C TUTORIALS FOR BEGINNER SLIDEBOOK

AUTHOR

- Author Name: Mahmoud said (محمود سعيد)
- Exist: Egypt, Alex
- Works: instructor in Computer science
- Full stack developer
- Logic design and algorithms
- Contact Author: https://www.facebook.com/mahmoud.said.NST
- The Following content is provide New System Technology Training Organization
- https://www.facebook.com/NSTechnologyO

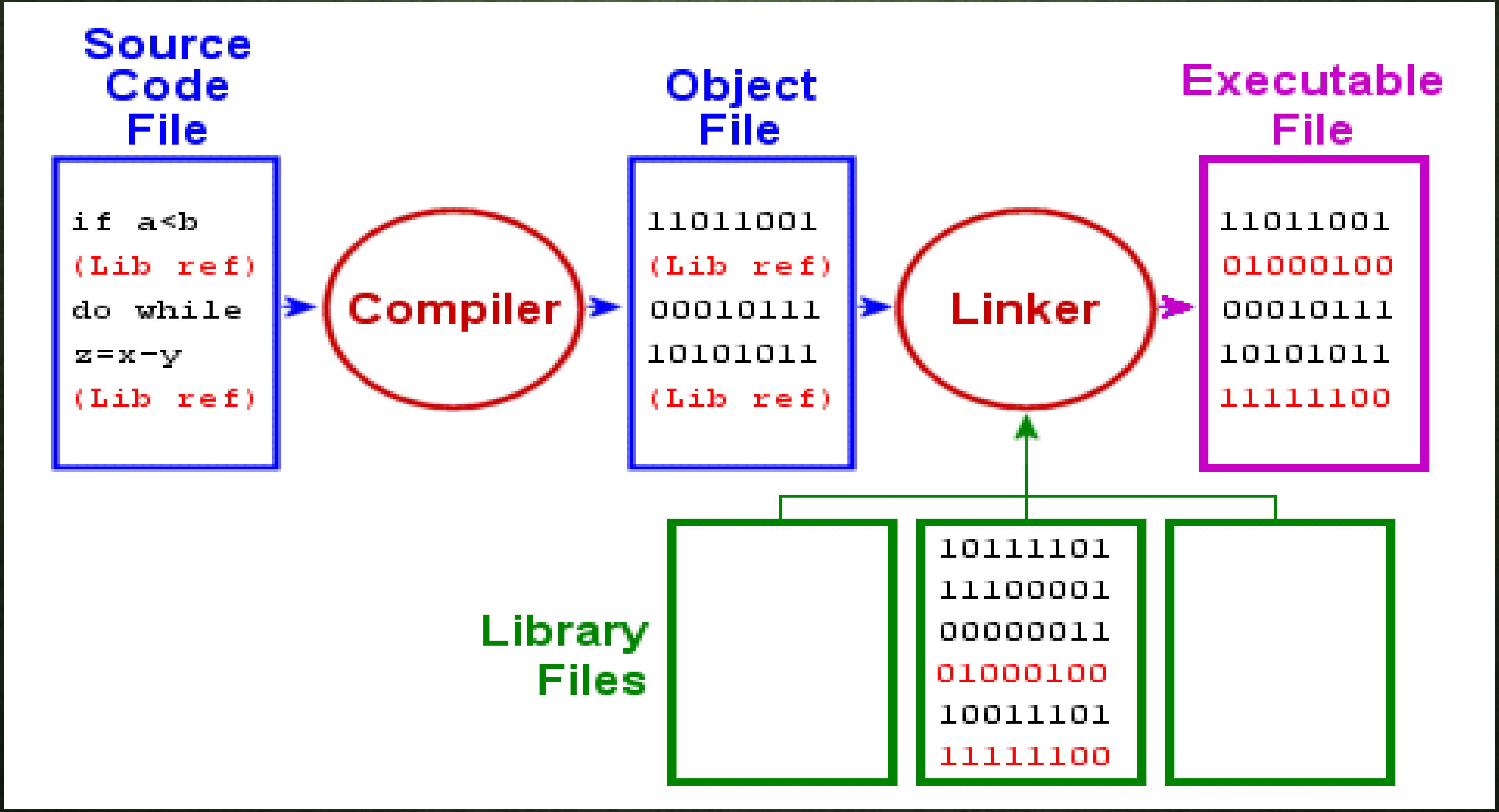
INTRODUCTION

- C programming language is a high-level language, it was first developed by Dennis M. Ritchie at Bell labs.
- C is one of the most commonly used programming language because it simple and easy to learn and efficient and more advantage.
- Can you build many big project by c as commonly application daily you use it or operating systems and based on to other programming language like (c++,java, python, c#, etc…)

WHAT IS ADVANTAGES OF C?

- Easy to learn
- It is portable
- Fast, powerful and efficient
- solving problems for software or hardware
- C is the native language of Linux/Unix OS
- Easy connecting with system devices

C, HOW TO COMPILE



STRUCTURE OF PROGRAM IN C

- This program print hello tech.
-
```
#include <stdio.h>
int main()
{
        printf(“hello Tech”);
        return 0;
}
```

1. #include : this is a preprocess to tell compiler link this program with library and including all of codes to run it and stdio.h file it’s file have lot of codes called shared library.
2. Int is a return data type it’s short name for integer number.
3. Main() : this main function to executes program c programs cloud consist of one or more functions, but only function called main.
4. Printf() : it is function to print on user screen, return 0; return value 0 to operating system to terminate program, (;) semicolon write to end any statement.

CASE SENSITIVE LANGUAGE

- C is a case sensitive language, so the names of the function must be typed in lower case if it actually lower.
- We can use white spaces, taps and new line characters to make our code easy to read.

