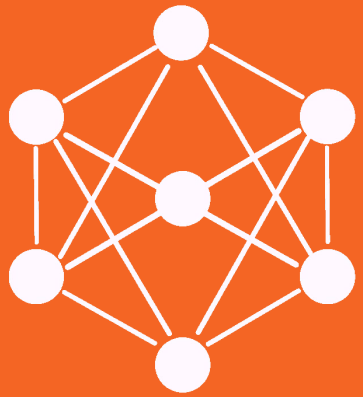


Infero

The Algorithmic and Logical Reasoning Club

Introduction To C

Computer

- **Computer** - Device capable of performing arithmetic and logical operations.
- **Computer Programs** - Set of Instructions run by computer to perform a specified task.
- Computers do not understand the language humans use - deal only with machine language - 0s and 1s.
- Instructing the computer in machine language very difficult for humans.

Computer Languages - Types

- **Machine Language**
  - Contains binary or hexadecimal instructions that computer follows directly.
- **Assembly Language**
  - Symbolic instruction code which is converted to machine language by assemblers.
- **High Level Language**
  - Close to everyday english and generally translated into machine language by compilers or interpreters.
  - E.g. : C, C++, Python, Java, etc.

C

- High level programming language.
- Hardware independent (portable).
- Developed by Dennis M. Ritchie in 1972 at Bell Labs.
- C was used to develop UNIX operating system.
- Many software tools are written in C.

Algorithm

- Step by step procedure for solving a problem
- Convert algorithm to code in a programming language
- Steps of a program
  - Input
  - Processing
  - Output
- Algorithms in daily life - using a telephone

Writing a C program

- Extension of C source code - “.c”
- Create a text file.
- Use editors such as sublime,gedit etc.
- Open Terminal and type “ **`gedit filename.c`** ”. (Commands in terminal are case sensitive)
- If using **Sublime Text** type “**`subl filename.c`** ” in terminal.
- This file is known as the source code.

Hello World! in C

```
#include <stdio.h>
  int main()
   {
       printf(“Hello World!\n”);  // prints - Hello World!
       return 0;
   }
```

Compiling

First of All what is a Compiler ?

a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses.

Compiling/Running a C programme :

- Save the code first :P
- Open terminal and go to the path where your '.c' file is located.
- Write '`gcc filename.c -o objectfile`'
- Repeat until you don't get any errors.
- Run file using '`./objectfile`'

Hello World! in C - explained

- `#include <stdio.h>` - preprocessor
- `int main()` - main function that is first called by the OS
- `printf("Hello World!\n");` - Prints Hello World! to screen.
- `return 0;` - returns 0 to the OS and program exits

The `#include` preprocessor

- Set of functions for executing a set of tasks are grouped into header files Have **.h** extension
- Use **`#include <filename>`**
- Example - **`stdio.h`** header file contains function printf() to display to the standard output - screen
- **`#include <stdio.h>`** tells the compiler to include the code from the header file into the program

The `main()` function

- Every C program starts execution from the main() function

```
int main()
{
//Your code here
}
```

- Curly braces { } indicate a block of code
- int - return data type
- Comment in C

  Single line comment : `// This is a comment`

  Block comment : `/* This is a comment */`

The `printf()` function

- Defined in stdio.h
- Used for printing (displaying) characters to the computer screen
  `printf("Hello World!\n");`
- Every valid statement in C ends with a semicolon `;`
- The above will print Hello World! to the screen
- `\n` indicates that after printing Hello World!, printing will move to a new
- line - similar to pressing Enter/Return key

Exercise

- Write a program to print your name, roll number and branch on the screen, in three separate lines

Variables in C

- Variables represent storage units in a program.
- Variable name in C is composed of letters, digits or underscore and cannot start with a digit.
- Variable declaration in C

  `type var_name;`

  E.g. : `int a;`
- Variable initialization in C

  `type var_name;`

  E.g. : `int a;`

  E.g. : `a = 5;`

  Combined E.g. : `int a = 5;`

Data Types in C

- The datatype of a variable determines how much space it occupies in storage and how the stored bits in the memory are interpreted.
- Four Basic data types in C

```
char
int
float
double
```

- Type Modifiers : short, long, signed, unsigned

E.g. : `int num = 5;`

Here we declare a variable num with int data type and store the value 5 in it.

