Date
Page

Structure of C program :

Header → #include<stdio.h>

main() → int main()
{

Variable declaration → int a=10;

Body → printf("%d", a);

Return → return 0;
}

Header files inclusion:
The first & foremost component is the inclusion of the Header files in a C program.

A header file is a file with extension .h which contains C function declarations & macro definitions to be shared between several source files.

- stddef.h → Defines several useful types & macros
- stdint.h → Defines exact width integer types.
- stdio.h → Defines core input & output functions

- stdlib.h → Defines numeric conversion function, pseudo-random network generator, memory allocation
- string.h → Defines string handling functions
- math.h → Defines common mathematical functions

2) Main method declaration: The next part of C program is to declare the main function. The syntax to declare the main function is

```
int main ()
{ }
```

3) Variable declaration: The next part of any C program is the variable declaration. It refers to the variables that are to be used in the function. Please note that in the C program, no variable can be used without being declared. Also in a C program, the variables are to be declared before any operation in the function.

```
int main ()
{
    int a;
```

4) Body: The body of a function in the C program refers to the operations that are performed in the functions. It can be anything like manipulations, searching, sorting, printing etc.

```
int main ()
{
    int a;
    printf("%d", a);
```

Return Statement: The last part of any C program is the return statement. The return statement refers to the returning of the values from a function. The return statement & return value depend upon the return type of the function. For example, if the return type is void, then there will be no return statement. In any other case, there will be a return statement & the return value will be of the type of the specified return type.

```
int main ()
{
    ~~inta~~ int a;
    printf("%d", a);
    return 0;
}
```

Writing first C program

~~Step 1:~~

```
#include <stdio.h>
int main(void)
{
    printf("Geeks quiz");
    return 0;
}
```

Line 1: [#include <stdio.h>]
In a C program, all lines that start with # are processed by a preprocessor which is a program invoked by the compiler. In a very basic term, the preprocessor takes a C program & produces another C program. The produced program has no lines starting with #, all such lines are processed by preprocessor. In the above example, the preprocessor copies the preprocessed code of stdio.h to our file. The .h files are called header files in C. The header files contain generally declarations of functions. We need stdio.h for the function printf() used in the program.

Line 2: [int main(void)]
There must be a starting point from where execution of compiled C program begins. In C, the execution typically begins with the first line of main(). The void written in brackets indicates that the main doesn't take any parameter. main() can be written to take parameters also.

Q) Diff. bet" int main() & int main(void)
In C++, there is no diff. -> both are same.
void is considered technically better as it clearly specifies that main can only be called without any parameter.
In C, if a function signature does not specify any ~~argu~~ argument, it means that the function can be called with any number of parameters

The int was written before main indicates return type of main(). The value returned by main indicates the status of program termination.

Line 3 & 6 [{ and }]
In C, a pair of curly brackets define scope & are mainly used in functions & control statements like if, else, loops. All functions must start & end with curly ~~bra~~ braces.

Line 4 [printf("Geeks quiz");]
printf is a standard library function to print something on standard output. The semicolon at the end of printf indicates line termination.
In C, a semicolon is always used to indicate end of a statement

Line 5 [return 0;]
The return statement returns the value from main(). The returned value may be used by an

operating system to know the termination status of your program. The value 0 typically means successful termination

Main() function

→ Every C program must have one main function.
→ The function contains 2 parts declaration part & Executable part.
→ The declaration part declares all the variables used in the executable part.
→ The executable portion of the main function will have 3 types of statements: Input, Output & Processing statements.
→ All C statements end with a semi-colon.

C fundamentals

→ Character set → Identifiers
→ Operators → Expressions

C character set

→ To form words, numbers & expressions, characters are divided into:

A] Letters: (a...z, A.....Z)

B] Digits: (0 .... 9)

C] Special characters: *, %, #, Etc.

D) Escape sequence: non-printable characters

E.g.: \t: Horizontal tab, \n: New line,
\b: backspace, \0: null character

C Keywords

→ C keywords have standard & predefined meaning & used only for intended purpose

→ Keywords are generally written in lower case letters

→ ANSI C supports 32 keywords

1) ~~Au~~ Auto	7) default	13) float	19) register
2) break	8) ~~do~~ Do	14) for	20) return
3) case	9) double	15) goto	21) short
4) Char	10) else	16) if	22) signed
5) const	11) enum	17) int	23) size of
6) continue	12) extern	18) long	24) static

24) ~~long~~ 25) Long 29) unsigned

~~26)~~ 26) Switch 30) Void

27) typedef 31) volatile

28) Union 32) ~~White~~ While

C Identifiers

→ Identifiers are used to name the programming elements like variables, constants, functions, arrays & structures.