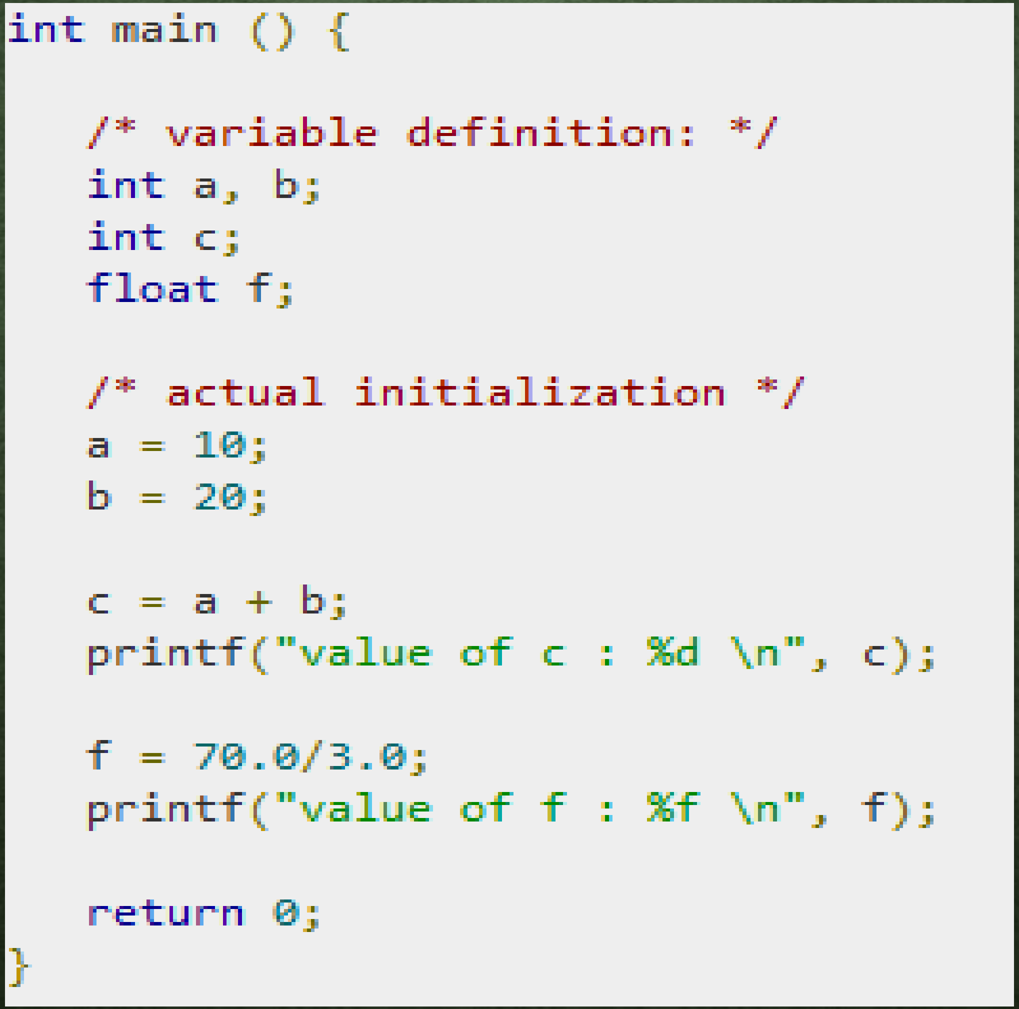
VARIABLES

- Variables is a visual space allocate memory items that you can access memory to write and read actually data in short memory.
- All of variables pointer of address in memory to keep data and access at run time.
- Variables has more of types called Data Type to give programmer flexibility to interact with end-user data
- Variables name has some rule as :-
  - Can't use any character like(!,@,#,%,^,&) but you can use _ or   **$**
  - can't use numbers in first like (1hello,8wow) but you can use number within word like hello1,go2o
- Variables is important for programming or for any equation or functions.

DATA TYPE

- Short : from min -128 to max 127
- Unsigned short from 0 to 255
- Char : from 0 to 255
- Signed char -128 to 127
- Int from -2147483648 to 2147483647
- unsigned int from 0 to 4294967295
- Float : +/- 3.4e +/- 38 (~7 digits)
- Double : +/- 1.7e +/- 308 (15 digits)
- short int from -32768 to 32767
- long int from -9,223,372,036,854,775,808   to   9,223,372,036,854,775,807

USING VARIABLES



```
int main () {

   /* variable definition: */
   int a, b;
   int c;
   float f;

   /* actual initialization */
   a = 10;
   b = 20;

   c = a + b;
   printf("value of c : %d \n", c);

   f = 70.0/3.0;
   printf("value of f : %f \n", f);

   return 0;
}
```

OPERATORS

- An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators.

- Arithmetic Operators (+, −, *, /,%)
