***History of C Programing***

The C programming language is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs. C was originally first implemented on the DEC PDP-11 computer in 1972.

In 1978, Brian Kernighan and Dennis Ritchie produced the first publicly available description of C, now known as the K&R standard.

The UNIX operating system, the C compiler, and essentially all UNIX applications programs have been written in C. The C has now become a widely used professional language for various reasons.

 Easy to learn

 Structured language

 It produces efficient programs.

 It can handle low-level activities.

 It can be compiled on a variety of computer platforms.

**C Environment Setup**

This section describes how to set up your system environment before you start doing your

programming using C language.

Before you start doing programming using C programming language, you need the following

twosoftwares available on your computer, (a) Text Editor and (b) The C Compiler.

**Text Editor**

This will be used to type your program. Examples of few editors include Windows Notepad,

OS Edit command, Brief, Epsilon, EMACS, and vim or vi. Name and version of text editor can vary on different operating systems. For example, Notepad will be used on Windows, and vim or vi can be used on windows as well as Linux or

**UNIX.**

The files you create with your editor are called source files and contain program source

code. The source files for C programs are typically named with the extension “.c”.

Before starting your programming, make sure you have one text editor in place and you

have enough experience to write a computer program, save it in a file, compile it and finally

Execute it.

**The C Compiler**

The source code written in source file is the human readable source for your program. It

needs to be "compiled", to turn into machine language so that your CPU can actually

execute the program as per instructions given.

This C programming language compiler will be used to compile your source code into final

executable program. I assume you have basic knowledge about a programming language

compiler.

Most frequently used and free available compiler is GNU C/C++ compiler,

otherwise you can have compilers either from HP or Solaris if you have respective Operating Systems.

Following section guides you on how to install GNU C/C++ compiler on various OS. I'm

mentioning C/C++ together because GNU gcc compiler works for both C and C++

programming languages.

**Why to use C?**

C was initially used for system development work, in particular the programs that make up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

**Semicolons ;**

In C program, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.

**Identifiers**

A C identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore \_ followed by zero or more letters, underscores, and digits (0 to 9).

C does not allow punctuation characters such as @, $, and % within identifiers. C is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in C. Here are some examples of acceptable identifiers:

**C Data Types**

In the C programming language, data types refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

The types in C can be classified as follows:

Basic Types: They are arithmetic types and consists of the two types: (a) integer types and (b) floating-point types.

**Integer Types**

|  |  |  |
| --- | --- | --- |
| Char | 1 byte | -128 to 127 or 0 to 255 |

|  |  |  |
| --- | --- | --- |
| unsigned char | 1 byte | 0 to 255 |

|  |  |  |
| --- | --- | --- |
| signed char | 1 byte | -128 to 127 |

|  |  |  |
| --- | --- | --- |
| Int | 2 or 4 bytes | -32,768 to 32,767 or -2,147,483,648 to 2,147,483,647 |

|  |  |  |
| --- | --- | --- |
| unsigned int | 2 or 4 bytes | 0 to 65,535 or 0 to 4,294,967,295 |

|  |  |  |
| --- | --- | --- |
| Short | 2 bytes | -32,768 to 32,767 |

|  |  |  |
| --- | --- | --- |
| unsigned short | 2 bytes | 0 to 65,535 |

|  |  |  |
| --- | --- | --- |
| Long | 4 bytes | -2,147,483,648 to 2,147,483,647 |

|  |  |  |
| --- | --- | --- |
| unsigned long | 4 bytes | 0 to 4,294,967,295 |

**Floating-Point Types:-**

|  |  |  |  |
| --- | --- | --- | --- |
| float | 4 byte | 1.2E-38 to 3.4E+38 | 6 decimal places |

|  |  |  |  |
| --- | --- | --- | --- |
| double | 8 byte | 2.3E-308 to 1.7E+308 | 15 decimal places |

|  |  |  |  |
| --- | --- | --- | --- |
| long double | 10 byte | 3.4E-4932 to 1.1E+4932 | 19 decimal places |

**C Variables:-**

A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C is case-sensitive. Based on the basic types explained in previous chapter, there will be the following basic variable types:

|  |  |
| --- | --- |
| Char | Typically a single octet(one byte). This is an integer type. |

|  |  |
| --- | --- |
| Int | The most natural size of integer for the machine. |
| Float | A single-precision floating point value. |

|  |  |
| --- | --- |
| Double | A double-precision floating point value. |

|  |  |
| --- | --- |
| Void | Represents the absence of type. |

**Variable Definition in C:**

A variable definition means to tell the compiler where and how much to create the storage for the variable. A variable definition specifies a data type and contains a list of one or more variables of that type as follows:

int i, j, k;

char c, ch;

float f, salary;

double d;

|  |  |
| --- | --- |
| **Escape sequence** | **Meaning** |
| \\ | \ character |
| \' | ' character |
| \" | " character |
| \? | ? character |
| \a | Alert or bell |
| \b | Backspace |
| \f | Form feed |
| \n | Newline |
| \r | Carriage return |
| \t | Horizontal tab |
| \v | Vertical tab |
| \ooo | Octal number of one to three digits |

**C Operators:-**

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C language is rich in built-in operators and provides the following types of operators:

 Arithmetic Operators

 Relational Operators

 Logical Operators

**Arithmetic Operators:-**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two operands | A + B will give 30 |
| - | Subtracts second operand from the first | A - B will give -10 |
| \* | Multiplies both operands | A \* B will give 200 |
| / | Divides numerator by de-numerator | B / A will give 2 |
| % | Modulus Operator and remainder of after an integer division | B % A will give 0 |
| ++ | Increments operator increases integer value by one | A++ will give 11 |
| -- | Decrements operator decreases integer value by one | A-- will give 9 |

**Relational Operators:-**

|  |  |  |
| --- | --- | --- |
| == | Checks if the values of two operands are equal or not, if yes then condition becomes true. | (A == B) isnot true. |
| != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |

**Logical Operators:-**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | Called Logical AND operator. If both the operands are non-zero, then condition becomes true. | (A && B) is false. |
| || | Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true. | (A || B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false. | !(A && B) is true. |

**Input & Output**

When we are saying Input that means to feed some data into program. This can

be given in the form of file or from command line. C programming language provides a set

of built-in functions to read given input and feed it to the program as per requirement.

