;
  system("cls");

  printf("\n Enter 6 values for 2x3 array!");
  for( x = 0 ; x < 2 ; x++ )
  {
    for ( y = 0 ; y < 3 ; y ++ )
    {
    scanf ("%d", & array [x] [y]);
    }
  }
```

```
  printf ( "\n\t PRINTING 2 DIMENSIONAL ARRAY ! " ) ;
  for ( x = 0 ; x < 2 ; x++ )
  {
    for ( y = 0 ; y < 3 ; y ++ )
    {
     printf( "\n Memory address = %u  Entered value = %d " , & array [ x ] [ y ] , array [ x ] [ y ] ) ;
     }
  }
  return 0 ;
}
```

Now let's understand how above program works , first the program control enters into main function and creates an array of size [2][3] . It reserves continues 6 memory locations from array [0][0] , array [0][1] , array [0][2] , array [1][0] , array [1][1] , array [1][2] . Now after that we have declared variable x and y to store values at these array elements with the help of nested for loop . after that system(" cls ") will clear the computer screen . Next it will print the message on computer screen that " Enter 6 values for 2x3 array ! " and the program control will enter into outer for loop and initializes the x=0 and checks the condition x < 2 , here the condition becomes true and it will enter into inner for loop and initializes y=0 and checks the condition y<3 , here again condition becomes true and then it will enter into scanf() and store our first entered value from keyboard to array element array[0][0] because of &array[x][y] , because of current values of x and y are 0,0 respectively . after receiving first value it will goes to y++ , now value of " y " becomes 1 , it will again checks condition y<3 , here condition becomes true , so next value will be stored at array[0][1] , because now current value x and y is 0,1 respectively . So in this fashion all the entered values get stored at continues memory locations until the outer for loop becomes false . we have set the values of outer for loop and inner for loop on the basis of size of our array . Following is the output of above program which will shows you entered value along with memory locations .



ASSIGNMENT :- >> Write a c program to receive and print an array of size 5 x 5 .

2 Dimensional array Initialization :- >> Above we have learned how to initialize one dimensional array , like one dimensional array we can also initialize 2 Dimensional array . Following is the basic syntax to initialize 2 Dimensional array .

Syntax :- >>

```
Data_type array_name[ size 1 ][ size 2 ] = { value1 , value 2 , value 3 , value 4 } ;
```

Example :- >>

```
A ] int array [ 4 ] [ 2 ] = { 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 } ;

                    or

B ] int array [ 4 ][ 2 ] = {
                        { 1 , 2 } ,
                        { 3 , 4 } ,
```

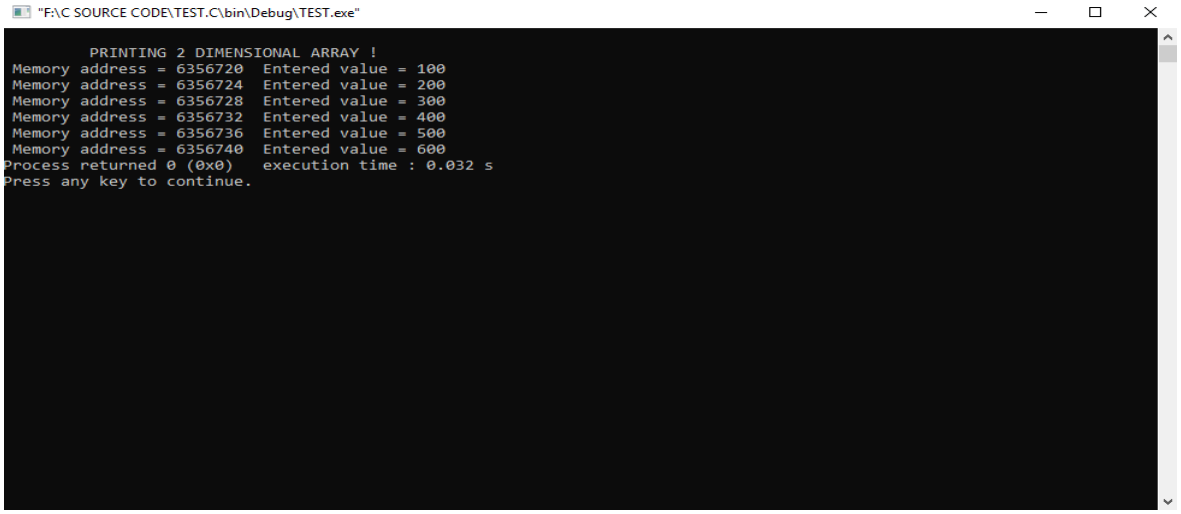
```
                    { 5 , 6 } ,
                    { 7 , 8 }
                  } ;
```

You can initialize the 2 Dimensional array either in style A or style B . Let's understand the 2 dimensional array initialization with an practical example .

```
/* Write a c program to understand 2 Dimensional array initialization .
   Author :- Prashant Shinde . */
#include < stdio.h >
#include < conio.h >
#include  < stdlib.h >

int main ()
{
  int array [ 2 ][ 3 ] = { 100 , 200 , 300 , 400 , 500 , 600 } ;
  int x , y ;
  system ( " cls " ) ;
  printf ( " \n \t PRINTING 2 DIMENSIONAL ARRAY ! " ) ;
   for ( x = 0 ; x < 2 ; x++ )
   {
    for ( y = 0 ; y < 3 ; y++ )
    {
    printf( " \n Memory address = %u  Entered value = %d " , &array [ x ][ y ] , array[ x ][ y ] ) ;
    }
  }
  return 0 ;
```

}

Following is the output of above program where you can see the initialized values to the output .



I think you don't need to explain how 2 D array initialization works inside c programming . To understand it in more detail solve the following assignment .

ASSIGNMENT :- >> Write a c program to print an already initialized array of size 5x5 .

Character array : ->> When we declare any character variable then we can store only one character at a time to that variable , but by declaring character array we can store many characters at a time .

Let's understand how we can use character variable to store single characters with the following example .

```
/*  Write a c program to understand the working of character variable
   Author :- Prashant Shinde  */
#include < stdio.h >
#include < conio.h >
#include  < stdlib.h >
int main ()
{
  char  x , y ;
  system( " cls " ) ;
  x = ' $ ' ;
  y = ' # ' ;
  printf ( " %c %c " , x , y ) ;
  return 0 ;
}
```

So as you can see we can only store a single character at a time to a character variable inside the single quote . In the above example we have initialized x = ' $ ' and y = ' # ' . You can not store a complete name as "prashant" to single character variable , so to avoid this problem you need to use character array . Following output shows the initialized characters to computer screen .

Declaring character array :- >> Now let's understand how character array can be declared . character array is declared in the same way as we declare normal array , but here we need to use character data type , instead of integer or float data type . so following is the same syntax .

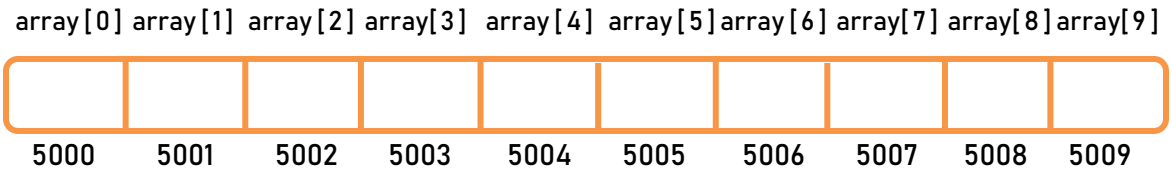
Syntax :- >>

```
char array_name [ size ] ;
```

Example :- >>

```
char  array [10] ;
```

Now let's see how character array elements are stored inside the computers memory .Each elements acquires one byte inside the computers memory . following is the memory map for above example . Each array element acquires continuous memory locations with a difference of one byte .



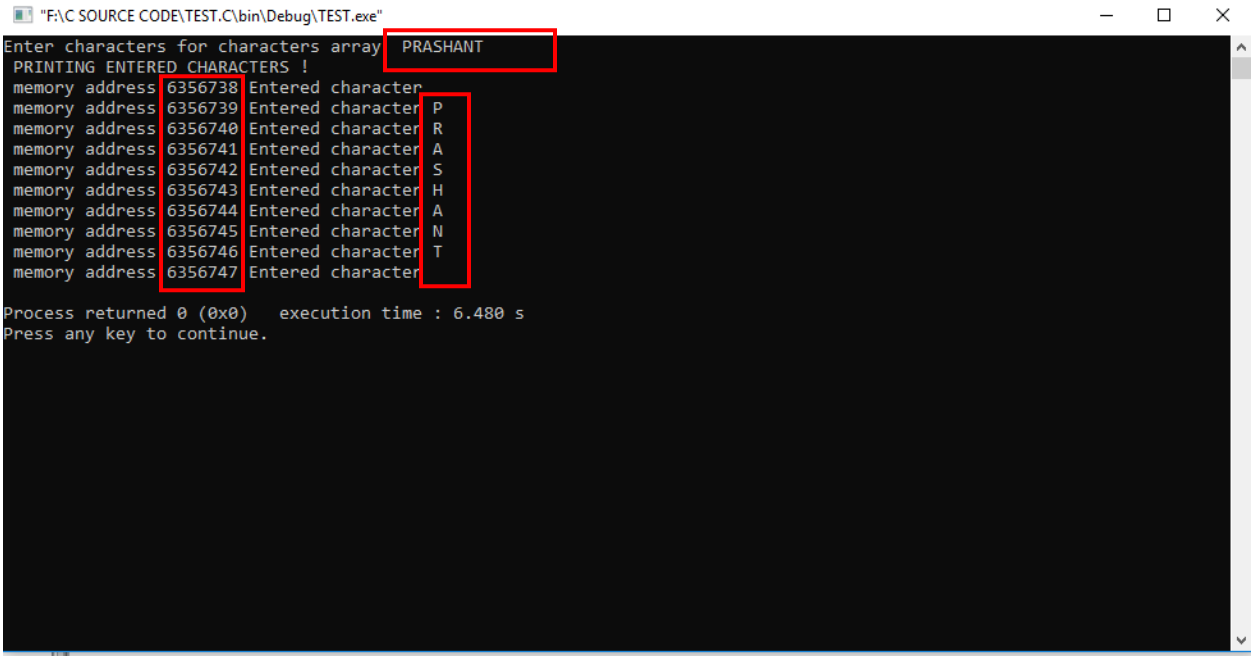
Accessing character array :- >> We can access the character array in two ways . The first way is with the help of for loop and the second one is with the use of format specifier " %s " .

A] So let's understand how to declare and access character array with for loop with the help of following example .

```
/*  Write a c program to declare and access character array with for loop .

    Author :- Prashant Shinde   */


#include < stdio.h >

#include < conio.h >
```

```
#include <stdlib.h>


int main ()
{
  char c[10];
  int x;
  system ("cls");
  printf ("Enter characters for characters array");
  for (x = 0; x < 10; x ++)
    {
  scanf( "%c", &c [x]);
     }
  printf ("PRINTING ENTERED CHARACTERS !");
  for (x = 0; x < 10 ; x++)
    {
  printf ("\n memory address %u Entered character %c", &c[x], c[x]);
  }
  return 0;
}
```

Above is the basic program to declare and access character array . There is no difference here in between previous one dimensional array and character array . so instead of explaining it's working , please observe the following output and you will understand it automatically on your own . Each character variable in the output occupies 1 byte and total of 10 bytes , because character variable needs one byte to store data inside the computers memory .



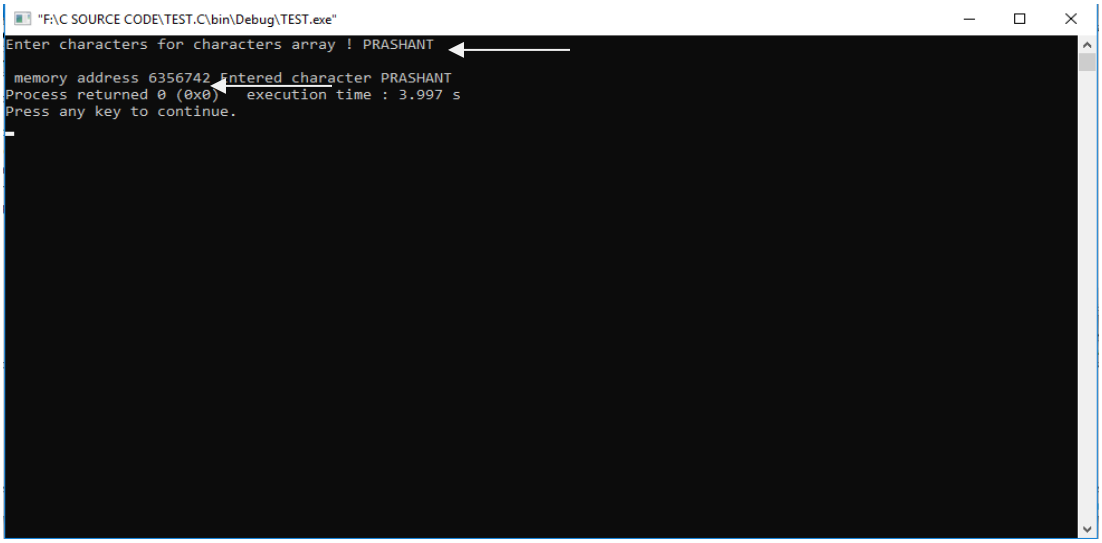
In the above output you can see I have entered name " PRASHANT " of 8 characters . after that in the output each character is stored to a different memory location , with a difference of one byte .

B] Now let's see how to use " %s " format specifier to access character array . So let's try the same example shown above by using " %s " format specifier.