- Relational Operators (==, !=, >, <, >=, <=)
- Logical Operators (&&, ||, !)
- Bitwise Operators (&, |, ^, ~, <<, >>)
- Assignment Operators (=, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, |=)
- Misc Operators (sizeof(), &, *, ? :)

USING OPERATORS

- Asthmatic : int a = 5 + 3;
- Relational : int A = 3 , B = 3;   (A == B) is true.
- Logical : (A && B) is true.
- **Assignment : C = A + B will assign the value of A + B to C**
- C += A is equivalent to C = C + A
- Misc : sizeof(a), where a is integer, will return 4.

INPUT/OUTPUT

- Formatted output – printf
- It takes text and values from within the program and sends it out onto the screen
- Printf(“%f is your weight”,w);
- W is variable store value to print
- %f meaning that a floating value to print
- Formatted input – scanf
- Scanf is function used to get input from user and to store it in the specified variable
- Scanf(“%d”,&x);
- Read integer from keyboard and store value in memory address of the variable x

INPUT/OUTPUT

- getchar or putchar are used for the input or output only one character.
- Character io – getchar
- It returns int which is either EOF or next character in the standard input stream
- Character io – putchar
- Puts the character on the output stream like putchar(c);
- Int main(){

```
        int a;
        a = getchar();    // get input user data
        putchar(a);       // output user data
        return 0;
}
```