When we are saying Output that means to display some data on screen, printer or in any

file. C programming language provides a set of built-in functions to output the data on the

computer screen as well as you can save that data in text or binary files.

The Standard Files

C programming language treats all the devices as files. So devices such as the display are

addressed in the same way as files and following three file are automatically opened when

a program executes to provide access to the keyboard and screen.

**Decision Making in C**

Decision making structures require that the programmer specify one or more

conditions to be evaluated or tested by the program, along with a statement or statements

to be executed if the condition is determined to be true, and optionally, other statements to

be executed if the condition is determined to be false.

Following is the general form of a typical decision making structure found in most of the

programming languages:

C programming language assumes any non-zero and non-null values as true, and if it is

either zero or null, then it is assumed as false value. C programming language provides

following types of decision making statements.

**if statement**

An if statement consists of a boolean expression followed by one or more statements.

Syntax .The syntax of an if statement in C programming language is:

if(boolean\_expression)

{

/\* statement(s) will execute if the boolean expression is true \*/

}

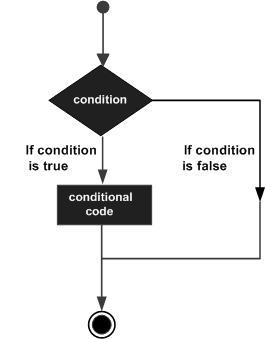
If the boolean expression evaluates to true, then the block of code inside the if

statement will be executed. If boolean expression evaluates to false, then the first set of

code after the end of the if statement (after the closing curly brace) will be executed.

C programming language assumes any non-zero and non-null values as true and if it is

either zero or null then it is assumed as false value.



**switch statement :-**

A switch statement allows a variable to be tested for equality against a list of values. Each

value is called a case, and the variable being switched on is checked for each switch case.

Syntax

The syntax for a switch statement in C programming language is as follows:

switch(expression){

case constant-expression :

statement(s);

break; /\* optional \*/

case constant-expression :

statement(s);

break; /\* optional \*/

/\* you can have any number of case statements \*/

default : /\* Optional \*/

statement(s);

}

You can have any number of case statements within a switch. Each case is followed by

the value to be compared to and a colon.



The constant-expression for a case must be the same data type as the variable in the

switch, and it must be a constant or a literal.



When the variable being switched on is equal to a case, the statements following that

case will execute until a break statement is reached.



When a break statement is reached, the switch terminates, and the flow of control jumps

to the next line following the switch statement.



Not every case needs to contain a break. If no break appears, the flow of control will fall

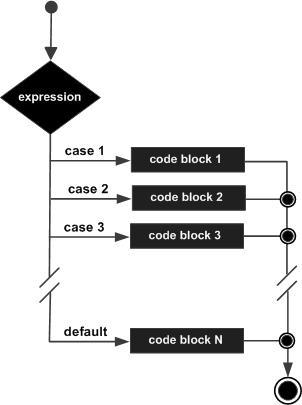
through to subsequent cases until a break is reached.



A switch statement can have an optional default case, which must appear at the end of

the switch. The default case can be used for performing a task when none of the cases

is true. No break is needed in the default case.



**while loop in C**

A while loop statement in C programming language repeatedly executes a target

statement as long as a given condition is true.

Syntax

The syntax of a while loop in C programming language is:

**while(condition)**

**{**

**statement(s);**

**}**

Here, statement(s) may be a single statement or a block of statements.

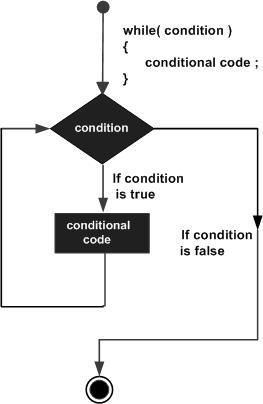
The condition may be any expression, and true is any nonzero value. The loop iterates

while the condition is true.

When the condition becomes false, program control passes to the line immediately

following the loop.

Flow Diagram



**For Loop in C**

A for loop is a repetition control structure that allows you to efficiently write a loop that

needs to execute a specific number of times.

Syntax

The syntax of a for loop in C programming language is:

for ( init; condition; increment )

Here is the flow of control in a for loop:

1.

The init step is executed first, and only once. This step allows you to declare and

initialize any loop control variables. You are not required to put a statement here, as long

as a semicolon appears.

2.

Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is

false, the body of the loop does not execute and flow of control jumps to the next

statement just after the for loop.

3.

After the body of the for loop executes, the flow of control jumps back up to

the increment statement. This statement allows you to update any loop control

variables. This statement can be left blank, as long as a semicolon appears after the

condition.