```
/*  Write a c program to declare and access character array with for loop .
    Author :- Prashant Shinde  */

#include< stdio.h >
#include< conio.h >
#include < stdlib.h >
int main ()
{
```

```
  char c [10] ;
  system ("cls") ;
  printf(" Enter characters for characters array ! ") ;
  scanf(" %s ", c) ;
  printf(" \n memory address %u Entered character %s ", & c , c) ;
  return 0 ;
}
```

Observe the example above , how we have used " %s " format specifier . You don't need to use & before character array " c " . we have declared an character array of size 10 . So 10 bytes are reserved inside the computers memory . So suppose i have entered my name " PRASHANT " , then it will occupy first 8 bytes to store the name because it is 8 character long . Following is the output of above program . The characters are stored in same way as when we use "%c" inside the computers memory . But when you use " %s " , then compiler will automatically inserts null terminator character at the end of our characters which we have entered . The null terminator character is denoted by '\0' . This means our characters are ended here . It determines end of the characters or you can also call our bunch of characters as " string " .



ASSIGNMENT :- >> Write a c program to declare and access character array using for loop and %s format specifier .

Character array initialization :- >> As we have initialized one dimensional array above , we can also initialize character array in c . We can initialize character array in two way , the first one is directly initializing them to a string inside the double quote as shown below in first example or we can initialize them to a single character one by one inside a single quote separated by a comma as shown in second example below .

Syntax :- >> Data_type array_name[size] = " CHARACTERS " ;

OR

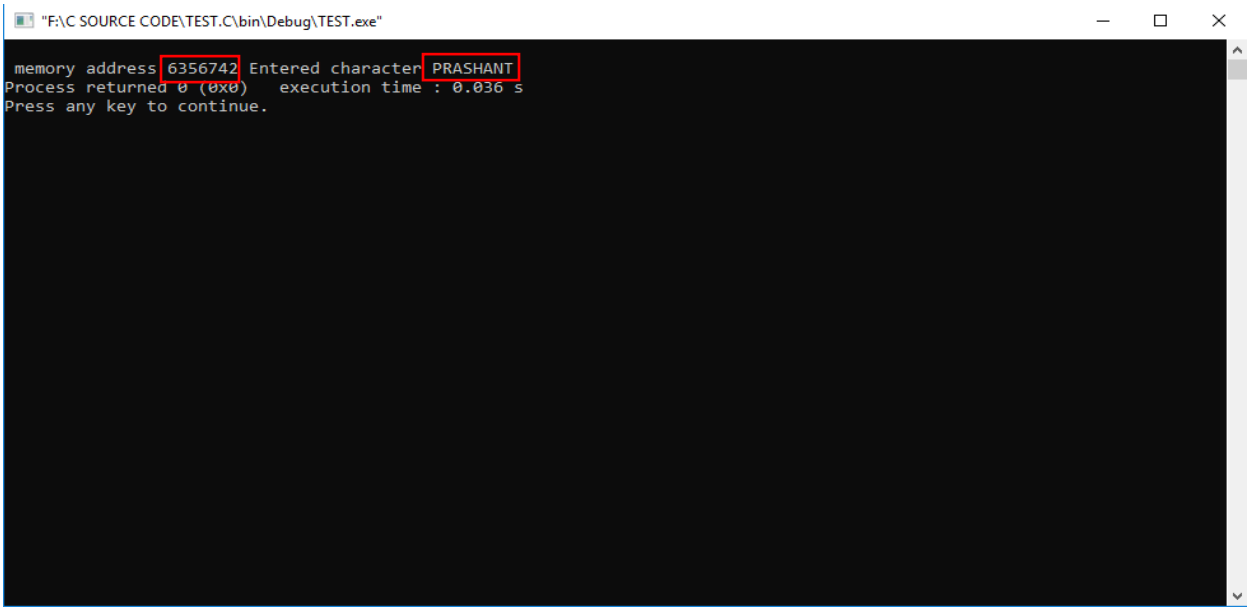
Data_type array_name[size] = { 'C' , ' H ' , ' A ' , ' R ' , ' A ' , 'C' , ' T' , ' E ' , ' R ' , 'S' } ;

Now let's understand how to initialize character array with the help of an practical example .

```
/*  Write a c program to initialize character array .
    Author :- Prashant Shinde  */
#include< stdio.h >
#include< conio.h >
#include < stdlib.h >

int main ()
{
  char c [10 ] = " PRASHANT " ;
```

```
  system ( " cls " ) ;

  printf ( " \n memory address %u Entered character %s ",&c ,c);

  return 0;

}
```

Following is the output of above program , where you can see the memory address of first character " P " out of initialized characters , and next you can see the name which is initialized .



ASSIGNMENT :- >> write a c program to initialize an array of 30 character and initialize it to a name " C Programming for beginners " and print it to computer screen .

KEY POINTS :->>

- An array is a group of similar type of data elements .
- An array can be of any data type such as int , char or float etc .
- Every array element always start with zero and ends with less than one by array size .
- An array occupies continuous memory locations inside the computers memory .
- The size of array elements always depends upon the data type used .
- An array can one one dimensional , two dimensional or multidimensional .
- You can declare an empty array as int array [ ] , when you don't mension the array size then it will adjust size of array depending upon number of data you store . Suppose you have entered 5 values to an empty integer array then the array size will become 5 automatically .

8. STRING HANDLING

Before actually starting to learn , how to handle strings , first of all let's understand what is a string , a string is nothing but group of characters . When we use a single character like " P " ," S " then it is called as the " character " & when we create group of characters like " PRASHANT " then it is called as " string ".

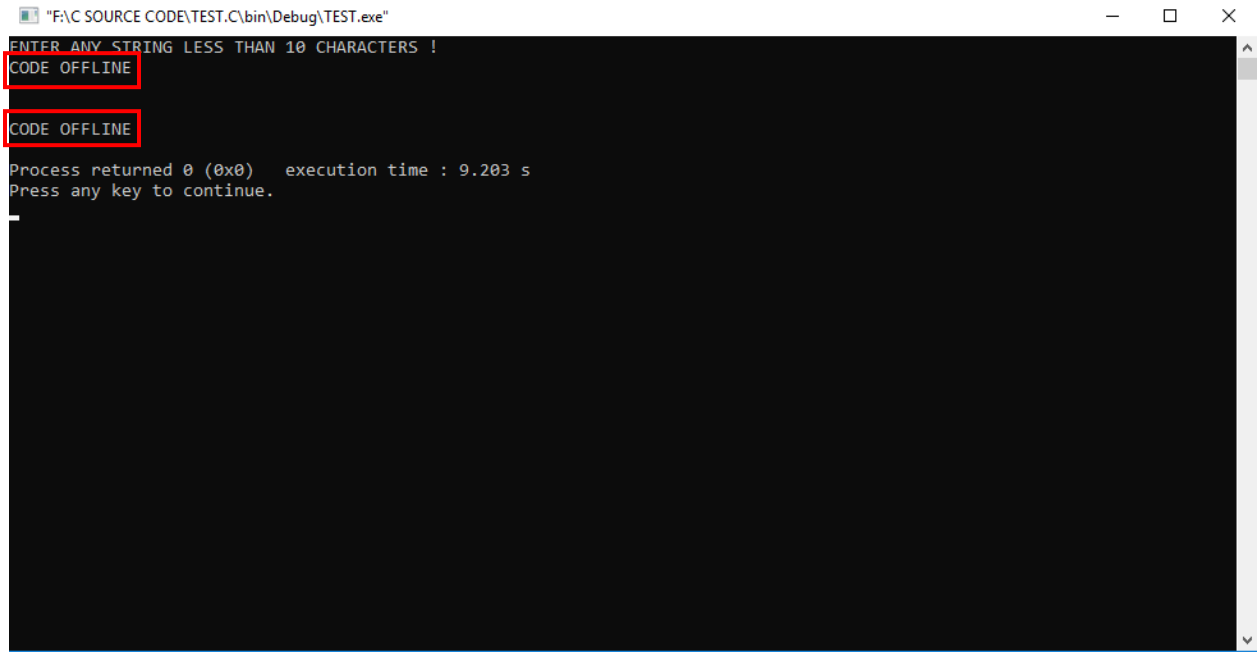
Until now we have learned how to receive input and output with the help of printf() and scanf() function . In this section we gonna talk about different functions to handle strings to perform input and output in c .So there are two functions i.e. gets() and puts() to receive a string from keyboard and to put a string to the output respectively . These are the functions which gonna help us in string handling .

So let's see how to use gets() & puts() functions to receive and print the string from keyboard to output . To work with these functions you need to declare character array . so let's see a practical example with it's working .

```
/*  Write a c program to understand working of gets() & puts() functions .
    Author :- Prashant Shinde  */
#include < stdio.h >
#include < conio.h >
#include  < stdlib.h >

int main ( )
{
  char c [ 10 ] ;
  system( " cls " ) ;
  puts( " ENTER ANY STRING LESS THAN 10 CHARACTERS ! " ) ;
  gets ( c ) ;
```

```
  puts ("\n");
  puts (c);
  return 0;
}
```

Now let's understand how it works , first of all the program control enters in to main( ) function and creates an array of 10 characters with name " c " . after that it will clear the computer screen and due to puts() function , it will print the message "ENTER ANY STRING LESS THAN 10 CHARACTERS !" on computer screen . To display any message on computer screen you just need to add any message inside the double quote inside the puts() function and it will display that message on computer screen . after printing that message our gets() function will start to work . whatever input we provide via gets() , it will receive that input and store it to our array char c [10] and in the next line we have set " \n " to get in to new line and then used puts(c) to display the inserted message on computer screen . Whenever you want to put the input string on computer screen then just place the character array variable inside the puts( ) function . following is the output of above program where it displays the input string from keyboard along with same string displayed with puts() .



ASSIGNMENT :- >> Write a c program with the help of gets() and puts() function to print your full name to the computer screen .

Ok above we have learned how to perform input and output with the help of gets() & puts() functions & i hope you have practiced the assignment . Now we have the basic understanding of what is a string . In c there are different functions are provided to handle these strings . To use these functions you need to add header file " string.h " in your program . Each string handling function provided has it's own use according to condition or requirement of the program . So let's see them one by one .

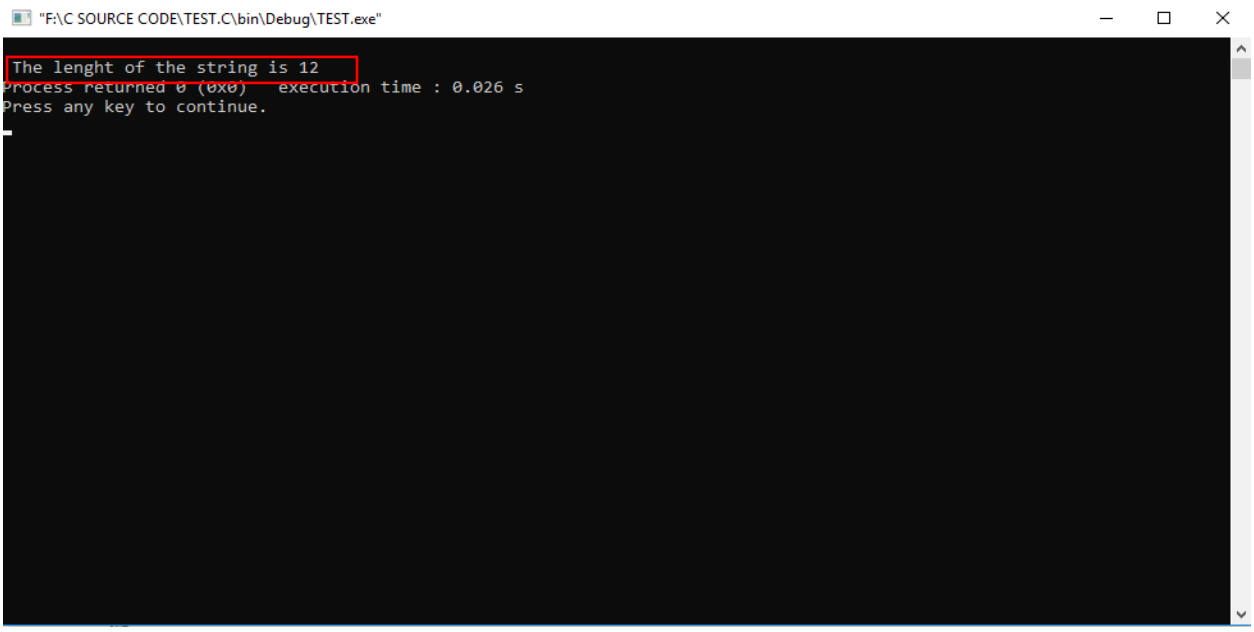
A] strlen() function :- >> This function is used to count the length of the string . In programming there are some situations where you need to count the length of the string , so strlen() function will help us to count the length of the string . Following is the syntax to use this function .

Syntax :- >> int var = strlen (string_variable) ;

When you pass a string variable (character array) via strlen function , then it will count the length of the string and this function returns that number and we store that number to a variable . Let's understand this with a practical example shown below .