EXAMPLE OF INPUT/OUTPUT

```
• int main(){
           char name[20];
                      printf("please Enter your name : ");
                      scanf("%s",name);
                      printf("Welcome MR.%s", name);
          return 0;
}
```

CONTROL FLOW

- If condition
- If else condition
- Nested if
- Switch
- While
- Do while
- For loop
- Nested for loop
- For each loop

CONTROL FLOW – IF CONDITION

- The ability to control the flow of your program, letting it make decisions on what code to execute
- Without a conditional statement such as the if statement, programs would run almost the exact same way every time, always following the same sequence of function calls.
- Example :
  if (55 < 100)

  printf("55 is now less than 100");
- If condition check is true then code will execute if false not excite and jump to an other code
- Syntax has 2 deferent way
- if (statement is TRUE) Execute this line of code
- if (TRUE) {

  /* between the braces is the body of the if statement */

  Execute all statements inside the body

}

CONTROL FLOW – IF ELSE CONDITION

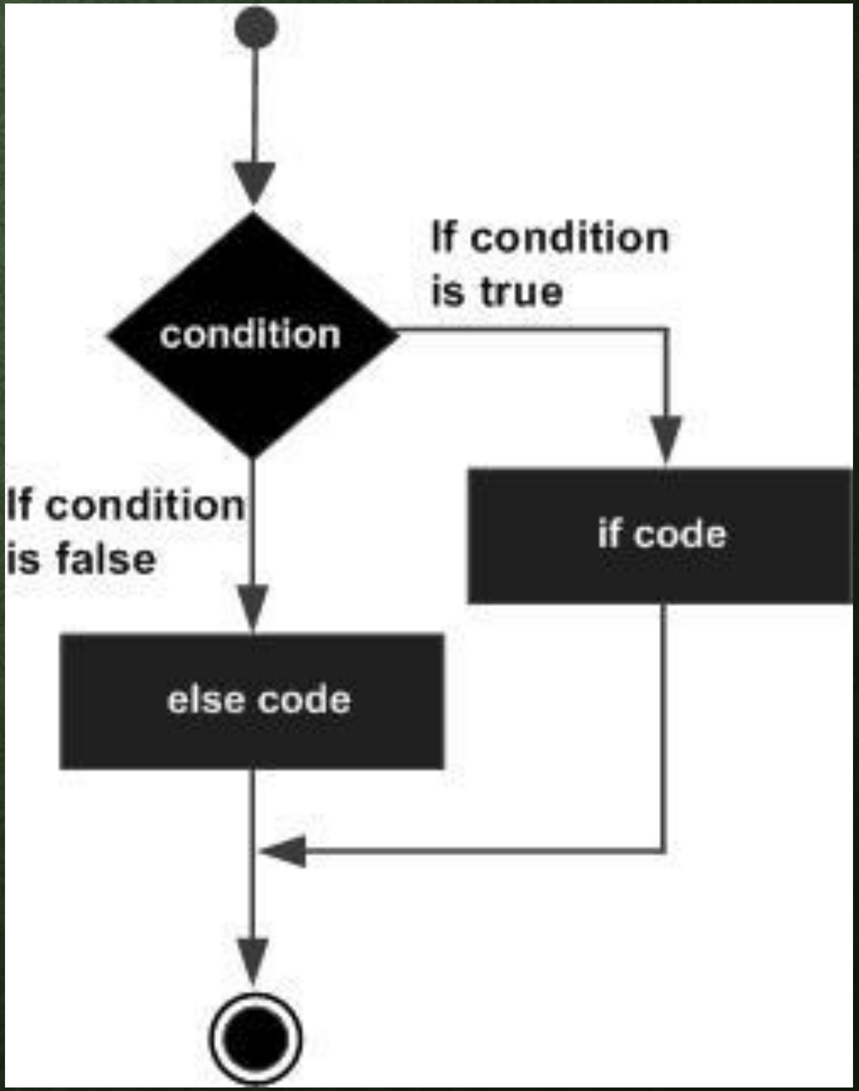
- when the condition in an if statement evaluates to false, it would be nice to execute some code instead of the code executed when the statement evaluates to true

```
if(Boolean expression) {

   /* statement will execute if the Boolean expression is true */

}

else {

   /* statement will execute if the Boolean expression is false */

}
```