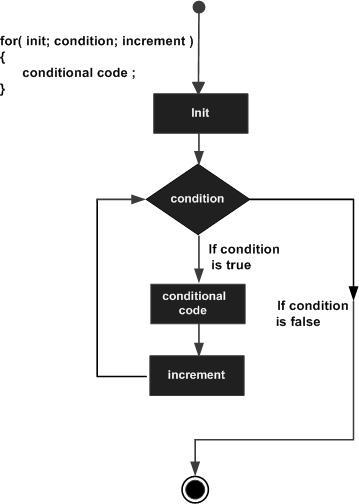
4.

The condition is now evaluated again. If it is true, the loop executes and the process

repeats itself (body of loop, then increment step, and then again condition). After the

condition becomes false, the for loop terminates.

Flow Diagram



**do...while loop in C**

Unlike for and while loops, which test the loop condition at the top of the loop,

the do...while loop in C programming language checks its condition at the bottom of the

loop.

A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to

execute at least one time.

Syntax

The syntax of a do...while loop in C programming language is:

do

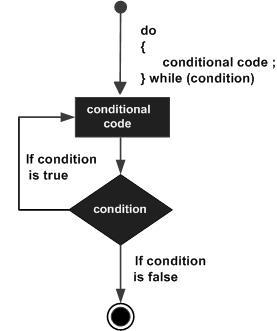
{

statement(s);

}while( condition );

Notice that the conditional expression appears at the end of the loop, so the statement(s)

in the loop execute once before the condition is tested.



C programming language allows to use one loop inside another loop. Following section

shows few examples to illustrate the concept.

Syntax

The syntax for a nested for loop statement in C is as follows:

for ( init; condition; increment )

{

for ( init; condition; increment )

{

statement(s);

}

statement(s);

}

The syntax for a nested while loop statement in C programming language is as follows:

while(condition)

{

while(condition)

{

statement(s);

}

statement(s);

}

The syntax for a nested do...while loop statement in C programming language is as

follows:

do

{

statement(s);

do

{

statement(s);

}while( condition );

}while( condition );

**break statement in C**

The break statement in C programming language has the following two usages:

1.

When the break statement is encountered inside a loop, the loop is immediately

terminated and program control resumes at the next statement following the loop.

2.

It can be used to terminate a case in the switch statement (covered in the next chapter).

If you are using nested loops (i.e., one loop inside another loop), the break statement

will stop the execution of the innermost loop and start executing the next line of code after

the block.

Syntax

The syntax for a break statement in C is as follows:

break;

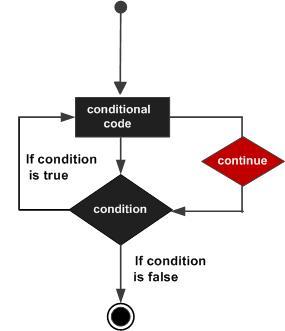
**The Infinite Loop :-**

A loop becomes infinite loop if a condition never becomes false. The for loop is

traditionally used for this purpose. Since none of the three expressions that form the for

loop are required, you can make an endless loop by leaving the conditional expression

empty.



**C Structures**

C arrays allow you to define type of variables that can hold several data items of the same

kind but structure is another user defined data type available in C programming, which

allows you to combine data items of different kinds.

Structures are used to represent a record, suppose you want to keep track of your books in

a library. You might want to track the following attributes about each book:



Title



Author



Subject



Book ID

Defining a Structure

To define a structure, you must use the struct statement. The struct statement defines a

new data type, with more than one member for your program. The format of the struct

statement is this:

struct [structure tag]

{

member definition;

member definition;

...

member definition;

} [one or more structure variables];

The structure tag is optional and each member definition is a normal variable definition,

such as int i; or float f; or any other valid variable definition. At the end of the structure's

definition, before the final semicolon, you can specify one or more structure variables but it

is optional. Here is the way you would declare the Book structure:

struct Books

Accessing Structure Members

To access any member of a structure, we use the member access operator (.). The member access operator is coded as a period between the structure variable name and the structure member that we wish to access. You would use struct keyword to define variables of structure type. Following is the example to explain usage of structure:

**Practical Program**

***Assignment-1 (Intro to c how to create & run)***

#include<stdio.h>

#include<conio.h>

void main()

{

clrscr();

printf("my name is kailashpunjabi.\n");

printf("i reside at:\t109 prembhuvan,");

printf("\n\t\tbazar road,");

printf("\n\t\tbandra(w),");

printf("\n\t\tmumbai 400050");

getch();

}

Explanation :-

#include<stdio.h>🡪This is header file standard input out

#include<conio.h>🡪This is also header file Console Input Out put

Void main() 🡪 Main program where u have to write code

Printf🡪 It will show out put on turbo C

Getch() 🡪 it will hold the output.

Clrscr(); 🡪 it will clear the screen

***Assignment- 2 (Declaring AndIntitalizing Variables)***

#include <stdio.h>

#include <conio.h>

void main()

{

//Declaring And Intitalizing Variables

char xyz= 'A';

int inum= 21;

float fnum=87.65;

clrscr();

//Displaying the values with Conversion And Escape Characters

printf("\n\n");

printf("Char is \t= %c \n",xyz);

printf("Int is \t= %d \n",inum);

printf("Float is \t= %f \n",fnum);

getch();

}

***Assignment- 3 (Use of printf() and scaf() function with output)***

#include <stdio.h>

#include <conio.h>

#include<string.h>

void main()

{

//Declaring Variables

int rollno;

float height;

char name;

//Use of printf() and scanf() function

clrscr();

printf("\nEnter Your name ");

scanf("%s",&name);

printf("\nEnter the roll no: ");

scanf("%d",&rollno);

printf("\nEnter the height: ");

scanf("%f",&height);