Data Modifiers

Modifiers are prefixed with basic data types to modify (either increase or decrease) the amount of storage space allocated to a variable.

- `short`
- `long`
- `signed`
- `unsigned`

Data Type	Memory (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4		%f
double	8		%lf
long double	12		%Lf

Printing data from variables

- Suppose we have an integer variable var with some value
- We want to print Value of var is followed by the value of var
- `printf("Value of var is %d",var);`
- %d - format specifier
- Value of var replaces %d in the output
- For multiple variables, say var1 and var2 - separate variables by commas
- `printf("Value of var1 is %d and value of var2 is %d",var1,var2);`
- `printf("String",var1,var2,...);`

Format Specifiers

- Specify the type of the data - to be printed, in this case
- **`%c`** - character, **`%d`** - integer, **`%f`** - float, **`%s`** - string
- In a **`printf`** statement, it tells that the next argument of **`printf`** is of a particular format
- Suppose you want to display the cost of x apples, given that cost of **1 apple = y rupees**. **x** is **int** and **y is float.** Let cost of **x apples = z =x*y rupees**
- Want to display Cost of x apples is **z rupees**
- **`printf("Cost of %d apples is %f",x,z);`**

The `scanf()` function

- In the previous programs, the values of the variables were pre-decided.
- Useful if the user could decide the input - for example, to find the simple- interest based on data provided by the user
- `scanf()` function accepts input from the user and stores it in a variable
- Defined in `stdio.h`

```
scanf(format specifier,&var1,&var2,...);
```

- For example, to take an integer and store it in an int variable var

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

Type Casting/Cascading

Type casting is a way to convert a variable from one data type to another data type.
New data type should be mentioned before the variable name or value in brackets which to be typecast.
For Example -

```
#include<stdio.h>
 int main()
{
    float x;
    x = (float) 7/5;
    printf("%f",x);
}
```

Operators

- Arithmetic operators : **+ , - , * , / , % , ++ , --**
  Example - **a+b**
- Relational operators : **== , < , > , <= , >= , !=**
  Example - a==b
- Logical operators : **&& , | | , !**
  Example - **a>3 && a<6**
- Bitwise operators : **& , | , ^ , << , >> , ~**

Program to add two integers-with user input

```
#include<stdio.h>
int main()
{
  int a,b,c;   // Declaring Variables
  printf("Enter the values of a and b:\n");
  scanf("%d %d",&a,&b);  // Taking Input
  c=a+b;
  printf("The sum of %d and %d is %d",a,b,c);
//Output
  return 0;
}
```

Exercise

- Write a program to set the principal (int, in rupees), rate (int, in %) time period (int, in years) and compute the simple interest.

Challenge:

- Find the minimum of two integers without using if-else statements.

Arrays

- An array is a container object that holds a fixed number of values of a single type.

OR

- A collection of elements (values or variables), each identified by at least one array index or key.
- Arrays can be Linear (1-D),2-D etc.

Array as mentioned is collection of data types, An 1-D integer array is declared as

```
int array[100];    // here 100 is length of our array
```

Conditional Statements

A conditional statement lets us choose which statement will be executed next.
Conditional statements give us the power to make basic decisions.