CONTROL FLOW – NESTED IF

- If we want make some rule or condition for some operation so we want other solution to make data flow or make multi choices as If-else if like this Example :

```
 int a = 100;
   if( a == 10 ) {   printf("Value of a is 10\n" );
   } else if( a == 20 ) {  printf("Value of a is 20\n" );
   } else if( a == 30 ) {   printf("Value of a is 30\n" );
   } else {   printf("None of the values is matching\n" );
   }
   printf("Exact value of a is: %d\n", a );
```



CONTROL FLOW – SWITCH

- 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. Like Example:

```
char grade = 'B';
 switch(grade) {   case 'A' :  printf("Excellent!\n" );    break;
    case 'B' :  case 'C' :
       printf("Well done\n" );     break;
    case 'D' :
       printf("You passed\n" );
       break;
default :
       printf("Invalid grade\n" );
 }
```

CONTROL FLOW – WHILE LOOP

- when a block of code needs to be executed several number of times.
- while loop in C repeatedly executes a target statement as long as a given condition is true.
- The syntax of while is:

```
while(condition) {
  statement
}
```

- Example:

```
int a = 10;
while( a < 20 ) {
    printf("value of a: %d\n", a);
    a++;
  }
```

CONTROL FLOW – DO-WHILE

- if the fact that it is guaranteed to execute at least one time then condition is true can re-execute
- The syntax of do…while()

```
do {  statement   } while( condition );
```

- Example:

```
int a = 1;
do {
     printf("value of a: %d\n", a);
     a = a + 1;
  }while( a <= 10 );
```

CONTROL FLOW – FOR LOOP

- for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
- Example of for loop:

```
int a;
for( a = 10; a < 20; a ++ ){
printf("value of a: %d\n", a);
}
```

CONTROL FLOW – NESTED FOR LOOP

- Nested allow make loop inside loop with a deferent statement loop type.
- You can show For loop nested by an other for loop or while nested by do…while or can make For loop inside while loop inside tow level of For.
- Example:

```
int i, j;
 for(i = 1; i<6; i++) {
    for(j = 1; j <= i; j++){
       printf("*");
}
     printf("\n");
  }
```

FUNCTIONS

- Simple function
- Function prototype
- Overload function
- Passing parameter
- Inline function

FUNCTION –SIMPLE FUNCTION

- Functions like main or printf and scanf function it has many types to definition or declaration
- Any simple function contain return type, function name, parameter and function body.
- function may return a value like integer or any data type, but some function does return value which declare it by void
- This is the actual name of the function. If you want call it you can use name to called as printf.
- 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 function body contains a collection of statements

FUNCTION – PROTOTYPE

- Will make example to show how to declare function as prototype.
- Example 1:
- int max(int num1, int num2);
- void print_add(int sum1,int sum2);
- That prototype should declare before main function or in external header file to implement it in your file.

FUNCTION – OVERLOADING

- Some time you want to make function make the same functionality with same name , but with other values or data type how do you make that?
- Using overloading function you can show other information in this link:
- https://stackoverflow.com/questions/479207/function-overloading-in-c
- Because it's very complicated to explain for beginner .