→ It starts with an alphabet or an underscore & followed by alphabets, digits or underscore

→ It does not contain any special characters.

→ E.g: temp, Distance, M1, a123, _temp, a_10

→ Array declaration: int temp[20];

→ char abc;

Data types

It defines

→ The type of value represented or stored in a variable,

→ Number of bytes to be reserved for variable

→ Range of values on any

Two Data Types

Premitive Data types:

→ int: Integer data (2/4 bytes)

→ float: floating point numbers (4/8 bytes)

→ char: stores a single character (1 byte)

→ double: double precision real nos (8 bytes)

→ E.g. of int: 2, 30, 4567, 32584

Data Type Qualifiers

→ short: may require less bytes (int: 1/2)

→ long: may require more bytes (int: 2/4)

→ Signed: -32,564 to +32,564

→ Unsigned: 0 to 64,000

- long int a;
- signed int b;
- unsigned int c;
- float b;
- Print b; 34.0

Derived Data Types

Unsigned char → Unsigned character (positive)

signed char char → Represents single character

unsigned int unsigned short int → Represents positive integer numbers

Short

signed short short int → Represents both positive & negative integer quantity

signed short int

unsigned ~~long~~ long → represents positive long integer

signed long → represents both positive & long int negative long integer

float → Floating point number

Double → A more accurate floating point number than float

long double → Increases the size of double

void → Defines an empty data type which can then be associated with some data types. It is useful with pointers

Derived Data Types

Derived from primitive data types

→ Arrays: int roll [60]

→ Structure

→ Union

→ Pointers

Variables

→ Variable is an identifier that represents some value
→ Value of the variable can be changed during execution of program
→ Variable declaration:

: data type variable name;

E.g. int a;
float b;
char c;

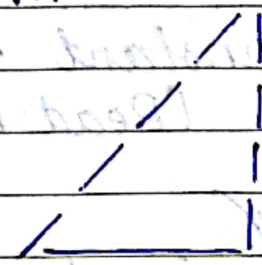
DRAWING SHAPE

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

int main()
{
    printf("   /|\n");
    printf("  / | \n");
    printf(" /  | \n");
    printf("/___|\n");

    return 0;
}
```

Soln:



Variable Initialization

Syntax

data type variable_name = value

- This is initial value
- This value can change during program execution

```
int a = 10;
float b = 5.7;
char c = 'A';
char d = '1';
char p = '&';
```

Constants

It refers to fixed value of the variable which cannot change during program execution.

Types:
1) Symbolic constants 2) Constant variables (Read only)

Symbolic Constants

→ Defined on preprocessor area

→ It's value is constant throughout the program

→ Usually, represented on upper case letters.

Constant Variables

→ A constant variable is declared & initialized in the variable declaration section of the program using Keyword ~~const~~ 'const'.

→ It cannot be modified thereafter.

E.g. const int a=25

const char b='G'

C Operators

→ Operators are used to manipulate data:

⇒ Arithmetic Operators

⇒ Relational Operators

⇒ Logical Operators

⇒ Assignment Operators

⇒ Ternary/Conditional Operator

⇒ Bitwise Operators

Arithmetic Operators

+ → Addition

− → Subtraction

* → Multiplication

/ → Division

% → Modulus

Relational operators: It evaluates to True or False

< → Less than

<= → Less than or equal to

> → Greater than

>= → Greater than or equal to

== → Equals to of (a!=b)

!= → not equals to

Logical Operators

&& → Logical AND (true only if both the operands are true)

|| → Logical OR (true if one operand is true)

! → Logical NOT (negate the operand)

Assignment Operators

= Assignment operator which assign a value to an identifier

Compound assignment operators

+=, *=, -=, /=, %=

+= → 1st Addn, *= → 1st Multi, -= → 1st Subn, /= → 1st div., %= → 1st Mod.

E.g. a = a+b equals to a += b

p*=10 ; p=p*10;

Ternary operator / Conditional operator

Notation: ?:

E.g.: Max=(a>b)?a:b OR

(a>b)?printf("a is >"):printf("b is >");

In the above statement, if condition is evaluated to true, the value of variable a will be assigned to variable big else b will be assigned.

Bitwise operators

& Bitwise AND

| Bitwise OR

<< Left shift

>> Right shift

Increment & Decrement operators

→ ++ may be in the form of pre-increment or post increment

++a : pre increment , a++ ; post increment

E.g.