```
/* Write a c program to understand strlen function .

   Author :- Prashant Shinde .  */

#include < stdio.h >

#include < conio.h >

#include < string.h >

#include  < stdlib.h >


int main ( )

{

  char c [ 15 ] = " CODE OFFLINE " ;
```

```
  int length ;
  system( " cls " ) ;
  length = strlen ( c ) ;
  printf ( " \n The lenght of the string is %d " , length ) ;
  return 0 ;
}
```



Now let's understand how this program works step by step . First of all the program control enters into main () function and creates an array of size 15 character length long and initialize it to name " CODE OFFLINE " . Next it creates an integer variable length and then clears computer screen . after that we have passed character array c via strlen() function as " length = strlen (c) " , So here strlen ( ) function will count the length of the string and apply the result to length variable . The strlen function counts the characters present inside any string including blank space and return that number and we can get that number with assignment operator = as shown above . So by observing above output you can easily understand how it works ." CODE OFFLINE " actually contains 11 characters but it has printed 12 to the output because there is blank space in between CODE and OFFLINE , so that's why it has printed 12 to the output . So after counting the length of character array " c " it will apply it integer variable length . after that it goes to printf() function and prints that

result . One important point that you need to include is , you need to add the header file " string.h " to make this program works .

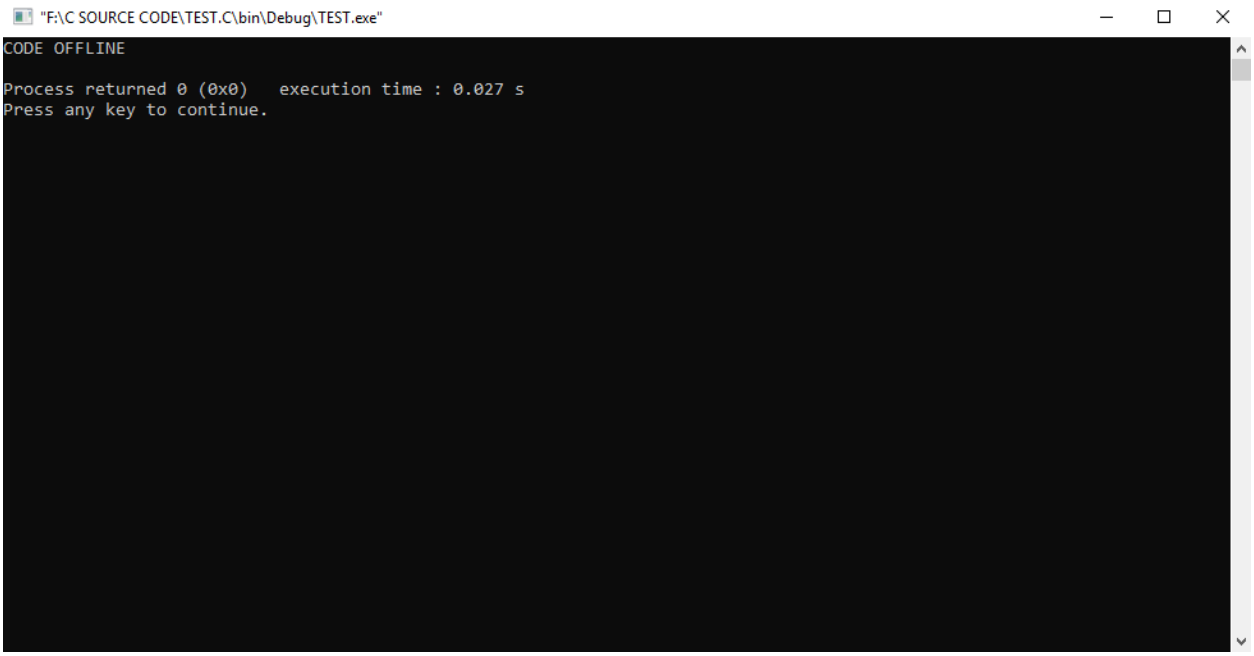
ASSIGNMENT :- >> Declare a character array of length 20 and initialize it to your mobile number and print the length of the string to computer screen .

B] strupr () function :- >> This function is used to convert the string to upper case . If you have a string in lower case then this function will convert that string into upper case . To use this function you need to pass the string variable via this function . In short you can also say this function converts small letters into capital letters . Let's understand this with a practical example .

```
/* Write a program to understand strup() function .
   Author :- Prashant Shinde */


 #include < stdio.h >
#include < conio.h >
#include < string.h >
#include  < stdlib.h >
int  main ( )
{
  char c [ 15 ] = " code offline " ;
  system ( " cls " ) ;
  strupr ( c ) ;
  puts ( c ) ;
  return 0 ;
```

}

By observing the above program you can easily understand how strupr() function works . In the above program we have declared a character variable " c " and initialized it to a string " code offline" . The string present there is in small letters . after that we have passed that character array via string upper function as strupr(c) , so here the small letters will be automatically get converted into capital letters and then we have used puts() function to print those capital letters to output screen as puts(c) . In the following output you can see the string is converted into capital letters or upper case .

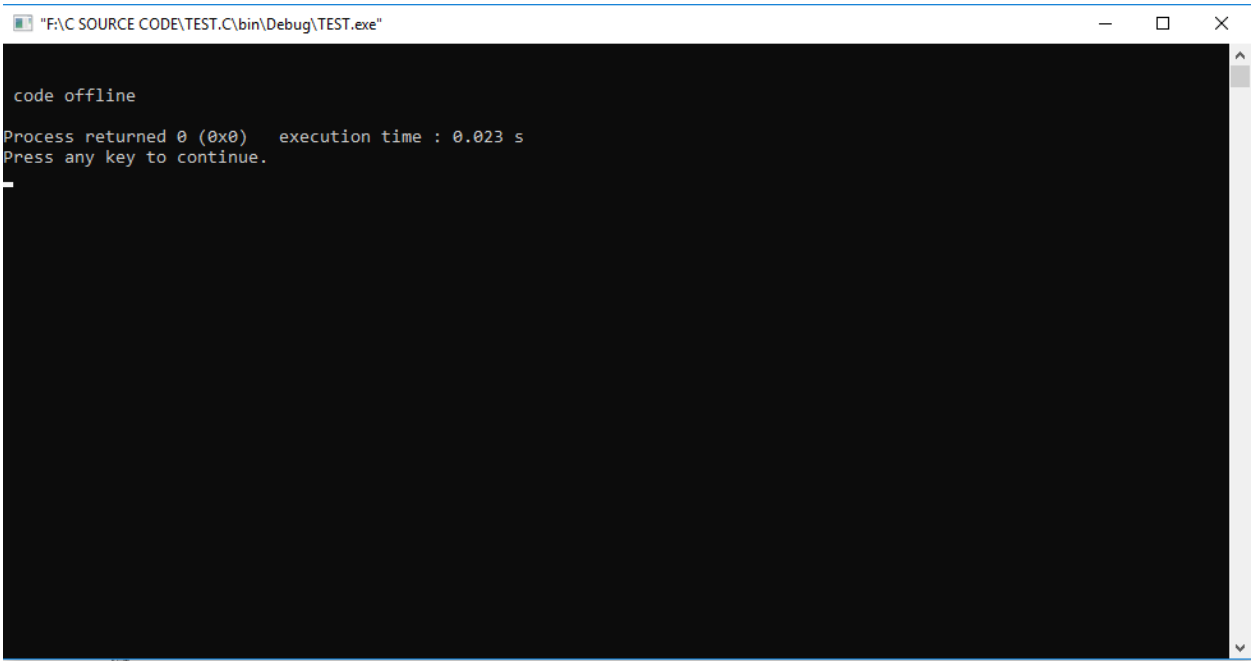


ASSIGNMENT :- >> Declare and initialize a character variable to your country name and convert it in to capital letters with the help of strupr ( ) function .

C] strlwr ( ) function :- >> strlwr ( ) function is used to convert a string from upper case into lower case . The syntax is same as we use for strupr() function . So let's understand the working of this function with an practical example .

```
/*  Write a c program to understand the working of strlwr ( ) function .
    Author :- Prashant Shinde  */
#include < stdio.h >
#include < conio.h >
#include < string.h >
#include  < stdlib.h >
int main ( )
{
  char c [ 15  ] = " CODE OFFLINE " ;
  system ( " cls " ) ;
  strlwr ( c ) ;
  puts ( c ) ;
  return 0 ;
}
```

In the above program we have declared a character array of size 15 & initialized it to a name " CODE OFFLINE " . after that we have passed that character array variable through strlwr() function and finally printed it on computer screen with puts(c) function . So the above program will convert the upper case name " CODE OFFLINE" into small letters as shown in the following output .



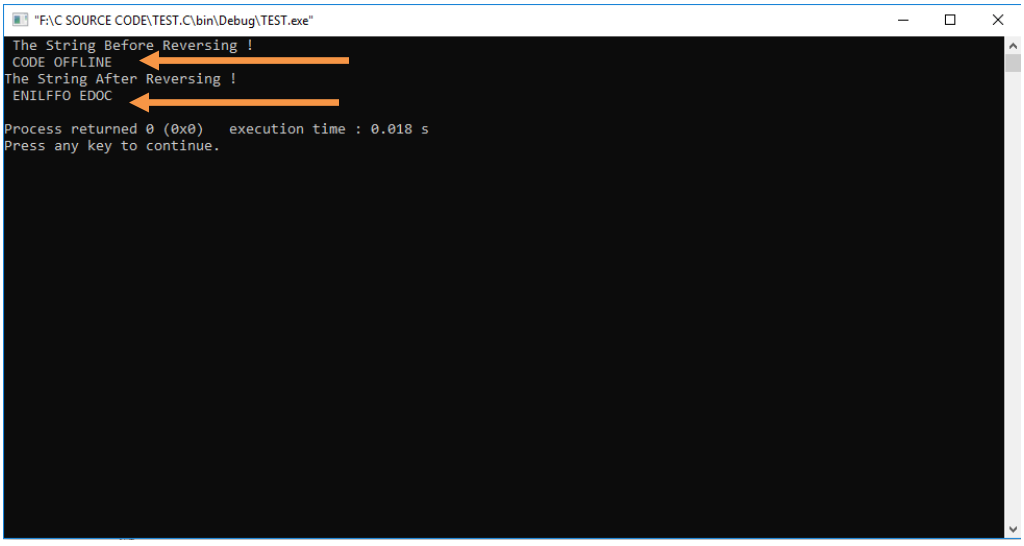
ASSIGNMENT :- >> Write a c program to convert a string " C PROGRAMMING " in to small letters .

D] Strrev ( ) function :- >> This function is used to reverse a string . What ever may be the string , either initialized by us or inserted from keyboard . This function will reverse the string and prints it to the output . So let's understand the working this function with the help of an practical example shown below .