FUNCTION – PASSING PARAMETER

- This example explain how to pass parameter by value:

```
void swap(int x, int y){
    int temp;
    temp = x;
    x = y;
    y = temp;                }
int main () {
   int a = 100, int b = 200;
   printf("Before swap, value of a : %d\n", a );
  printf("Before swap, value of b : %d\n", b );
  swap(a, b);
  printf("After swap, value of a : %d\n", a );
 printf("After swap, value of b : %d\n", b );
  return 0;
}
```

FUNCTION –INLINE FUNCTIO

- There are various ways to define inline functions; any given kind of definition might definitely emit stand-alone object code, definitely not emit stand-alone object code, or only emit stand-alone object code if it is known to be needed. Sometimes this can lead to duplication of object code, which is a potential problem for following reasons:
- It wastes space.
- It can cause pointers to what is apparently the same function to compare not equal to one another.
- It might reduce the effectiveness of the instruction cache.
- Example:

```
inline int max(int a, int b) {
 return a > b ? a : b;
}
```

FILE I/O

- File io standard library
- What is CRWD?
- Create Text file
- Read Text file
- Write Text file
- Delete file
- Size of file

FILES STANDARD LIBRARY

- You can make file and write and read and delete using only standard library stdio.h
- the standard input and output devices handled by C programming language. This chapter cover how C programmers can create, open, close text or binary files for their data storage.

WHAT ID CRWD?

- C is create file to append or read from it .
- R is read file can you read file and know size of an file path
- W is write in files text or binary to store information
- D is accessible to delete your file or any file just specified path
- C programming allow to use high level function to make action on low level functionality . Thanks C for this future.

CREATE TEXT FILE

- Example

```
FILE *fp = fopen("textFile.txt" ,"a");
```

- You can use the fopen( ) function to create a new file or to open an existing file. This call will initialize an object of the type FILE
- filename is a string literal, which you will use to name your file, and access mode

MODE OF FILES

- r
- Opens an existing text file for reading purpose.
- w
- Opens a text file for writing. If it does not exist, then a new file is created. Here your program will start writing content from the beginning of the file.
- a
- Opens a text file for writing in appending mode. If it does not exist, then a new file is created. Here your program will start appending content in the existing file content.
- r+
- Opens a text file for both reading and writing.
- w+
- Opens a text file for both reading and writing. It first truncates the file to zero length if it exists, otherwise creates a file if it does not exist.
- a+
- Opens a text file for both reading and writing. It creates the file if it does not exist. The reading will start from the beginning but writing can only be appended.

READ FILE

- Using fscanf function.
- Example:

```
FILE *fp;
   char buff[255];
fp = fopen("/tmp/test.txt", "r");
   fscanf(fp, "%s", buff);
   printf("1 : %s\n", buff );
 fclose(fp);
```

- We used array of char to store the file stream to make operation like parse or print etc…

WRITE FILE

- Using fprinf function to write or append to file.
- Example

```
FILE *fp;


   fp = fopen("/tmp/test.txt", "w+");
   fprintf(fp, "This is testing for fprintf...\n");
 fclose(fp);
```

- You should be close file after write to don’t have any exception and to free memory.

DELETE FILES

- The C library function int remove(const char *filename) deletes the given filename so that it is no longer accessible.
- Example:

```
int result;
  FILE *fp;
  char filename[] = "file.txt";
  fp = fopen(filename, "w");
  fprintf(fp, "%s", "This is tutorialspoint.com");
  fclose(fp);
  result = remove(filename);
  if(result == 0) {
    printf("File deleted successfully");
  }else {
    printf("Error: unable to delete the file");
  }
```

SIZE OF FILE

- Using group of functions as (fseek(),ftell())
- fseek() take 3 parameter return nothing and ftell() take pointer of FILE object return decimal number

```
FILE *fp;

char filename[80];

long length;

printf("input file name:");

gets(filename);  // or scanf("%s",filename)

fp=fopen(filename,"r");

if(fp==NULL) {

printf("file not found!\n");

}else {fseek(fp,OL,SEEK_END);

length=ftell(fp);

printf("the file's length is %1dB\n",length);

fclose(fp);

}

return 0;
```

OTHER FAMOUS WORKS

- ICDL concept (Arabic book)
- Facts of hackers Handbook(Arabic book)
- Object-oriented programming in C++ Cookbook(Arabic book)
- Groovy Tutorial in simple English TutorialBook(English book)
- computer programming at general concept in simple English (English book)

THANKS FOR INDEXES

- Stack over flow
- Tutorial Point
- C programming 2 edition book