```
int a=10
printf("%d", ++a); /* prints 11-pre-increment */
printf("%d", a++); /* prints 11-post increment*/
printf("%d", a); /* prints 12 */
```

Special Operators

& → Address operator

* → Indirection operator

, → Comma operator

Size of () Size of operator (size of(int) = 2 bytes)

Order of Precedence

→ All the operators have its precedence & associativity.

→ High priority operators are evaluated prior to lower priority ones

→ Operators of the same priority group are evaluated from left to right fashion

Order of Precedence Table

Operator	Name	Association
()[]→.	Parentheses, Index, Member access operators	L to R
! - sizeof() (Typecast) * &	Logical NOT, unary minus, indirection, address	R to L
++ --	Increment & Decrement Operators	R to L
* / %	Multiplicative operators	L to R
+ -	Additive operators	L to R
<><=>=	Inequality comparators	L to R

== !=	Equality comparators	L to R
&&	Logical AND	L to R
||	Logical OR	L to R
?.	Conditional	R to L
= op =	Assignment	R to L
,	Comma	L to R

Format Specifiers

→ ~~% c~~ → % c → character

% d → Integer % u → unsigned

% f → float

% lf → double

% s → string

% ld → long integer

% o → octal

% x → hexadecimal

Formatted Input Statement

scanf(): scanf() function is used to read formatted data items.

Syntax:
scanf("format string", list of variables);

E.g.
scanf("%d%f", &a, &b);

List of variables specify the address of memory locations where the data is to be stored. Address operator (&) is used before the variable

Formatted Output

printf():
→ printf() function is used to output the values.
→ This function returns the characters printed

Syntax

→ printf("format string", list of variables);

E.g.
printf("Hello World")
printf("Max=%d Men=%d", p, q);

C programming Constructs

3 types:

- Sequence: The instructions are executed in the same order in which they appear in the program.
- Selection: The control flow can be altered by evaluating conditions
- Iteration: A group of instructions is executed repeatedly, until some condition is satisfied.

Sequence Construct Examples

- Declaration statements
- Input / Output statements
- Simple Expression statements

Date
Page

Selection Statements

A group of instruction is executed repeatedly, until some condition is satisfied

→ "If" statement → Conditional / Ternary operator (?:)

→ Switch statement

If - Decision making statement

The If statement has 3 basic forms

→ Simple If-else
→ Nested If
→ If-else If ladder

Q What is programming?

Computer programming is a medium for us to communicate with computers. Just like we used 'Hindi' or 'English'. to communicate with each other programming is a way for us to deliver our instructions to the computer.

C was developed by Dennis Ritchie at AT&T Bell Labs, USA in 1972.

Uses of C

C language is used to program a wide variety of systems, some of the uses of C are as follows

→ Major parts of Windows, Linux & other operating systems are written in C.

→ C is used to write driver programs for devices like tablets, printers etc.

→ C language is used to program embedded systems where programs need to run faster on limited memory. (Microwave, Cameras etc.)

Ch 1: Variables, Constants & Keywords

Variables

A variable is a container which stores a 'value'. In kitchen we have containers storing Rice, Dal etc. Similar to that variables in C stores value of a constant. E.g.