```
/*  Write a c program to understand the working of strrev() function .

    Author :- Prashant Shinde  */

#include < stdio.h >

#include < conio.h >

#include < string.h >

#include  < stdlib.h >
```

```
int main ()
{
  char c [15] = " CODE OFFLINE ";
  system (" cls ");
  puts(" The String Before Reversing ! ");
  puts (c);
  puts(" The String After Reversing ! ");
  strrev (c);
  puts (c);
  return 0;
}
```

We have used the same example that we had used for strlwr() function , here we have printed two outputs to the computer screen . The first one before reversing the string and second one after reversing the string . To reverse the string you just need to pass the string via strrev ( ) function . following output will show you how strrev() function reverses the string .



ASSIGNMENT :- >> Write a c program to receive your name from keyboard and show your name in reverse order with the help of strrev ( ) function .

E] Strcpy ( ) function :- >> This function is used to copy a string from one location to the another location . Following is the basic syntax to decalre this function .

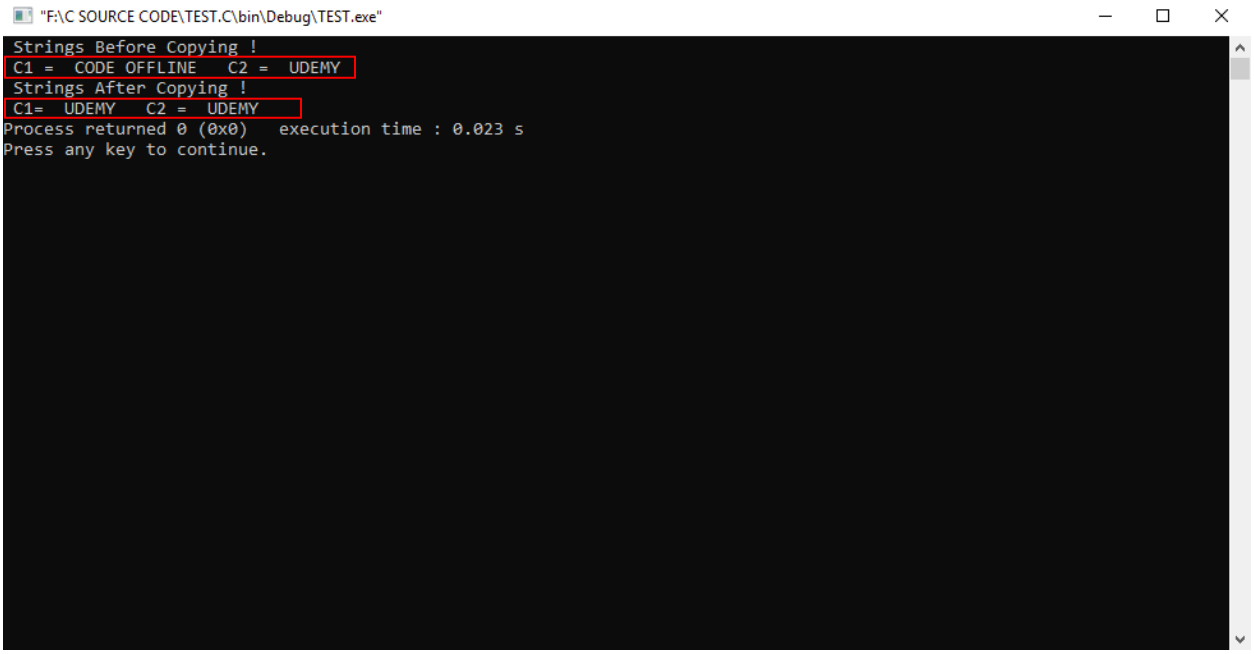
Syntax :- >> strcpy (string 1, string 2) ;

In this function we need to pass two arguments string 1 , string 2 respectively . When we pass these two arguments then the string 1 is get replaced with string 2 . Means you are copying string 2 at memory location of string 1 here . So let's understand this with a practical example as shown below .

```
/*  Write a c program to understand the working of strcpy() function .

    Author :- Prashant Shinde  */

#include < stdio.h >

#include < conio.h >

#include < string.h >

#include  < stdlib.h >

int main ()

{

  char c1 [ 15 ] = " CODE OFFLINE " ;

  char c2 [ 15 ] = " UDEMY  " ;

  system ( " cls " ) ;

  puts ( " Strings Before Copying ! " ) ;

  printf ( " C1 = %s  C2 = %s " , c1 , c2 ) ;

  strcpy ( c1 , c2 ) ;
```

```
  puts( " Strings After Copying ! " ) ;

  printf ( " C1= %s  C2 = %s ",  c1 , c2 ) ;

  return 0;

}
```

In our program above we have declared two character arrays as C1,C2 respectively & initialized them to " CODE OFFLINE " & " UDEMY " , So due to strcpy() function the string at C1 will get replaced with string at C2 . By observing the following output you can easily understand how this function works .



ASSIGNMENT :- >> Write a c program to copy string " HACKER " to " CRACKER " , make sure both the strings are already initialized to character arrays .

F] Strcat ( ) function :- >> This function is used to concatenates two strings or you can also say that this function is used to add one string at the end of another string . Following is the Basic syntax to declare strcat ( ) function .

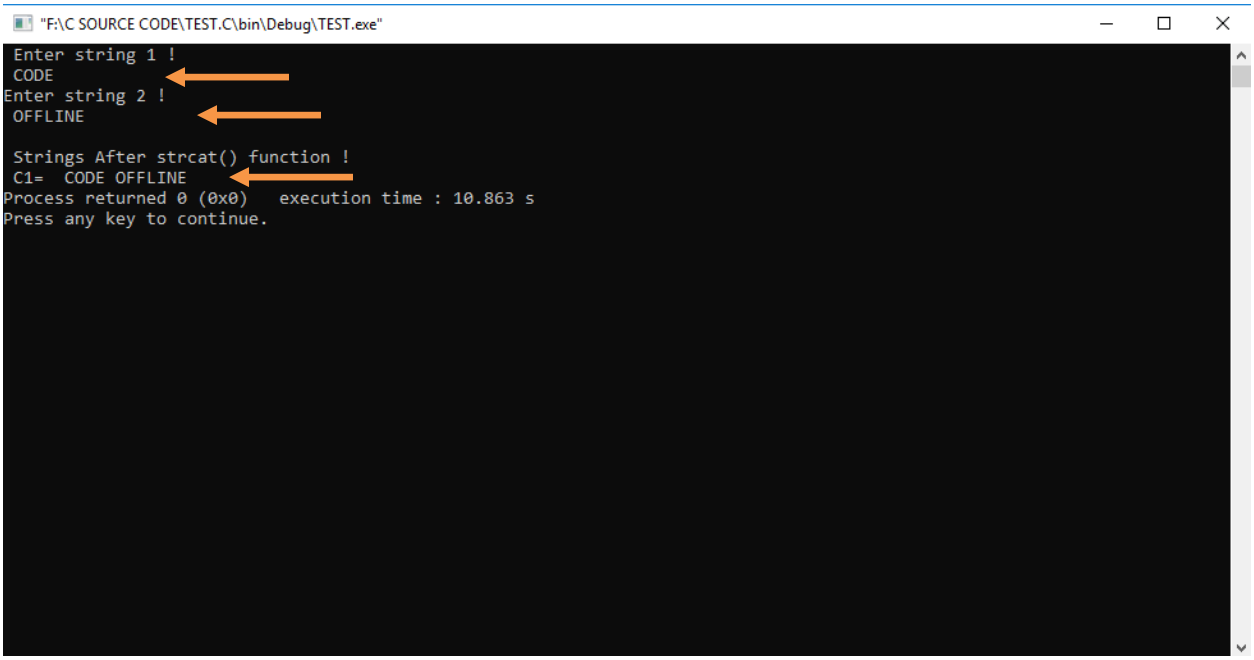
Syntax :- >> strcat (string1 , string2) ;

When you pass string 1 and string 2 via strcat( ) function then you will find the string 2 is get added at the end of string 1 in the output . Let's understand this with the help of a practical example .

```
/*  Write a c program to understand strcat() function .

    Author :- Prashant Shinde  */

#include < stdio.h >

#include < conio.h >

#include < string.h >

#include  < stdlib.h >


int main ( )

{

  char c1 [ 15 ] ;

  char c2 [ 15 ] ;

  system ( "cls " ) ;

  puts ( " Enter string 1 !" ) ;

  gets ( c1 ) ;

  puts ( " Enter string 2 ! " ) ;

  gets ( c2 ) ;

  strcat ( c1 ,c2 ) ;

  puts( "\n Strings After strcat() function ! " ) ;

  printf( " C1= %s " , c1 ) ;
```

```
  return 0 ;
}
```

In the above program we have declared two character arrays c1 , c2 respectively . after that we have received two strings for c1 and c2 separately and respectively . after receiving two strings we have passed c1 and c2 via strcat ( ) function as strcat(c1,c2) . after adding string 2 at the end of string1 , we can set it's final output with printf() function as C 1 . following output will explain everything to you .



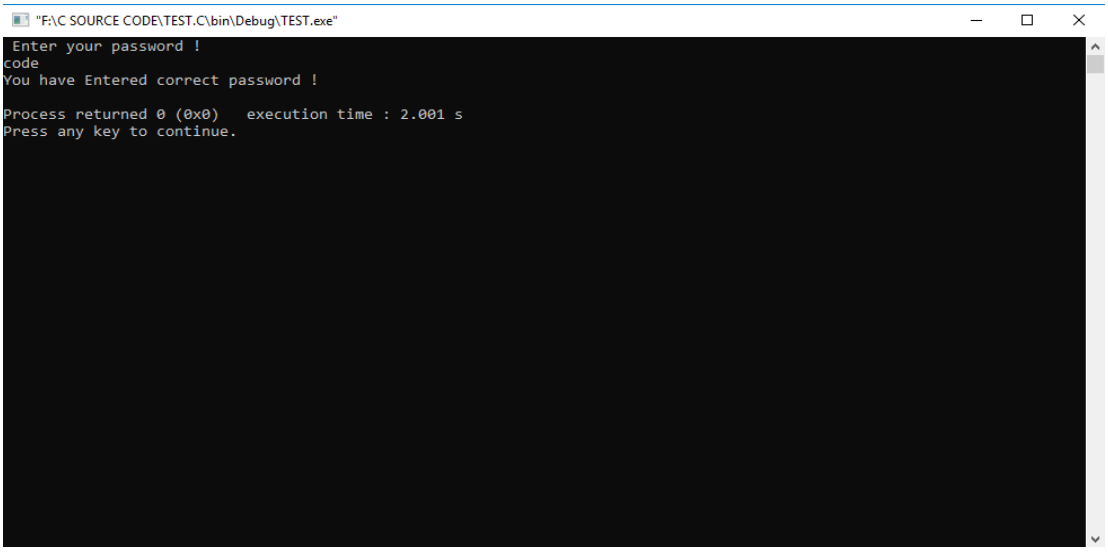
ASSIGNMENT :- >> Write a c program to receive two strings " CODE " & " HACKER " from keyboard separately & combine them with strcat() function and print it's output with printf() function .

G] strcmp () function :- >> This function is used to compare two strings with each other . This function takes two arguments string 1 & string 2 respectively . If both strings matches with each other then it returns 0 and if strings doesn't match with each other then it gives a

non-zero value . strcmp() function is normally used while writing password compare code . So let's understand how this function works with the help of an practical example .