//Displaying the values entered

printf("\nyour name is:%s",name);

printf("\nRoll no is:%d",rollno);

printf("\nHeight is:%f\n",height);

getch();

}

***Assignment- 4 (Arithmetic Performance)***

#include <stdio.h>

#include <conio.h>

void main()

{

//Declaration and Intialization of the variable

inta,b,c,d;

int sum, multi, div,remainder, minus, increase, decrease;

c=25;

d=12;

printf("\nEnter First Number: ");

scanf("%d",&a);

printf("\nEnter Second Number ");

scanf("%d",&b);

// Use of Arithmatic operators

sum = a+b;//Addition

minus = a-b;//Subtraction

multi = a\*b;//Multiplication

div = b/a;//Division

remainder = a%b;//Modular Division

increase = ++c;

decrease = --d;

//Displaying the results

clrscr();

printf ("Sum is %d\n",sum);

printf ("Minus of two number is %d\n",minus);

printf ("multiplication of two num is %d\n",multi);

printf ("Division of two number is %d\n",div);

printf ("Remainder is %d\n",remainder);

printf ("After Increasing the values is %d\n",increase);

printf ("After Decreasing the values is %d\n",decrease);

getch();

}

***Assignment- 5 (Displaying the size occupied by each data type)***

#include <stdio.h>

#include <conio.h>

void main()

{

//Declaring variables

clrscr();

int a;

char b;

float pi;

// Displaying the size occupied by each data type

printf("Size of Character variable is %d\n", sizeof(char));

printf("Size of Integer variable is %d\n", sizeof(a));

printf("Size of Float variable is %d\n", sizeof(float));

getch();

}

***Assignment- 6 (Swap Program)***

#include<conio.h>

# include <stdio.h>

void main()

{

inta,b,c;

printf("\nenter the first number:");

scanf("%d",&a);

printf("\nenter the second number:");

scanf("%d “,&b);

// Displayingthe numbers before interchanging

printf("\n\n printing the numbers before interchanging");

printf("\nthe first number is:%d",a);

printf("\nthe second number is:%d",b);

//Interchangingthe numbers

c=a;

a=b;

b=c;

// Displayingthe numbers after interchanging

printf("\n\n printing the numbers after interchanging");

printf("\nthe first number now is:%d",a);

printf("\nthe second number now is:%d",b);

getch();

}

***Assignment- 7 (Salary Calculation With If Else)***

#include<conio.h>

#include <stdio.h>

void main()

{

Float basic,da,hra,salary;

char d[15];

clrscr();

printf("\nEnter Your Name : ");

scanf("%s",&d);

printf("\n enter the basic salary :");

scanf("%f",&basic);

// Calculate the da,hra and salary

da=basic\*40/100;

hra=basic\*25/100;

salary=basic+da+hra;

//Displaying the details

printf("\nyour name is: %c ",d);

printf("\n\n salary details :");

printf("\nBasic salary is: %f",basic);

printf("\n Dearness Allowance is: %f",da);

printf("\n House Rent Allowance is: %f",hra);

printf("\n\n Total salary earned %f\n",salary);

if(salary>=80000)

{

printf("\nYou are ceo ");

}

else if(salary>=50000)

{

printf("\nYou are purchase manager");

}

else if(salary>=25000)

{

printf("\nyou are sales manager");

}

else

{

printf("\nyou are clerk");

}

getch();

}

***Assignment- 8 (Use Of If Else)***

#include <stdio.h>

#include <conio.h>

void main()

{

int num1,num2, sum;

printf("enter two numbers:");

scanf("%d %d", &num1,&num2);

sum= num1+num2;

if(sum>100)

printf("\n the sum of two numbers is greater than 100\n");

else

printf("\n the sum of two numbers is smaller than 100\n");

getch();

}

***Assignment- 9 (Voting System Using if Else)***

#include <stdio.h>

#include <conio.h>

void main()

{

int age;

char name;

clrscr();

printf("\n enter your name:");

scanf("%s", &name);

printf("\n enter your age:");

scanf("%d", &age);

if (age>=19)

{

printf("\n you are eligiable for voting\n");

}

else

{

printf("\n you are not eligible for voting\n");

}

getch();

}

***Assignment- 10 (Use Of Ladder If)***

#include <stdio.h>

#include <conio.h>

void main()

{

int num1,num2, num3;

clrscr();

printf("\n enter 3 numbers:");

scanf("%d %d %d", &num1,&num2,&num3);

if((num1>num2) && (num1>num3))

printf("\n the largest of three numbers is %d \n",num1);

if((num2>num1) && (num2>num3))

printf("\n the largest of three numbers is %d \n",num2);

if((num3>num1) && (num3>num2))

printf("\n the largest of three numbers is %d \n",num3);

getch();

}

***Assignment- 11 (Use Of If & Ladder Else If)***

#include <stdio.h>

#include <conio.h>

void main()

{

Char in\_char;

printf("\nEnter a character in lower case: ");

scanf("%c", &in\_char);

if(in\_char=='a' || in\_char=='A')

printf("\nThe character input is a vowel a\n");

else if (in\_char=='e')

printf("\nThe character input is a vowel e\n");

else if (in\_char=='i')

printf("\nThe character input is a vowel i\n");

else if (in\_char=='o')

printf("\nThe character input is a vowel o\n");

else if (in\_char=='u')

printf("\nThe character input is a vowel u\n");

else

printf("\nThe character input is not a vowel\n");

getch();

}

***Assignment- 12 (Switch Program)***

#include<stdio.h>

#include<conio.h>

void main()