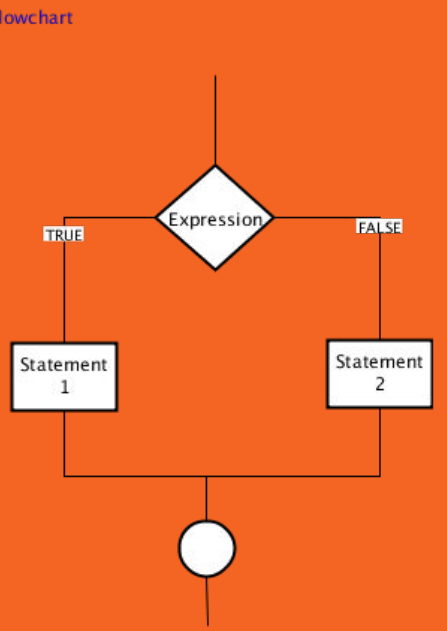
The **C** conditional statements are the:

- **`if`** statement
- **`if-else`** statement
- **`switch`** statement

The if Statement

Syntax :-

```
if (condition)
   {
      // statements
   }
```



`if-else` statements

Example -

```
if (a == b)
    printf ("%d is equal to %d",a,b);
else
    printf ("%d is not equal to %d",a,b);
```

A complete example on conditional `if-else` statement:-

```
#include<stdio.h>
int main()
 {
    int num;
    printf("Input a number : ");
    scanf("%d",&num);
    if(num>0)
      {
         printf("This is a positive integer\n");
      }
    else // else portion of if statement
      {
         printf("This is a negative integer.\n");
      }
    return 0;
 }
```

Sequential if-then statements :-

```
if (a == b)
   printf ("a = b");
if (a == c)
   printf ("a = c");
if (b == c)
   printf ("b = c")
```

else if

Multiway if-else-Statement :-

```
if (expression_1)
   statement_1
else if  (expression_2)
   Statement_2
else if (expression_n)
   statement_n
else
   other_statement
```

```
#include<stdio.h>
int main()
{
  int num1=5, num2=3, num3=-12, min;
   if(num1<num2) {
                 if(num1<num3)
                      min = num1;
                 else
                      min = num3;
    }
  else{
                  if(num2<num3)
                       min = num2;
                 else
                     min = num3;
    }
  printf("Among %d, %d, %d minimum number is %d",num1,num2,num3,min);
}
```

Nested if-else-Statement :

`switch` Case

The switch-case statement is a multi-way decision statement. Unlike the multiple
decision statement that can be created using if-else, the switch statement evaluates the conditional expression and tests it against numerous constant values.

Syntax Of `switch` Case

```
switch (<variable>) {
case this-value: Code to execute if <variable> == this-value
      break;
case that-value: Code to execute if <variable> == this-value
      break;
default: Code to execute if <variable> != any of the above
cases
      break;
}
```

```
# include <stdio.h>
int main() {
char operator;
double firstNumber,secondNumber;
printf("Enter an operator (+, -)");
Example
scanf("%c", &operator);
printf("Enter two operands: ");
scanf("%lf %lf",&firstNumber, &secondNumber);
switch(operator)
{
case '+':
printf("%.1lf + %.1lf = %.1lf",firstNumber, secondNumber,
firstNumber+secondNumber);
break;
case '-':
printf("%.1lf - %.1lf = %.1lf",firstNumber, secondNumber,
firstNumber-secondNumber);
break;
// operator is doesn't match any case constant (+, -,)
default:
printf("Error! operator is not correct");
}
return 0;
}
```

Loops

- Used for doing something again and again
- Code inside loop runs repeatedly till condition given to the loop becomes false

Three types -

- `for` loop
- `while` loop
- `do-while` loop

for Loop

- Usually used when number of times the loop needs to run - number of iterations - is known beforehand

```
for(<initialization>, <condition>, <update>)
{
//some code
}
To print the first n natural numbers -
for(int i=1;i<=n;i++)
{
printf("%d ",i);
}
```

- **i** is called the iterator.

while Loop

- while(<condition>)

```
{
//some code
}
```

- Example -

```
int i=1;
while(i<=10)
{
printf("%d ",i);
}
```

do-while loop

```
do
{
//some code
}while(<condition>)
int i=1;
do
{
printf("%d ",i);
}while(i<=10);
```