```
/* Write a c program to understand the working of strcmp() function .
    Author :- Prashant Shinde */
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include <stdlib.h>

int main()
{
  char c1 [ 15 ] = " code " ;
  char c2 [ 15 ] ;
  int result ;
  system( " cls " ) ;
  puts( " Enter your password !" ) ;
  gets( c2 ) ;
  result = strcmp( c1 , c2 ) ;
  if( result == 0 )
    puts( " You have Entered correct password ! " ) ;
  else
    puts( " You have Entered Wrong password ! " ) ;
  return 0 ;
}
```

Above is the most interesting program which will help us to find out whether the password entered from keyboard is correct or not . In this example first of all we have declared two character arrays c1 & c2 respectively . c1 is initialized to a string " code " which is our password in this program and c2 is kept empty to receive password from keyboard . The variable " result " is declared to store the result of strcmp() function . The strcmp() function return 0 if two strings matches and if doesn't then it gives any non zero value . so that we can store the result of strcmp() to result variable . After that cleared computer screen with system(" cls ") ; . Then received a password from keyboard with gets( ) function and stored it to character variable c2 as gets(c2) . after that passed both the character array variables via strcmp ( ) function as strcmp (c1 , c2) . Then next we have stored the result of strcmp() to result variable as result = strcmp (c1 , c2) . So if we enter correct password from keyboard then it gets matched with string initialized to c 1 & strcmp() will return 0 . so condition in if block becomes true i.e. if (result==0) and the computer will print the message on screen " You have Entered Correct Password ! " & if you enter wrong password then it will return a non-zero value and condition of if block will become false here & the statement within else block will get executed and it will print message within else block as " You have Entered Wrong password ! " . I hope all of you have get the basic understanding of how this code works . You can easily understand this by observing the following output .



ASSIGNMENT :- >> Write a c program to test whether the entered username is correct or not , the username will be “Prashant ”.

KEY POINTS :- >>

- To use any of the string handling function you should include “ string.h ” header file in your program .
- String handling functions are useful for writing some password protection code .
- The strcmp () function returns zero if the entered strings are similar otherwise it will returns non-zero value .
- With strlen() function you can limit the password length for your password protection program .

9. FUNCTION .

Until now we have learned how to use the already existing functions in c library ,like we have used printf ( ) function to show output . The definition of this function is defined inside stdio.h header file . In c library there are many functions are available & to use those functions you need to include a particular header file according that function . For example to use printf() function you need to include stdio.h header file , to use graphics related functions you need to include graphics.h header file , to use math related functions you need to include math.h header file and so on . So all the functions which are already exists in c are called as " library functions " . Like these library functions we can create our own function in c . By creating our own function we can avoid the writing of same code again and again . Suppose there is a code to add , divide & multiply two numbers and all this code is in sequential order and you have performed addition of two numbers and after that you have performed division . After performing two operations , if you want to perform the same operation of addition of two numbers again in c then you should write the same code after performing addition and division or you need to restart the program . After adding the same code the code will become too large . So to avoid this problem the concept of function is introduced in c . You can write that code inside your own function once & then call it again & again whenever you want . Those functions which are defined by the users are called as " User defined functions " & those functions which are already exists are called as " Library functions ".

There are 3 steps involved in creating a user defined function . The first one is function declaration , second one is function definition and last one is calling function . So let's see theme one by one .

I] Function Declaration :- >> In this step you need to define a function name along with it's return type . Following is the basic syntax to declare user defined function .

Syntax :- >> return_type function _name ( ) ;

Above is the basic syntax to declare user defined function . Where you can set return type as any data type such as int , float ,double . Then after typing a space you can write any name for your function that you want . After that you need to type opening & closing brackets ( ) and semicolon ; to close the function declaration . Following is the basic example of function declaration .

Example :- >> int addition ( ) ;

II] Function Definition :- >> In function definition you need to type the return_type , then function name and then opening & closing brackets () and after that opening & closing curly brackets { } and inside those opening & closing curly brackets you need to type the code which we wanting to execute via our function . Following is syntax to define function definition .

```
Syntax :- >>  return_type  function_name ( )
                {
                    code to execute ;
                }
```

```
Example :- >>  int addition ( )
                {
                    int a , b , c ;
            printf ( " Enter value for a & b respectively " ) ;
                scanf ( " %d %d " , & a , & b ) ;
             c = a + b ;
            printf ( " The addition of two numbers is = %d " , c ) ;
                                                    }
```

Above is an example of function definition .The most important point to write a function definition is that you need to type it outside the main ( ) function . so this is the way to write a function definition .

III] Calling function : ->> After declaring and writing function definition , now it's time to call the function . Without calling a function you can't execute the code that we have written inside our function definition . so to execute you function definition it is more important to call it . Following is the basic syntax to call a function .

```
Syntax :- >>  function_name ( ) ;
```

You just need to type function name along with opening and closing brackets ( ) , ending with semicolon . This is the simple way to call a function whenever you want . You can call a function any number of times that you want . You just need to call the function within main function .

Example :- >> addition ( ) ;

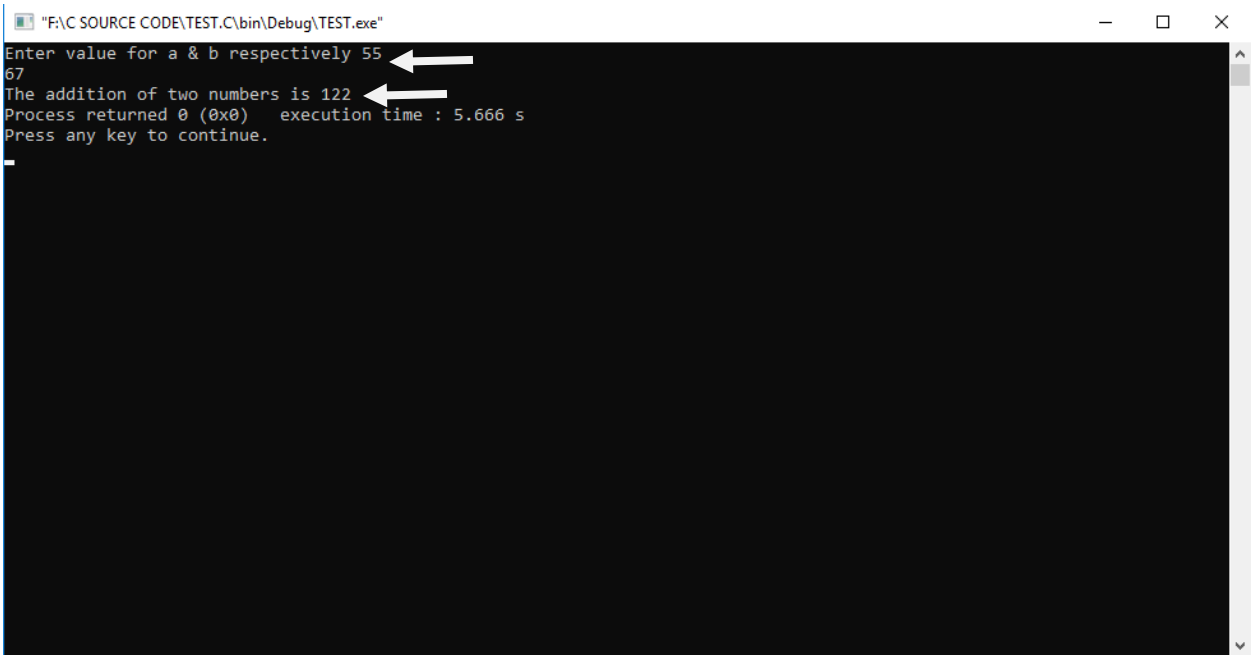
Now after understanding all the format of userdefined function let's see how to create our own function with the help of practical example .

```
/*  Write a  c  program to understand userdefined function

    Author :- Prashant Shinde    */

#include < stdio.h >

#include < conio.h >

#include  < stdlib.h >


/*  function declaration outside main function  */

void addition ( ) ;

int main ( )

{

  system ( " cls " ) ;


/* Calling a function within main function */

  addition ( ) ;

  return 0 ;

}

/*   Function Definition outside the main () function   */

 void addition ( )
```

```
{
  int a , b , c ;
  printf ( "\n Enter value for a & b respectively " ) ;
  scanf ( " %d  %d " , &a , &b ) ;
  c = a + b ;
  printf ( "\n The addition of two numbers is  %d ",c ) ;
}
```

Above is a basic program to understand userdefined function . Before main ( ) function we have declared our function as void addition ( ) ; , so first of all let's understand what is void , void is a keyword introduced in c , void means empty or nothing . Because i don't want to return anything from our function that's why i have declared it as void . after declaring the function , I have called it within the main ( ) function , after system(" cls ") ; . after that we have written the function definition outside the main function as void addition ( ) { } .

Now let's understand how this program works , first of all the program control enters into our main( ) function and executes system(" cls ") ; . after that our function addition( ) will get called . after calling the function the program control will first checks the format of " declared function and function definition is correct or not " . First of all it will check whether the return type , function name of declared function & function definition is correct or not . If it's correct then it will get into function definition & executes the code within it , and after exeuting the code it will get back to the location from where it was called ,that means it will get inside the main function . So this is how the function concept works . Following is the output of above program . In the following output we have entered two values 55 and 67 respectively & then at next line displayed the addition as 122 .



Ok after understanding how function works , let's understand the same with very simple example . In the following example I am going to define 2 functions and gonna call them many times , so that you can understand how function works in more clearly .

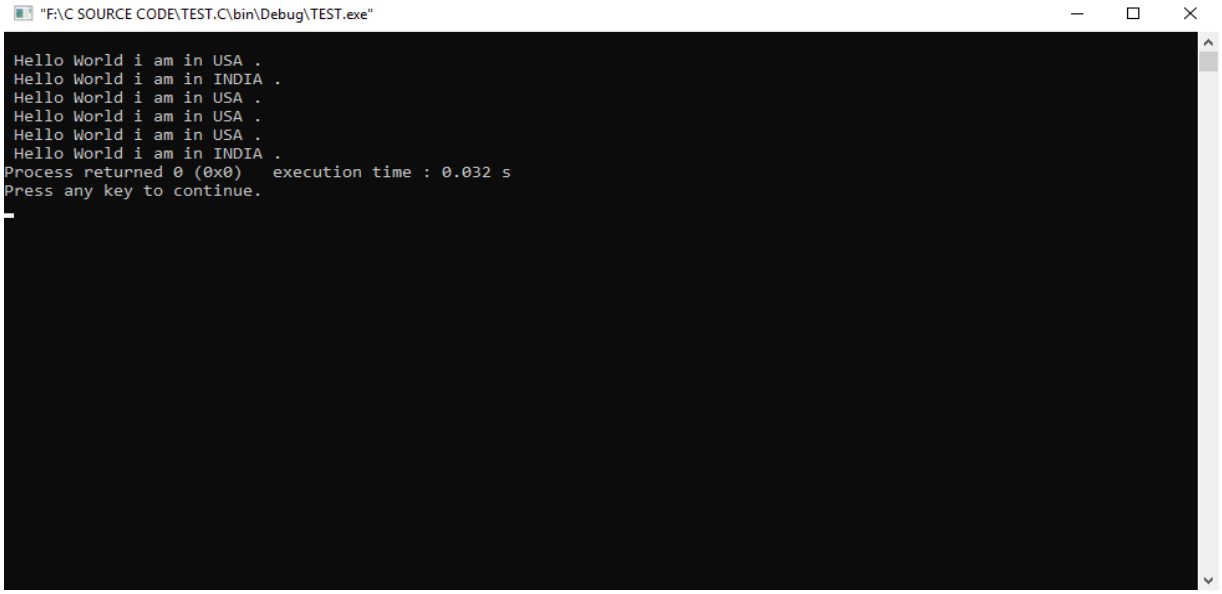
/ * Write a c program which will display 2 messages " Hello world i am in U.S.A " & " Hello World i am in INDIA " after calling two different functions one by one .

Author : - Prashant Shinde */

```
# include < stdio.h >

# include < conio.h >

# include  < stdlib.h >

void function 1 ( ) ;

void function 2 ( ) ;

int main()

{

  system ( " cls " ) ;
```

```
  function1 ( ) ;
  function2 ( ) ;
  function1 ( ) ;
  function1 ( ) ;
  function1 ( ) ;
  function2 ( ) ;
  return 0 ;
}
void function 1 ( )
{
   printf("\n Hello World i am in USA .");
 }
void function 2 ( )
{
   printf("\n Hello World i am in INDIA .");
}
```

In above program we have declared two functions as function 1( ) and function 2( ) . Where we gonna show only two different messages as " Hello World i am in USA " and " Hello World i am in INDIA " . We have called these functions six times . This will help you understand we can call our functions in any number of times that we want and in any order . By Observing the following output you will easily understand how it works .



ASSIGNMENT :- >> Write a c program to perform division & multiplication of two numbers entered via keyboard with two different functions as division( ) & multiplication ( ) .

KEY POINTS :- >>

- A function is such a great concept introduced in c which helps you to avoid repetition of code .
- Function concept helps you to increase readability of code .
- Program modification becomes easier due to function .
- Function helps us to divide a complex program into simple steps .

10. STRUCTURE & UNION IN C .

Until now we have learned how to create vriables based on our choice , like we have created integer variable to store integer values , we have created character variable to store characters . But is there any way to store any type of data to the same variable , means you can store integer , character & float data to a single variable , yes there is a way introduced in c programming where we can store different types of data such as character , integer or float to the same variable . We can achieve this by declaring a structure variable in c .

Ok then what is structure ,structure is a userdefined data type where we can store different data elements together in a single structure variable .so there are three steps in using a structure the first one is declaring a structure , then second declaring a structure variable & third one is accessing a structure variable .

I] Declaring a structure : ->> This is the first step in creating a user defined data type in c . You can define a structure either outside the main( ) function or inside the main ( ) at the beginning . Following is the basic syntax to declare a structure a variable .