{

//Declaration and Intialization of the variable

int a;

printf("\nEnter Number with in one to seven: ");

scanf("%d",&a);

printf ("you have entered number is %d\n",a);

switch(a)

{

case 1 : printf("you have selected monday");

break;

case 2 : printf("you have selected tuesday");

break;

case 3 : printf("you have selected wednesday");

break;

case 4 : printf("you have selected thursday");

break;

case 5 : printf("you have selected friday");

break;

case 6 : printf("you have selected saturday");

break;

case 7 : printf("you have selected sunday");

break;

default :printf("wrong choice");

}

getch();

}

***Assignment- 13 (Arithmetic Using Switch)***

#include<stdio.h>

#include<conio.h>

void main()

{

//Declaration and Intialization of the variable

int a,b,choice,add,sub,div,mul;

clrscr();

printf("\nEnter 1 st Number: ");

scanf("%d",&a);

printf("\nEnter 2nd Number");

scanf("%d",&b);

printf("\nPress 1 for addition\nPress 2 for substraction \nPress 3 for Division \nPress 4 for multiplication");

scanf("%d",&choice);

add=a+b;

sub=a-b;

div=a/b;

mul=a\*b;

switch(choice)

{

case 1 : printf("addition of two num is %d",add);

break;

case 2 : printf("substraction of two num is %d",sub);

break;

case 3 : printf("division of two num is %d",div);

break;

case 4 : printf("multiplication of two num is %d",mul);

break;

default :printf("wrong choice");

}

getch();

}

***Assignment- 14 (While Loop )***

#include <stdio.h>

#include <conio.h>

void main()

{

int num1=1, num2=5, count=1;

int product;

while (count<=5)

{

product = num1\*num2;

printf("Product=%d\n", product);

count = count+1;

num1 = num1+1;

}

getch();

}

***Assignment- 15 (While Loop with calculation)***

#include<conio.h>

#include<stdio.h>

void main()

{

int x=1,n,r;

clrscr();

printf("enter any number ");

scanf("%d",&n);

while(x<=10)

{

r=n\*x;

printf("\n%d\*",n);

printf("%d=",x);

printf(" %d ",r);

x++;

}

getch();

}

***Assignment- 16 (Star Printing using While Loop)***

#include <stdio.h>

#include <conio.h>

void main()

{

int i=1, j;

while(i<=10)

{

j=1;

while (j <= i)

{

printf("\*");

j++;

}

printf("\n");

i++;

}

getch();

}

***Assignment- 17 (While Loop With If Else)***

#include <stdio.h>

#include <conio.h>

void main()

{

int number;

int sum=0;

printf("\nEnter the number:");

scanf("%d", &number);

if (number>0)

{

while(number>0)

{

sum = sum+number;

number = number-1;

printf(“\nnumber value is: %d”,number);

printf(“\nsum is : %d”,sum);

}

printf("\nThe sum %d\n", sum);

}

else

printf("\n%d is not valid.", number);

getch();

}

***Assignment- 18 (Do While Loop )***

#include <stdio.h>

#include <conio.h>

void main ()

{

int a = 10;

do

{

printf("value of a: %d\n", a);

a = a + 1;

}while( a < 20 );

getch();

}

***Assignment- 19 (table printing using For Loop)***

#include<conio.h>

#include<stdio.h>

void main()

{

inta,b,c;

clrscr();

printf("enter table number : \n");

scanf("%d",&b);

for(a=1;a<=10;a++)

{

c=a\*b;

printf("%d\*",b);

printf("%d=",a);

printf("%d\n",c);

}

getch();

}

***Assignment- 20 (Star Printing Using For Loop)***

#include<conio.h>

#include<stdio.h>

void main()

{

inta,b;

for(a=1;a<=10;a++)

{

for(b=1;b<=a;b++)

{

printf("\*",b);

}

printf("\n");

}

getch();

}

***Assignment- 21 (Star Printing REVERSE-RIGHT-ANGLE)***

#include<stdio.h>

#include<conio.h>

void main()

{

inti,j,k,a,b,c;

for(i=1;i<=8;i++)

{

for(j=8;j>=i;j--)

printf(" ");

for(k=1;k<=i;k++)

printf("\*");

printf("\n");

}

getch();

}

***Assignment- 22 (SUN-RAYS star design)***

#include<stdio.h>

#include<conio.h>

void main()

{

inta,b,c,x,y,z;

printf("Enter a number..\n");

scanf("%d",&x);

for(a=1;a<=x;a++)

{

for(y=1;y<=a;y++)

{

printf(" ");

}

for(z=1;z<=a;z++)

{

printf("\* ");

}

printf("\n");

}

getch();

}

***Assignment- 23 (Array one dimension****)*

#include<stdio.h>

#include<conio.h>

void main()

{

int x[3];

x[0]=10;

x[1]=20;

x[2]=30;

printf("%d",x[0]);

printf("%d",x[1]);

printf("%d",x[2]);

getch();

}

***Assignment- 24 (Array multi dimension)***

#include<stdio.h>

#include<conio.h>

void main()

{

int x[2][3];

//first row of array

x[0][0]=10;

x[0][1]=20;

x[0][2]=30;

clrscr();

printf("%d",x[0][0]);

printf("%d",x[0][1]);

printf("%d",x[0][2]);

//secon row of array

x[1][0]=40;

x[1][1]=50;

x[1][2]=60;

printf("\n%d",x[1][0]);

printf("%d",x[1][1]);

printf("%d",x[1][2]);

getch();