```
a = 3 ;     // a is assigned 3
b = 4.7 ;   // b is assigned 4.7
c = A ;     // c is assigned A
```

Rules for naming variables in C

1) First character must be an alphabet or underscore (_)

2) No commas, blanks allowed.

3) No special symbol other than (_) allowed

4) Variable names are case sensitive. (Lower case & Higher case diff.)

We must create meaningful variable names in our programs. This enhances readability of our programs.

Constants

An entity whose value doesn't change is called as a constant.

A variable is an entity ~~which~~ whose value can be changed

Types of Constants

Primarily there are 3 types of constants

1) Integer Constant → -1, 6, 7, 9

2) Real Constant → -322·1, 2·5, 7·7

3) Character Constant → 'a', '$', '@' [Must be enclosed with single inverted comma].

Keywords

These are reserved words whose meaning is already known to compiler. There are 32 keywords available in C.

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

→ Every program execution starts with main function.
→ All statements are terminated with semi-colon.
→ Instructions are case sensitive.
→ Instructions are executed in the same order in which they are written.

Comments

Comments are used to clarify something about the program in plain language. It is a way for us to add notes to our program. There are 2 types of comments in C.

1. Single line comment: // This is a comment
2. Multi-line comment: /* This is a multi-line comment */

Comments in a C program are not executed & are ignored.

Date
Page

Compilation & Execution

first.c ⟶ C Compiler ⟶ first.exe

In VS code — gcc

A compiler is a computer program which converts a C program in to machine language so that it can be easily understood by the computer

A C program is written in plain text. The plain text is combination of instructions in a particular sequence. The compiler performs some basic checks & finally converts the program into an executable function.

Library Functions

C language has a lot of valuable ~~li to~~ library functions which is used to carry out certain tasks. for instance printf function is used to print values ~~for the~~ on the screen.

```
printf("The is %d", i);
```

%d for integers
%f for real values
%c for characters

Date
Page

Q Code

```
#include <stdio.h>

int main()
{
  int a = 4;
  //int b = 8.5; Not recommended because 8.5 is not a integer */

  float b=8.5;

  char c='u';
  int d = 45;
printf("The value of a is %d \n", a);
printf("The value of b is %f \n", b);
printf("The value of c is %c \n", c);
printf("The sum of a & d is %d \n",a+d);
 return 0;

}
```

Soln:

The value of a is 4
The value of b is 8.5
The value of c is u
Sum of a & d is 49

Types of variables

1) Integer variables → int a = 3;
2) Real variables → ~~int a = 7.7;~~ float a = 7.7

3) Character Variables → char a = 'B';

Receiving input from the User

In order to take input from the user & assign it to a variable, we use scanf function

Syntax for using scanf:

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

& is the "address of" operator & it means that the supplied value should be copied to the address which is indicated by variable i.

Code

```
#include <stdio.h>

int main()
{
    int a, b;
    printf("Enter the value of a \n");
    scanf("%d", &a);

    printf("Enter the value of b \n");
    scanf("%d", &b);

    printf("The sum of a & b is %d", a+b);

    return 0;
}
```

Soln:
Enter the value of a
☐ → You can enter any value here
For. e.g. 3
Enter the value of b
☐ → You can enter any value here
For. e.g. 7

The sum of a & b is 10

Q1 Write a C program to calculate area of a rectangle
(a) Using Hard coded inputs
Code:
```
#include<stdio.h>

int main(){
    int length=3, breadth=8;
    int area = length * breadth;
    printf("The area of the rectangle is %d", area);

    return 0;
}
```

Ans:
The area of the rectangle is 24

(b) By using inputs supplied by user
Code:
```
#include<stdio.h>

int main()
{
  int length, breadth;
```

```
printf("What is the length of rectangle\n");
scanf("%d", &length);

printf("What is the breadth of rectangle\n");
scanf("%d", &breadth);

printf("The area of your rectangle is %d",
                    length * breadth);

    return 0;
}
```

Ans.

What is the length of rectangle
☐ For. e.g 6

What is the breadth of rectangle
☐ For. e.g. 2

The area of your rectangle is 12

Q2 Calculate the area of a circle and modify the same program to calculate the volume of cylender given its radius & height

```
#include<stdio.h>

int main()
{
    int radius=3;
    float pi=3.14;
```

```
printf("The area of the circle is %.f\n
        pi * radius * radius);
int height = 3;
printf("Volume of the cylender is %.f\n",
        pi * radius * radius * height);

return 0;

}
```

Ans:

The area of the circle is 28.26

The Volume of the cylender is 84.78

Q3 Write a program to convert Celsius to farenheit

```
#include <stdio.h>

int main()
{
    float celsius, far;

    far = celsius * 9/5 + 32;

printf("The value of the celsius temperatu
    in Fah farenheit is %.f", far);

    return 0;
}
```

Ans:

The value of the celsius temperature in Farenheit is 98.599998

Q.4 Write a program to calculate simple interest for a set of values representing principal no. of years & rate of interest

Code:

```
#include<stdio.h>

int main()
{
int principal=100, rate=4, years=1;
int simple Interest=(principal*rate*years)/100;

printf("The value of simple Interest is %d",
                                simple Interest);

  return 0;
}
```

Ans:

The value of simple interest is 4