```
Syntax :- >>     struct  structure_name
                 {
                     structure member 1 ;
                     structure member 2 ;
                     structure member 3 ;
                     structure member 4 ;
                   } ;
```

above is the basic syntax to declare a structure . at the beginning you need to declare a keyword " struct " then you need to define a structure name & after that in between the opening & closing curly brackets { } you need to define structure members based on your choice or program requirements . These structure members are nothing but different

types of data types along with variable names . after that at the end of closing curly bracket you need to add semicolon . Following is the basic example of structure declaration . In the following example i have declared a structure as “student ” where it contains three structure members such as name , roll_no , marks to store the basic information of student .

Example :- >>

```
struct student
{
  char name [30] ;
  int roll_no ;
  float marks ;
} ;
```

above is an example of declaring a structure but declaring a structure is not enough . you need to declare a variable to access those structure members . So after this let’s see how to declare a structure variable .

Declaring a structure variable :->> There are two ways to declare a structure variable in c . You can either declare a structure variable at the end of closing curly bracket of structure or with a separate format .

Method I :- >>

```
struct student
{
char name [30] ;
int roll_no ;
float marks ;
} s ;  /* Here S is a structure variable */
```

OR

Method II :- >>

```
struct student S .
```

Above are the two methods of declaring a structure variable . when you declare a structure variable then it reserves a space inside a computers memory .

ACCESSING STRUCTURE MEMBERS :->> After declaring a structure variable now let's see how to access the structure members via structure variable . To access the structure members you need to use dot [.] operator . Following is the basic syntax to declare a structure variable .

Syntax :- >> structure_variable . structure_member ;

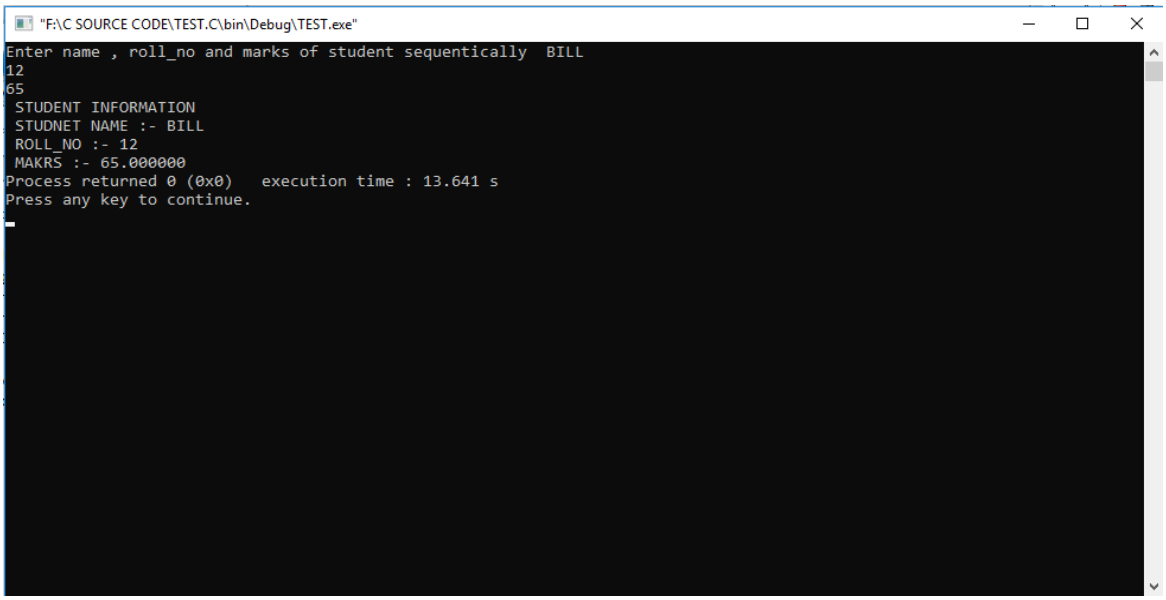
Example :- >> s.name , s.roll_no , s.marks ;

Ok , so after understanding how to declare and access structure , let's see how to implement the above information practically .

```
/*  Write a c program to understand the concept of structure

    Author :- Prashant Shinde   */

#include < stdio.h >

#include < conio.h >

#include  < stdlib.h >

struct student

  {

  char name [ 30 ] ;

  int  roll_no ;

  float marks  ;

    } ;


int main ( )

{

  System ( " cls " ) ;
```

```
  struct student s;
printf("Enter name , roll_no and marks of student sequentially");
scanf("%s %d %f", &s.name, &s.roll_no, &s.marks);
printf("STUDENT INFORMATION");
printf("\n STUDNET NAME :- %s", s.name);
printf("\n ROLL_NO :- %d", s.roll_no);
printf("\n MAKRS :- %f", s.marks);
  return 0;
}
```

In above program we have declared a structure called as " student " outside the main ( ) function to store the basic information of students such as student name , roll_no & marks . To store this information we have declared structure member as name , roll_no & marks . after that inside the main ( ) function we have declared a structure variable as " struct student s " . After declaring the structure variable in the next line we have received the information of students from keyboard as ",&s.name,&s.roll_no,&s.marks " . After receiving the values at last we have printed them to the output with printf ( ) function one by one . Following is output of above program which will help you to understand how it works .



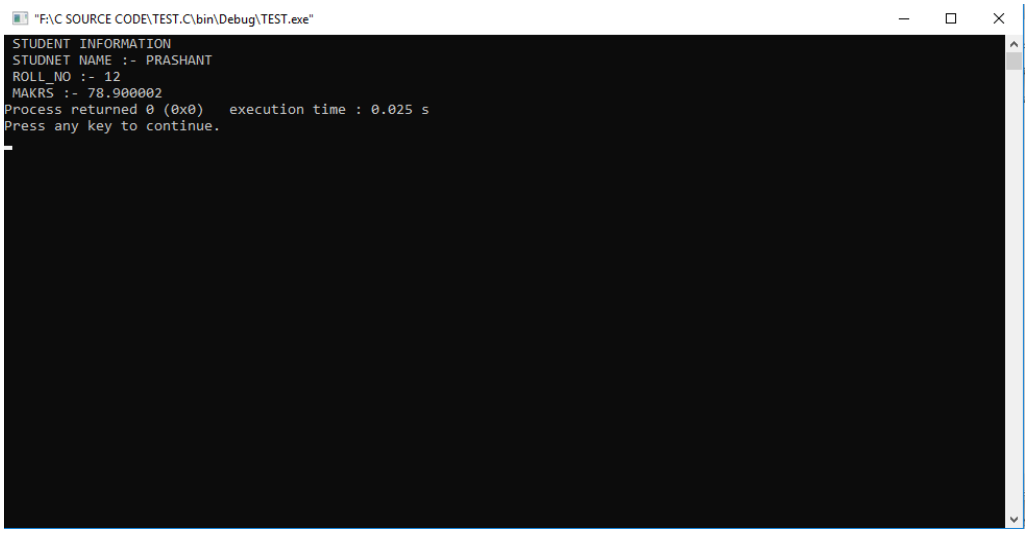
ASSIGNMENT :- >> Write a c program with the help of structure to store the basic information of an employee like name , age , salary .

Initialization of structure variable :- >> In the above program we have learned how to declare and access structure variable in c , now it's time to learn how to initialize a structure variable in c . Like a normal variable we can also initialize a structure variable . So let's understand how to initialize a structure variable practically . Following program is the modification of above program to understand the concept of initialization of structure variable .

```
/*  Write a c program to initialize a structure variable .
     Author :- Prashant Shinde     * /

#include < stdio.h >
#include < conio.h >
#include  < stdlib.h >
struct student
  {
  char name [ 30 ];
  int roll_no ;
  float marks ;
    };
int main ( )
```

```
{
  system ( " cls " ) ;


 struct student s = { " PRASHANT ", 12  , 78.9  } ;


 printf ( " STUDENT INFORMATION " ) ;

 printf ( " \n STUDNET NAME :- %s ", s.name ) ;

 printf( " \n ROLL_NO :- %d " , s.roll_no  ) ;

 printf( " \n MAKRS :- %f " , s.marks  ) ;

  return 0 ;

}
```

Now let's understand how it works . Here we have used the assignment operator to initialize the structure variable . After that inside the opening & closing curly brackets we have initialized the values in the same order in which we have declared the structure members . If you miss the order then you will get error while compiling the code . In the structure member first of all we have character array of size 30 called "name" then integer as roll_no and lastly float as marks . So that's why we have initialized the values in the same sequence . Following is the output of above program .



UNION :- >> After understanding the structure , now let's understand what is union & how it works . Union is a keyword introduced in c which is used to create userdefined data types . The format of declaring accessing the union is same as structure but there is one difference when it came to memory allocation . When you declare a structure variable then it occupies the memory based on addition of each data member present inside the computers memory . But in case of union , it occupies the memory based on biggest size of data member . So now let's understand how to declare & access the union . Following is the basic example of declare a union along with union variable .

Example :->>

```
union student
{
char name [ 30 ] ;
int roll_no ;
float marks ;
};
union student s ;
```

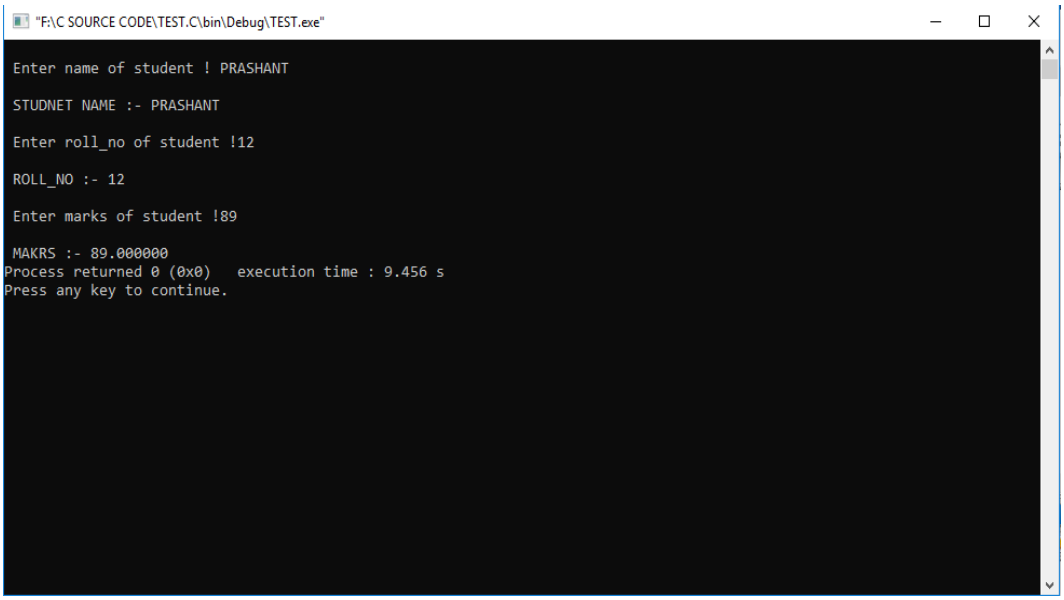
This is a basic example of declaring a union & union variable . Now let's understand how it occupies the memory inside the computer . Inside the above union , there are 3 data members inside it , amongst that char name [30] ; is the data member which has bigger size as compared to other data members . So when we declare the variable as " union student s " then the " S " variable will occupy the size of 30 bytes . Whenever we declare a variable with union then it will occupy the size which is maximum amongst the union members . Now let's understand the working of union variables with the help of a program .