}

***Assignment- 25 (String Compare Function)***

#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

char a[100], b[100];

printf("Enter the first string\n");

gets(a);

printf("Enter the second string\n");

gets(b);

if (strcmp(a,b) == 0)

{

printf("Entered strings are equal.\n");

}

Else

{

printf("Entered strings are not equal.\n");

 }

getch();

}

***Assignment- 26 (Strcat use for joining two string)***

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char a[20],b[20];

clrscr();

printf("ENTER 1st THE STRING");

scanf("%s",a);

printf("ENTER 2ndTHE STRING");

scanf("%s",b);

strcat(a,b);

printf("concatenation of string is %s",a);

getch();

}

***Assignment- 27 (string copy &strlenth)***

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char a[20],b[20];

intlen;

clrscr();

printf("ENTER 1st THE STRING");

scanf("%s",a);

len =strlen(a);

strcpy(b,a);

printf("copy string is %s",b);

printf("lenth of string is %d",len);

getch();

}

***Assignment- 28 (String reverse )***

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char a[20];

clrscr();

printf("ENTER 1st THE STRING");

scanf("%s",a);

strrev(a);

printf("reverse string is %s",a);

getch();

}

***Assignment- 29 (Pointer will read the address value of Variable)***

#include<stdio.h>

#include<conio.h>

void main()

{

int a=10;

int \*p; /\*pointer variable\*/

p=&a; /\*assign memory address of variable \*/

printf(“address of a =%u”,p);

printf(“value of a =%d”,\*p);

getch();

}

***Assignment- 30 (Pointer of Pointer)***

#include<stdio.h>

#include<conio.h>

void main()

{

int x,\*p1,\*\*p2;

x=5;

p1=&x;

p2=&p1;

printf(“x=%d”,x);

printf(“address of x =%u”,&x);

printf(“address of p1=%u”,p1);

printf(“address of p2=%u”,p2);

getch();

}

***Assignment- 31 (Use of FunctionDeclaration)***

#include<conio.h>

#include<stdio.h>

// function prototype, also called function declaration

float square ( float x );

// main function, program starts from here

int main( )

{

float m, n ;

clrscr();

printf ( "\nEnter some number for finding square \n");

scanf ( "%f", &m ) ;

// function call

n = square ( m ) ;

printf ( "\nSquare of the given number %f is %f",m,n );

getch();

}

float square ( float x ) // function definition

{

float p ;

p = x \* x ;

return ( p ) ;

}

***Assignment- 32 (Structure in c )***

#include<stdio.h>

#include<conio.h>

#include<string.h>

struct Books

{

char title[50];

char author[50];

char subject[100];

int book\_id;

};

int main( )

{

struct Books Book1; /\* Declare Book1 of type Book \*/

struct Books Book2; /\* Declare Book2 of type Book \*/

clrscr();

/\* book 1 specification \*/

strcpy( Book1.title, "C Programming");

strcpy( Book1.author, "Nuha Ali");

strcpy( Book1.subject, "C Programming Tutorial");

Book1.book\_id = 6495407;

/\* book 2 specification \*/

strcpy( Book2.title, "Telecom Billing");

strcpy( Book2.author, "Zara Ali");

strcpy( Book2.subject, "Telecom Billing Tutorial");

Book2.book\_id = 6495700;

/\* print Book1 info \*/

printf( "Book 1 title : %s", Book1.title);

printf( "\nBook 1 author : %s", Book1.author);

printf( "\nBook 1 subject : %s", Book1.subject);

printf( "\nBook 1 book\_id : %d", Book1.book\_id);

/\* print Book2 info \*/

printf( "\nBook 2 title : %s", Book2.title);

printf( "\nBook 2 author : %s", Book2.author);

printf( "\nBook 2 subject : %s", Book2.subject);

printf( "\nBook 2 book\_id : %d", Book2.book\_id);

getch();

}

**Question And Answer For interview Purpose**

**What is the difference between declaration and definition of a variable/function**  
**Ans:** Declaration of a variable/function simply declares that the variable/function exists somewhere in the program but the memory is not allocated for them. But the declaration of a variable/function serves an important role. And that is the type of the variable/function. Therefore, when a variable is declared, the program knows the data type of that variable. In case of function declaration, the program knows what are the arguments to that functions, their data types, the order of arguments and the return type of the function. So that’s all about declaration. Coming to the definition, when we define a variable/function, apart from the role of declaration, it also allocates memory for that variable/function. Therefore, we can think of definition as a super set of declaration. (or declaration as a subset of definition). From this explanation, it should be obvious that a variable/function can be declared any number of times but it can be defined only once. (Remember the basic principle that you can’t have two locations of the same variable/function).

**When should we use pointers in a C program?**  
**1.** To get address of a variable  
**2.** *For achieving pass by reference in C:* Pointers allow different functions to share and modify their local variables.  
**3.** *To pass large structures* so that complete copy of the structure can be avoided.  
C  
**4.** *To implement “linked” data structures* like linked lists and binary trees.

**What is NULL pointer?**   
**Ans:** NULL is used to indicate that the pointer doesn’t point to a valid location. Ideally, we should initialize pointers as NULL if we don’t know their value at the time of declaration. Also, we should make a pointer NULL when memory pointed by it is deallocated in the middle of a program.

**What are static functions? What is their use?**  
**Ans:**In C, functions are global by default. The “static” keyword before a function name makes it static. Unlike global functions in C, access to static functions is restricted to the file where they are declared. Therefore, when we want to restrict access to functions, we make them static. Another reason for making functions static can be reuse of the same function name in other files. See [this](http://www.geeksforgeeks.org/what-are-static-functions-in-c/)for examples and more details.