- Used when at least one iteration of the loop must occur

Empty loops and Infinite loops

- `for(i=0;f(i)<0;i++) {}` or
  `for(i=0;f(i)<0;i++);`

- ```
  for(i=0;i>=0;i++)
  {
  //some code that does not modify
  the value of i
  }
  ```
  ```

`break` And `continue`

- The **`break`** statement terminates the loop (for, while and do...while loop) immediately when it is encountered.
- The **`continue`** forces the next iteration of the loop to take place, skipping any code in between.

Exercise

- Write a program to accept an integer n from the user, where 1<=n<=20. Print the multiplication table of n. If the user gives an invalid input for n, print an error message.
- Write a program to accept a sequence of numbers from the user, and find
  the mean(average), maximum, and minimum of the numbers entered. Keep
  accepting numbers from the user till he enters a 0, after which find the above values.
- Write a program to input a number and check if it is a prime.

Functions

- A function is a set of statements that take inputs, do some specific computation and produces output.
- A function declaration tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.
- Every C program has at least one function, which is `main()`.
- The general form of a function definition in C programming language is as follows –

```
return_type function_name( parameter list )
{
        // body of the function
}
```

Functions

- A function definition in C programming consists of a function header and a function body. Here are all the parts of a function −
  1. **Return Type** − A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the **return_type** is the keyword void.
  2. **Function Name** − This is the actual name of the function. The function name and the parameter list together constitute the function Signature.

Functions

3. **Parameters** – A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no Parameters.

4. **Function Body** – The function body contains a collection of statements that define what the function does.

Example -

```
/* function returning the max between two numbers */
int max(int num1, int num2) {
/* local variable declaration */
int result;

if (num1 > num2)
                result = num1;
else
                result = num2;

 return result;
}
```

Strings in C

- Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character `'\0'`.
- Declaration of strings: Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string.
  ```
  char str_name[size];
  ```
- Please keep in mind that there is an extra terminating character which is the Null character (`'\0'`) used to indicate termination of string which differs strings from normal character arrays.

Strings in C

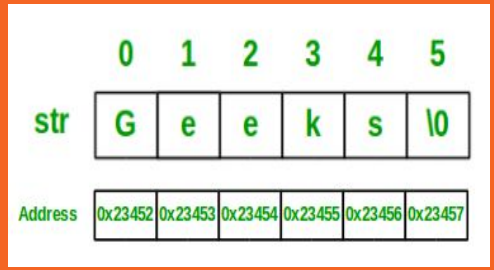
Initializing a String: A string can be initialized in different ways. We will explain this with the help of an example. Below is an example to declare a string with name as str and initialize it with **“GeeksforGeeks”.**

```
1. char str[] = "GeeksforGeeks";

2. char str[50] = "GeeksforGeeks";

3. char str[] =
{'G','e','e','k','s','f','o','r','G','e','e','k','s','\0'}
;

4. char str[14] =
{'G','e','e','k','s','f','o','r','G','e','e','k','s','\0'}
;
```



Strings in C

Let us now look at a sample program to get a clear understanding of declaring and initializing a string in C and also how to print a string.
// C program to illustrate strings

```
#include<stdio.h>
 int main()
{
    // declare and initialize string
    char str[] = "Geeks";
     // print string
    printf("%s",str);
     return 0;
}
```

Strings in C

Below is a sample program to read a string from user:
// C program to read strings

```
#include<stdio.h>
 int main()
{
    // declaring string
    char str[50];
    // reading string
    scanf("%s",str);
     // print string
    printf("%s",str);
    return 0;
}
```

- You can see in the above program that string can also be read using a single scanf statement. Also you might be thinking that why we have not used the '&' sign with string name 'str' in scanf statement! To understand this you will have to recall your knowledge of scanf. We know that the '&' sign is used to provide the address of the variable to the scanf() function to store the value read in memory. As str[] is a character array so using str without braces '[' and ']' will give the base address of this string. That's why we have not used '&' in this case as we are already providing the base address of the string to scanf.

Strings in C

Passing strings to function: As strings are character arrays, so we can pass strings to function in a same way we pass an array to a function. Below is a sample program to do this:

```
// C program to illustrate how to
// pass string to functions

#include<stdio.h>
 void printStr(char str[])
{
    printf("String is : %s",str);
}
int main()
{
    // declare and initialize string
    char str[] = "GeeksforGeeks";
     // print string by passing string
    // to a different function
    printStr(str);
     return 0;
}
```

Important Predefined functions in C

Libraries containing important functions :

- `math.h`
- `string.h`
- `stdlib.h`
- `ctype.h`
- `stdio.h`

math.h

- Includes Math based functions.
- Don't forget to add flag **`-lm`** while compiling

Some Mathematical Library Functions

Function	Header file	Argument	Result	Example
abs(x)	<stdlib.h>	int	int	abs(-5) is 5
fabs(x)	<math.h>	double	double	fabs(-2.3) is 2.3
sqrt(x)	<math.h>	double	double	sqrt(2.25) is 1.5
exp(x)	<math.h>	double	double	exp(1.0) is 2.71828
log(x)	<math.h>	double	double	log(2.71828) is 1.0
log10(x)	<math.h>	double	double	log10(100.0) is 2.0
pow(x,y)	<math.h>	double, double	double	pow(2.0,3.0) is 8.0 returns x^y
sin(x)	<math.h>	double	double	sin(PI/2.0) is 1.0
cos(x)	<math.h>	double	double	cos(PI/3.0) is 0.5
tan(x)	<math.h>	double	double	tan(PI/4.0) is 1.0
ceil(x)	<math.h>	double	double	ceil(45.2) is 46.0
floor(x)	<math.h>	double	double	floor(45.2) is 45.0

string.h

- Contains functions related to strings

String in C – Library Functions

Function	Purpose	Example
strcpy	Makes a copy of a string	strcpy(s1, “Hi”);
strcat	Appends a string to the end of another string	strcat(s1, “more”);
strcmp	Compare two strings alphabetically	strcmp(s1, “Hu”);
strlen	Returns the number of characters in a string	strlen(“Hi”) returns 2.
strtok	Breaks a string into tokens by delimiters.	strtok(“Hi, Chao”, “ ,”);

ctype.h

Prototype	Description
int isdigit(int c)	Returns **true** if **c** is a digit and **false** otherwise.
int isalpha(int c)	Returns **true** if **c** is a letter and **false** otherwise.
int isalnum(int c)	Returns **true** if **c** is a digit or a letter and **false** otherwise.
int isxdigit(int c)	Returns **true** if **c** is a hexadecimal digit character and **false** otherwise.
int islower(int c)	Returns **true** if **c** is a lowercase letter and **false** otherwise.
int isupper(int c)	Returns **true** if **c** is an uppercase letter; **false** otherwise.
int tolower(int c)	If **c** is an uppercase letter, **tolower** returns **c** as a lowercase letter. Otherwise, **tolower** returns the argument unchanged.
int toupper(int c)	If **c** is a lowercase letter, **toupper** returns **c** as an uppercase letter. Otherwise, **toupper** returns the argument unchanged.
int isspace(int c)	Returns **true** if **c** is a white-space character—newline ('**\n**'), space (' '), form feed ('**\f**'), carriage return ('**\r**'), horizontal tab ('**\t**'), or vertical tab ('**\v**')—and **false** otherwise
int iscntrl(int c)	Returns **true** if **c** is a control character and **false** otherwise.

How to write a good code

- Proper Indentation
- Use Comments to get to know what that line does.
- Use descriptive variable names, rather than just x and n etc.
- Use lots of parentheses and/or spaces in expressions
- Use shotgun initialization
- Place all local declarations at the start of functions
- Never use goto