```
/* Write a c program to understand the concept of union
   Author :- Prashant Shinde */

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

```
#include  < stdlib.h >

union student
  {
  char name[30];
  int roll_no;
  float marks ;
    };

int  main ( )
{
  system ( " cls " ) ;

 union student s ;
 printf (" \n Enter name of student ! " ) ;
 scanf ( " %s ", s.name  ) ;
 printf ( " \n STUDNET NAME :- %s ", s.name ) ;

 printf ( " \n\n Enter roll_no of student ! " ) ;
 scanf ( "%d ", &s . roll_no ) ;
 printf ( "\n ROLL_NO :- %d " , s . roll_no ) ;

 printf ( " \n\n Enter marks of student !" ) ;
 scanf ( " %f " ,  &s.name ) ;
```

```
printf (" \n MAKRS :- %f ",s.marks ) ;

 return 0 ;

}
```

Now let's understand how this program works . First of all the program control enters into our main fuction and clears the computer screen with system ("cls") function . after that creates a union variable " S " will have a of size 30 bytes , because amongst all the data members inside our union char name[30] is bigger in size that why the variable " S " will occupy 30 bytes inside the computers memory . After that you can see we have received the information of students separately , because if you try to insert the information in one single line then the memory will get corrupted & you will not get correct information to output . By observing the following output you can see how it works .



ASSIGNMENT :- >> Write a c program with the help of union to store the Basic information of an actor like his name , age & net worth & display it to the output .

KEY POINTS :- >>

- Structure concept helps us to create a hybrid variable with help of which we can store any type of data that we want such as int , char ,float .
- We can create more than one structure variable in c or in other words you can say we can create array of structure variables in c .
- We can also initialize the structure variable as we do with normal variable .
- Union also helps us to create hybrid variable to store different types of data .
- We can not store the data in one shot to union variable as we do with structure variable , if we try to store then our data will get corrupted .
- Both structure & union members are accessed with the help of dot operator .

11. POINTER

What is Pointer ?

Pointer is a special variable which is used to point to the memory address of another variable . Until now we have learned how to declare & use normal variable to store normal data like numbers or characters but now it's to learn how to store the memory address of another variable with a special variable called as " Pointer " .

Pointer is the most important as well as very interesting concept in c programming . Because you can directly access the memory address of another variable . In my computer course I have taught " How pointer can be used to create a computer virus ? " . You can see their i have created a computer virus called as " Screen killer " which will activate after running your program on turboc++ compiler & will start to eat your computer screen while you are working on it . If you are interested you can check the link in the last page of this book to get $ 190 worth of my course for just $ 10 for life time access on udemy .

Declaring & using pointer variable :- >>

So first of all let's see how to declare a pointer variable . Following is the Basic syntax to declare a pointer variable .

Syntax :- >> Data type * variable name ;

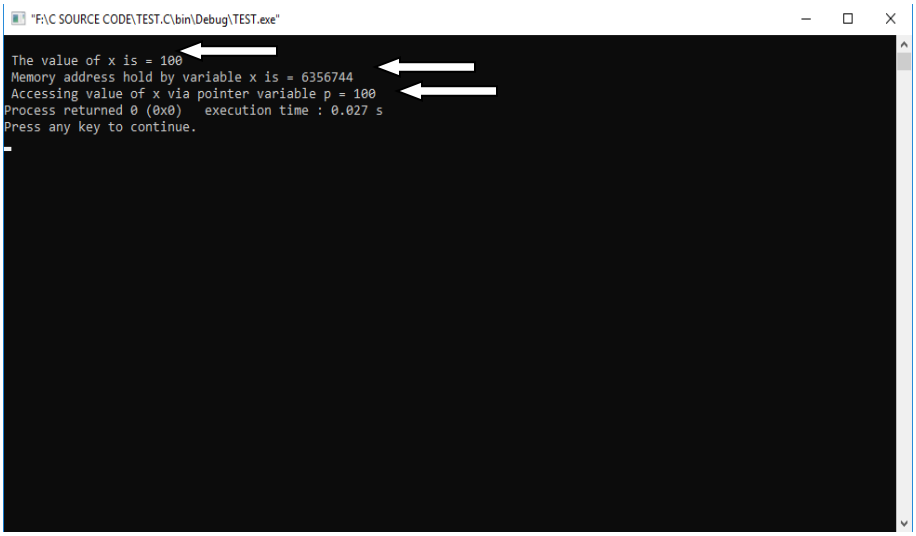
So above is the basic syntax to declare a pointer variable . The data type could be any valid data type like char , int , then astric symbol and then variable name . Following is the Basic example of integer pointer , where p is variable name & data type is integer .

Example :- >> int * p ;

Ok , now after understanding how to declare pointer , now let's see how to use it practically . Following is the Basic program which will help you to understand how to declare & use pointer variable .

/* Write a c program to understand the concept of pointer

```
    Author :- Prashant Shinde    */
#include < stdio.h >
#include < conio.h >
#include  < stdlib.h >
int main ()
{
  int * p ;
  int x = 100 ;
  system ( " cls " ) ;
  printf ( " \n The value of x is = %d " , x ) ;
  p = & x ;
  printf ("\n Memory address hold by variable x is = %d " , p ) ;
  printf (" \n Accessing value of x via pointer variable p = %d ", * p ) ;
 return 0 ;
}
```



Above you can easily observe the output & can understand how it works . Let's start from the beginning , first of all the program control enters into main ( ) function & creates a pointer variable *p , now this pointer variable holds a particular memory address inside the computers memory . After that it goes to next line & creates a variable " x " & initialize it to " 100 " this also holds a unique memory address . After that computer screen will get cleared & it will print the value of x i.e. 100 with the help of printf ( ) function . After that we have pointed the memory address of variable " x " to our pointer variable " p " as " p = & x " . At this line pointer variable "p" is holding the memory address of variable " x " that means we can access the value of variable x through pointer variable . Ok after that we have printed the memory address hold by pointer variable " p " & after that we have accessed the value present at variable " p " as *p with the help of printf ( ) function . So if you go through this program step by step you can easily understand how pointer variable is declared & used and it's working . Solve the following assignment to make the concept of pointer more clear .

ASSIGNMENT :- >> Write a c program to declare & access a value of T with the help of float pointer variable " P " , where value of T = 455.43 .

KEY POINTS :- >>

- Pointer is a variable which holds the memory address of another variable .
- Pointer helps to increase the execution speed of an program .
- Pointer helps in Dynamic memory allocation .
- Pointer is dangerous tool , if it is used by hackers . Hackers can write some types of viruses or password stealing programs with the help of this concept .

12. FILE HANDLING IN C .

Until now we have learned how to interact with data with the help of variables . All the operations that we have performed above are performed inside computers memory . Suppose you have performed an operation of addition or division and you want your data or result back then there is no way to get those result back . So here in this chapter we gonna talk about how to save your data to hard disk , so that we can get our results back whenever we want .

So in file handling we gonna talk about how to perform read & write operations on hard disk . Here we gonna save the data to a file and read that data whenever we want with the help of file handling functions .

What is a file ?

A file is a collection of bytes which is stored inside computers hard disk . A file can be a text file or a binary file . Every file contains an extension associated with it , for example a text file contains an extension .txt where as a binary file contains an extension .bin , also many known file types such as word file ,excel file contains extensions as .doc , .xls .

There are 4 steps which you need to follow in order to create ,read ,write data to a file & those steps are as follows .

1. opening a file .
2. read / write a file .
3. close a file .

So let's see each of them one by one .

1] Opening a File :- >> File creation is the first step in opening a file . But before creating a file it is important to create a file pointer which can hold our file inside the computers memory . Following is the syntax to declare a file pointer .

Syntax :- >> FILE * Pointer ;

In above syntax FILE is a predefined structure which is already defined in header file " stdio.h " , then asterisk symbol * and the pointer name . After defining the file pointer , the

next step would be to open that file either for reading or writing . so to open a file there is function introduced in C called as fopen ( ) . Following is the basic syntax to use this function .

Syntax :- >> pointer = fopen (" FILE_NAME " , " OPENING MODE ") ;

Above is the basic syntax to open up a file . first you need to assign fopen ( ) function to a file pointer to hold the file in computers memory after opening . After that you need to pass two parameters the first one is file name , here you need to set the file name which you wanting to open and second parameter determines the purpose of file opening , that could be for reading / writing / appending data . So this is how the file opening function works . You need to pass the two parameters . Following is the list of different file opening modes which are supported in c language .

File opening modes .	Description .
" r "	Opens an existing text file for reading purpose only .
" w "	Opens an already existing text file for writing purpose , if that file doesn't exist then creates a new file with the same name . The contents are overwritten if the file contains some data already .
" a "	Opens an already existing file for writing in append mode , if the file doesn't exist then creates a new one . It will add the data at the end of the file .
" r + "	It opens up a file for reading & writing purpose .
" w+ "	It also opens up a file for writing purpose , if the file doesn't present then it creates a new file . If the file already contains some data then that data will get overwritten .
" a+ "	It opens up the file in append mode for adding the data at the end of the file . It creates a new file if the file doesn't exists .

Following is an example of opening a file with file pointer in reading mode . You can openup a file in any mode depends on your requirement . You just need to pass the mode parameter via fopen ( ) function along with file name .

Example :- >> FILE * pointer ;

pointer = fopen (" student.txt " , " r ") ;

2] Read or Write to a File :- >> After knowing how a file can be opened with different modes , now let's understand how to read or write some data to a file . There are many functions available in c to read & write data to a file , in this part we gonna see how to read or write a single character to a file . So first talk about how to write a single character to a file . To write a single character to a file there is a function introduced in c called as fputc ( ) ; . This function writes a single character to a file . Following is the basic syntax to use this function .

Syntax :- >> fputc (character variable / 'single character' , file pointer) ;

According to above syntax you can either pass a character variable via fputc ( ) function to add many characters to a file or can pass a sinlge character in single quote as ' c ' to write only a single character to a file . after passing the first argument we need to pass the file pointer as second argument via this function . When we openup a file in writing mode with the help of fopen ( ) function then the file pointer will automatically get set to beginning of the file and the character will be added to it .

Ok after knowing how to write a character to a file with the help of fputc ( ) function now let's understand how to read a character from a file with the help of fgetc ( ) function . fgetc( ) function is used to read characters from a file one by one . It reads a single character at a time from a file .Following is the basic syntax to declare fgetc ( ) function .

Syntax :- >> char var = fgetc (file pointer) ;

Now let's understand how it works , fgetc ( ) function receives file pointer as an argument and reads a single character from a file from beginning and assign it to character variable var and we can see those characters to ouput screen with the help of printf ( ) function . This function read characters one by one . so this is how this function works and we can use it to read a file . As we move forward in this book we gonna see practically how to read and write to a file with the help of these functions .

3] Closing a File :- >> after performing all the above operations it is really important to close the file which has been opened either for reading or writing purpose . Otherwise there is a possibility of losing the data . This is the last part of file handling , so to close a file there is a function introduced in c called fclose ( ) . Following is the basic syntax to use this function .

Syntax :- >> fclose (file pointer) ;

You just need to pass the file pointer via fclose ( ) function , it will automatically close the file which has been opened .

Now let's implement the above information practically . Let's write a c program to read some characters from a text file .