**What are main characteristics of C language?**  
C is a procedural language. The main features of C language include low-level access to memory, simple set of keywords, and clean style. These features make it suitable for system programming like operating system or compiler development.

**Some coders debug their programs by placing comment symbols on some codes instead of deleting it. How does this aid in debugging?**

Placing comment symbols /\* \*/ around a code, also referred to as “commenting out”, is a way of isolating some codes that you think maybe causing errors in the program, without deleting the code. The idea is that if the code is in fact correct, you simply remove the comment symbols and continue on. It also saves you time and effort on having to retype the codes if you have deleted it in the first place.

**What is variable initialization and why is it important?**

This refers to the process wherein a variable is assigned an initial value before it is used in the program. Without initialization, a variable would have an unknown value, which can lead to unpredictable outputs when used in computations or other operations.

**In C programming, how do you insert quote characters (‘ and “) into the output screen?**

This is a common problem for beginners because quotes are normally part of a printf statement. To insert the quote character as part of the output, use the format specifiers \’ (for single quote), and \” (for double quote).

**What is the use of a ‘\0′ character?**

It is referred to as a terminating null character, and is used primarily to show the end of a string value.

**What is the difference between the = symbol and == symbol?**

The = symbol is often used in mathematical operations. It is used to assign a value to a given variable. On the other hand, the == symbol, also known as “equal to” or “equivalent to”, is a relational operator that is used to COMPARE two values.

**What is the modulus operator?**

The modulus operator outputs the remainder of a division. It makes use of the percentage (%) symbol. For example: 10 % 3 = 1, meaning when you divide 10 by 3, the remainder is 1.

**What is a nested loop?**

A nested loop is a loop that runs within another loop. Put it in another sense, you have an inner loop that is inside an outer loop. In this scenario, the inner loop is performed a number of times as specified by the outer loop. For each turn on the outer loop, the inner loop is first performed.

**Which of the following operators is incorrect and why? ( >=, <=, <>, ==)**

<> is incorrect. While this operator is correctly interpreted as “not  equal to” in writing conditional statements, it is not the proper operator to be used in C programming. Instead, the operator  !=  must be used to indicate “not equal to” condition.

**How do you declare a variable that will hold string values?**

The char keyword can only hold 1 character value at a time. By creating an array of characters, you can store string values in it. Example: “char MyName[50]; ” declares a string variable named MyName that can hold a maximum of 50 characters.

**Can the curly brackets { } be used to enclose a single line of code?**

While curly brackets are mainly used to group several lines of codes, it will still work without error if you used it for a single line. Some programmers prefer this method as a way of organizing codes to make it look clearer, especially in conditional statements.

**What are header files and what are its uses in C programming?**

Header files are also known as library files. They contain two essential things: the definitions and prototypes of functions being used in a program. Simply put, commands that you use in C programming are actually functions that are defined from within each header files. Each header file contains a set of functions. For example: stdio.h is a header file that contains definition and prototypes of commands like printf and scanf.

**What is syntax error?**

Syntax errors are associated with mistakes in the use of a programming language. It maybe a command that was misspelled or a command that must was entered in lowercase mode but was instead entered with an upper case character. A misplaced symbol, or lack of symbol, somewhere within a line of code can also lead to syntax error.

**What are variables and it what way is it different from constants?**

Variables and constants may at first look similar in a sense that both are identifiers made up of one character or more characters (letters, numbers and a few allowable symbols). Both will also hold a particular value.  Values held by a variable can be altered throughout the program, and can be used in most operations and computations. Constants are given values at one time only, placed at the beginning of a program. This value is not altered in the program. For example, you can assigned a constant named PI and give it a value 3.1415  .  You can then use it as PI in the program, instead of having to write 3.1415 each time you need it.

**How do you access the values within an array?**

Arrays contain a number of elements, depending on the size you gave it during variable declaration. Each element is assigned a number from 0 to number of elements-1. To assign or retrieve the value of a particular element, refer to the element number. For example: if you have a declaration that says “intscores[5];”, then you have 5 accessible elements, namely: scores[0], scores[1], scores[2], scores[3] and scores[4].

**Can I use  “int” data type to store the value 32768? Why?**

No. “int” data type is capable of storing values from -32768 to 32767. To store 32768, you can use “long int” instead. You can also use “unsigned int”, assuming you don’t intend to store negative values.

**Can two or more operators such as \n and \t be combined in a single line of program code?**

Yes, it’s perfectly valid to combine operators, especially if the need arises. For example: you can have a code like ” printf (“Hello\n\n\’World\'”) ” to output the text “Hello” on the first line and “World” enclosed in single quotes to appear on the next two lines.

**Why is it that not all header files are declared in every C program?**

The choice of declaring a header file at the top of each C program would depend on what commands/functions you will be using in that program. Since each header file contains different function definitions and prototype, you would be using only those header files that would contain the functions you will need. Declaring all header files in every program would only increase the overall file size and load of the program, and is not considered a good programming style.

**When is the “void” keyword used in a function?**

When declaring functions, you will decide whether that function would be returning a value or not. If that function will not return a value, such as when the purpose of a function is to display some outputs on the screen, then “void” is to be placed at the leftmost part of the function header. When a return value is expected after the function execution, the data type of the return value is placed instead of “void”.

**What is the advantage of an array over individual variables?**

When storing multiple related data, it is a good idea to use arrays. This is because arrays are named using only 1 word followed by an element number. For example: to store the 10 test results of 1 student, one can use 10 different variable names (grade1, grade2, grade3… grade10). With arrays, only 1 name is used, the rest are accessible through the index name (grade[0], grade[1], grade[2]… grade[9]).

**What are comments and how do you insert it in a C program?**

Comments are a great way to put some remarks or description in a program. It can serves as a reminder on what the program is all about, or a description on why a certain code or function was placed there in the first place. Comments begin with /\* and ended by \*/ characters. Comments can be a single line, or can even span several lines. It can be placed anywhere in the program.

**What is debugging?**

Debugging is the process of identifying errors within a program. During program compilation, errors that are found will stop the program from executing completely. At this state, the programmer would look into the possible portions where the error occurred. Debugging ensures the removal of errors, and plays an important role in ensuring that the expected program output is met.

**What does the && operator do in a program code?**

The && is also referred to as AND operator. When using this operator, all conditions specified must be TRUE before the next action can be performed. If you have 10 conditions and all but 1 fails to evaluate as TRUE, the entire condition statement is already evaluated as FALSE.

**What are logical errors and how does it differ from syntax errors?**

Program that contains logical errors tend to pass the compilation process, but the resulting output may not be the expected one. This happens when a wrong formula was inserted into the code, or a wrong sequence of commands was performed. Syntax errors, on the other hand, deal with incorrect commands that are misspelled or not recognized by the compiler.

**What is || operator and how does it function in a program?**

The || is also known as the OR operator in C programming. When using || to evaluate logical conditions, any condition that evaluates to TRUE will render the entire condition statement as TRUE.

**What are preprocessor directives?**

Preprocessor directives are placed at the beginning of every C program. This is where library files are specified, which would depend on what functions are to be used in the program. Another use of preprocessor directives is the declaration of constants.Preprocessor directives begin with the # symbol.

**How do you determine the length of a string value that was stored in a variable?**

To get the length of a string value, use the function strlen(). For example, if you have a variable named FullName, you can get the length of the stored string value by using this statement: I = strlen(FullName); the variable I will now have the character length of the string value.

**Why is C language being considered a middle level language?**

This is because C language is rich in features that make it behave like a high level language while at the same time can interact with hardware using low level methods. The use of a well structured approach to programming, coupled with English-like words used in functions, makes it act as a high level language. On the other hand, C can directly access memory structures similar to assembly language routines.

**What are the different file extensions involved when programming in C?**

Source codes in C are saved with .C file extension. Header files or library files have the .H file extension. Every time a program source code is successfully compiled, it creates an .OBJ object file, and an executable .EXE file.

**What is the difference between the expression “++a”  and “a++”?**

In the first expression, the increment would happen first on variable a, and the resulting value will be the one to be used. This is also known as a prefix increment. In the second expression, the current value of variable a would the one to be used in an operation, before the value of a itself is incremented. This is also known as postfix increment.

**In C language, the variables NAME, name, and Name are all the same. TRUE or FALSE?**

FALSE. C language is a case sensitive language. Therefore, NAME, name and Name are three uniquely different variables.

**What is an endless loop?**

An endless loop can mean two things. One is that it was designed to loop continuously until the condition within the loop is met, after which a break function would cause the program to step out of the loop. Another idea of an endless loop is when an incorrect loop condition was written, causing the loop to run erroneously forever. Endless loops are oftentimes referred to as infinite loops.

**What is a program flowchart and how does it help in writing a program?**

A flowchart provides a visual representation of the step by step procedure towards solving a given problem. Flowcharts are made of symbols, with each symbol in the form of different shapes. Each shape may represent a particular entity within the entire program structure, such as a process, a condition, or even an input/output phase.

**What is a newline escape sequence?**

A newline escape sequence is represented by the \n character. This is used to insert a new line when displaying data in the output screen. More spaces can be added by inserting more \n characters. For example, \n\n would insert two spaces. A newline escape sequence can be placed before the actual output expression or after.

**What are run-time errors?**

These are errors that occur while the program is being executed. One common instance wherein run-time errors can happen is when you are trying to divide a number by zero. When run-time errors occur, program execution will pause, showing which program line caused the error.

**What are control structures?**

Control structures take charge at which instructions are to be performed in a program. This means that program flow may not necessarily move from one statement to the next one, but rather some alternative portions may need to be pass into or bypassed from, depending on the outcome of the conditional statements.

**When is a “switch” statement preferable over an “if” statement?**

The switch statement is best used when dealing with selections based on a single variable or expression. However, switch statements can only evaluate integer and character data types.

**What are structure types in C?**

Structure types are primarily used to store records. A record is made up of related fields. This makes it easier to organize a group of related data.

**is it possible to create your own header files?**

Yes, it is possible to create a customized header file. Just include in it the function prototypes that you want to use in your program, and use the #include directive followed by the name of your header file.

**What are the different data types in C?**

The basic data types are int, char, and float. Int is used to declare variables that will be storing integer values. Float is used to store real numbers. Char can store individual character values.

**In a switch statement, what will happen if a break statement is omitted?**

If a break statement was not placed at the end of a particular case portion? It will move on to the next case portion, possibly causing incorrect output.

**What are pointers?**

Pointers point to specific areas in the memory. Pointers contain the address of a variable, which in turn may contain a value or even an address to another memory.

**What is the use of a semicolon (;) at the end of every program statement?**

It has to do with the parsing process and compilation of the code. A semicolon acts as a delimiter, so that the compiler knows where each statement ends, and can proceed to divide the statement into smaller elements for syntax checking.

https[://www.programmingsimplified.com/c-program-examples](https://www.programmingsimplified.com/c-program-examples)

<https://www.programiz.com/c-programming/examples>