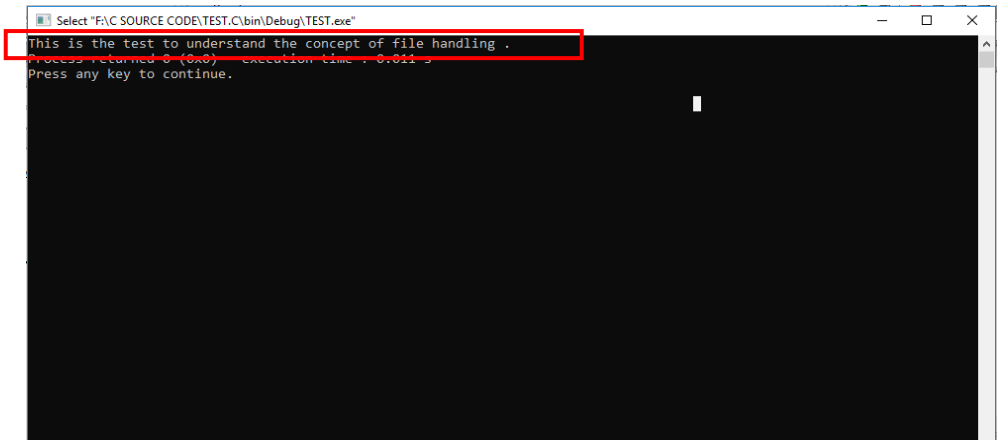
Practical example 1 :- >>

```
/*  Write a C program to read some characters from a file .

    Author :- Prashant Shinde .   */

#include < stdio.h >

#include< conio.h >

#include < stdlib.h >


int main ( )

{

  FILE *pointer ;
```

```
char var ;
pointer = fopen ( "test.txt", "r" ) ;
if ( pointer == NULL )
{
 printf ( " can not open the file test.txt " ) ;
 }
while (1)
{
 var = fgetc ( pointer ) ;
 if ( var == EOF )
 break ;
 printf ( "%c", var ) ;
}
fclose ( pointer ) ;
return 0  ;
  }
```

Now let's understand how above program works , first the program control enters in to main function & creates a pointer variable pointer & a character variable var .Then at the next line with the help of **pointer** = fopen (" test.txt " , " r ") function our file test.txt will get opened in reading mode .we have passed "r" for reading mode via fopen() function . The file test.txt is already created by me inside source directory . In your case you can create test.txt file in your source directory where you are storing your current source code files . After creating the file you need to add some data into it , so that later we can read that data . So after opening the file test.txt in reading mode we need to check whether the file is successfully opened or not . So we can do this with the next line of code that we have used above i.e. if(pointer==NULL) . The function will returns NULL when it unable to open or find the file . NULL means zero or useless value . So if the program unable to find test.txt then it will display the message on screen " Can not open the file test.txt " . This part is also called as error handling . Here we are handling file opening error with the help of already

provided features & functions in c . after error handling we have used while() loop here to read our file . as we know we can only read single character at a time with the help of fgetc ( ) function , that's why we have used here while ( ) loop . We have passed 1 via while ( ) loop that means the loop is going to run infinite times , here 1 means true . after program control entering into while ( ) loop due to fgetc(pointer) function it automatically goes at the beginning of the file & reads first character from our file test.txt & assigns it to variable var . after that we have next line of code to check end of the file . This code will check that whether we are at the end of the file or not . This code will help to detect end of file , so that we can quit our infinite loop . When our file pointer goes at the end of file it automatically returns EOF . EOF is a macro which determines end of file . So with the help of if (var == EOF) break ; we can determine our program control has reached end of file and we need to close the file with break keyword . The break will quit the while ( ) loop . But now we are not at the end of file . So the condition if (var == EOF) will become false & program control will goes to next line and print the current value of character variable var with the help of printf (" %c ",var) and goes at the beginning of while loop because we are inside infinite loop . after that it reads the second character and same procedure will be done until you reach at the end of file and each character will get printed one by one with the help of fgetc() function . when program control detects EOF it will goes to break & will comeout of while (1) loop . After reading & printing all the characters from our file now it's time to close our file . So the program control will goes to next line of code i.e. fclose (pointer) ; and will close the file . Following output will show you the contents of our file test.txt added by me .



ASSIGNMENT :- >> Create a file test.txt with notepad and add whatever content you want to it and read & print the contents of that file to computer screen with the help of fgetc() function .

After learning how to read data from a file , now let's understand how to write data to a file . In the following example we gonna learn how to write characters to a file with the help of fputc ( ) function in c .

Practical example 2 :- >>

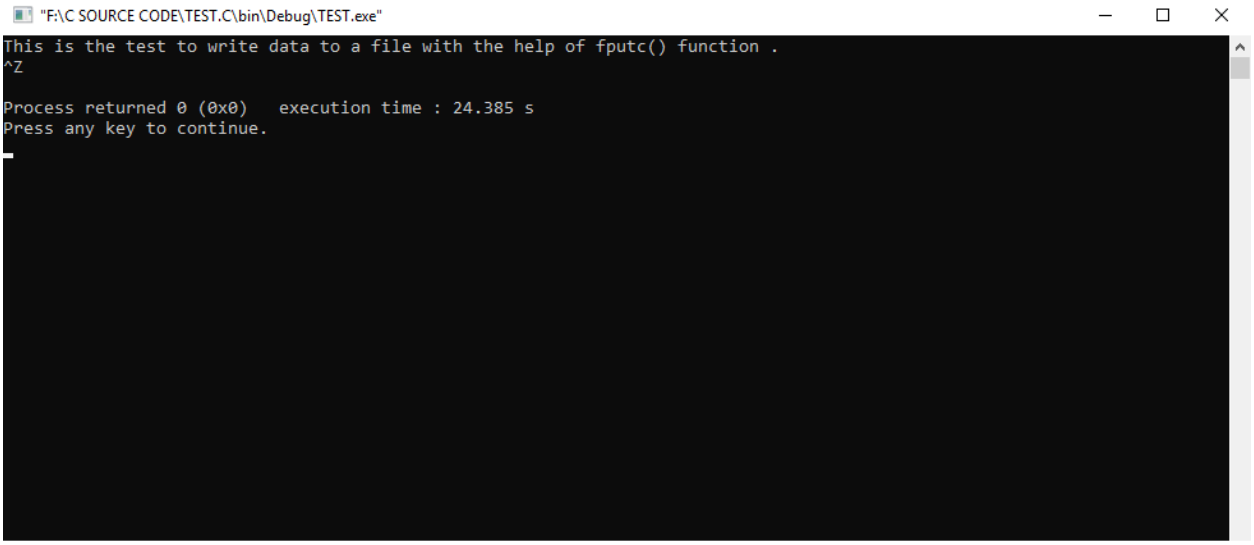
```
/*  Write a c program to write characters  to a  new file named example.txt one by one .
    Author :- Prashant Shinde  */
#include < stdio.h >
#include < conio.h >
#include  < stdlib.h >

int main ( )
{
  FILE * pointer ;
  char  var ;

  pointer = fopen ( "example.txt", "w" ) ;
  if ( pointer == NULL )
  {
   printf ( "can not create a file example.txt ." ) ;
  }
  while (1)
```

```
 {
   var = getchar ( ) ;
   if ( var == EOF )
    break ;
   fputc ( var , pointer ) ;
  }
 fclose ( pointer ) ;
 return 0 ;
}
```

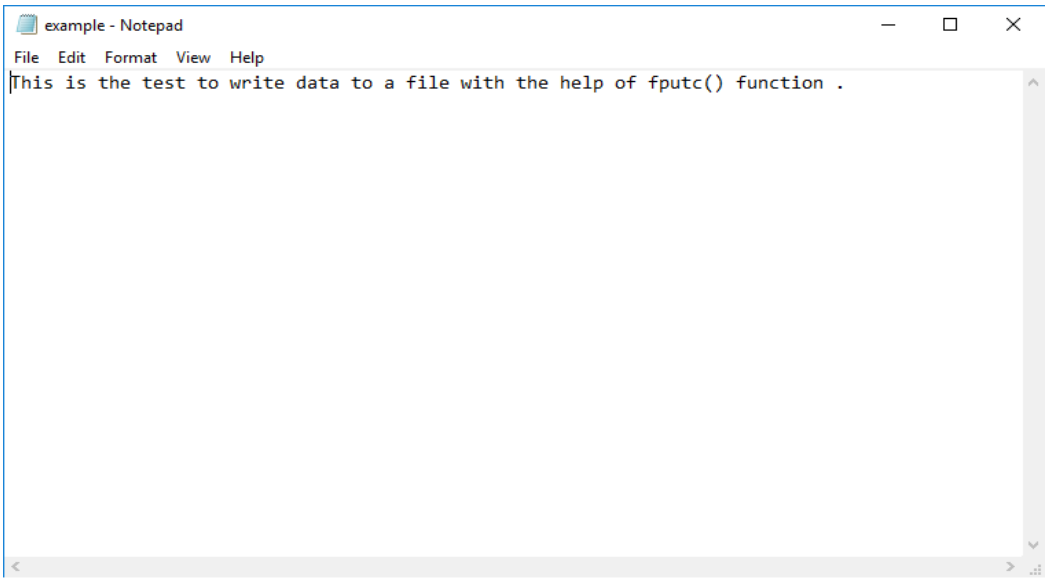
Now let's understand how this program works . First of all the program control enters in to main function and creates a file pointer with the name pointer & after that creates a character variable var to receive characters from keyboard and to write them to our file . after that a new file example.txt will get opened in writing mode due to fopen (" example.txt " , " w ") function . As you can see we have passed "w" parameter via our function fopen() which will help to open any file for writing purpose . After this error handling is done with the help of code if (pointer == NULL) . This code will check whether we have successfully managed to open our file or not . After successful opening our file next code is to write data to our file with the help of while(1) . Here we have used while loop because as we know fputc() can only write single character to a file at a time & we wanting to add many characters to file so that's why we have used this infinite loop . after the program control entering into while loop there is line of code var = getchar ( ) ; , in this code the function getchar() will read what ever character we type from our keyboard and assign it to character variable var . After assigning the character to variable var the next code is to check whether we have pressed control + Z on our keyboard . As we know EOF represents end of file and control + Z is a short cut key for EOF . So whenever we press control+ Z on our keyboard then the while loop will get closed automatically . Ok after checking whether we have pressed control+Z or not then the next code will write that character to our file with the code fputc (var,pointer) ; . The fputc() function will write a single character at a time to our file and again goes to the beginning of the loop and receives second character and again the same procedure will be done . This process repeats it self until we press control + Z on our keyboard . So when we press control+ Z on our keyboard the loop will be break and program control comes out of while loop and goes to fclose(pointer) ; & closes

our file . Following is the output of the data entered and data written to our file example.txt .

In the following output you can see the text that we have entered via keyboard . After entering the text , you need press Enter and then should press control+Z to close the file .



Following is the output of our text file “example.txt” where the data is written . You can check your text file inside your source code directory of codeblocks .



ASSIGNMENT :- >> **Write a c program to write some contents to the file with the help of fputc() function in c .**

KEY POINTS :- >>

- A file is nothing but collection of bytes which is stored on our hard disk .
- File handling helps us to save our data and have it back whenever we want .
- A file pointer is necessary to open up the file either for reading or writing .
- You can open a file in different modes i.e. reading , writing or appending .

---

13. NEXT IMPORTANT STEP IN C PROGRAMMING .

In above lessons we have learned the Basic fundamentals of c programming . Now it's for you to understand the indepth concepts in c programming . To master c programming you have to learn the following topics .

1] STORAGE CLASSES .

2] BITWISE OPERATORS .

3] RECURSION .

4] MEMORY MODELS .

5] DYNAMIC MEMORY ALLOCATION .

6] GRAPHICS PROGRAMMING .

7] COMMAND LINE ARGUMENTS .

8] PREPROCESSOR DIRECTIVE .

9] ADVANCED POINTERS .

10] MOUSE PROGRAMMING

11] KEYBOARD PROGRAMMING .

12] SOFTWARE DEVELOPMENT .

13] GAME PROGRAMMING .

14] WRITING A SMALL COMPUTER VIRUS .

NOTE :- If you want to master c programming or any other programming language then all the above topics in c are really important to understand clearly . If you want to learn above topics without wasting time and energy then you can simply get my course published on udemy platform . All the above topics are covered in detail in my computer course . The name of the course is " The complete c developer course – Build 7 exciting projects ". This course has been taken by over 9500+ students from 121 different countries all around the world . This course covers from very Basic topics to advanced concepts in c programming . It also covers 3 mini projects and two advanced level projects . The price of the course is $195 . If you want this course for just between $10-15 , then simply go to https://www.codeoffline.com . You will find the $10-15 dollar coupon

links here at any time . You can share these links to any one who is interested in computer studies . Your job is to follow the links present on our site to get a discount at any time .

Following is the list of courses published on udemy.com

The Complete C Developer Course - Build 7 Exciting Projects .

Prashant Shinde . $199

4.2

The complete Ubuntu Linux Server Administration Course.

Prashant Shinde . $190

4.3

Unrevealed Secret Dos Commands For Ethical Hackers .

Prashant Shinde . $90

4.4

The Complete Virtualization Technology for Computer Geeks .

Prashant Shinde . Free

4.5