Functions

- A function is a set of statements that take inputs, do some specific computation and produces output.
- The idea is to put some commonly or repeatedly done task together and make a function, so that instead of writing the same code again and again for different inputs, we can call the function.
- Function declaration tells compiler about number of parameters function takes, data-types of parameters and return type of function
- Putting parameter names in function declaration is optional in function declaration, but it is necessary to put them in definition

```
// A function that takes a charas parameters
// and returns an reference variable
char &call(char b);
```

What is the purpose of a function prototype?

The Function prototype serves the following purposes –

1) It tells the return type of the data that the function will return.
2) It tells the number of arguments passed to the function.
3) It tells the data types of the each of the passed arguments.
4) Also it tells the order in which the arguments are passed to the function.

Therefore essentially, function prototype specifies the input/output interlace to the function i.e. what to give to the function and what to expect from the function.

Prototype of a function is also called signature of the function.

Function prototype tells compiler about number of parameters function takes, data-types of parameters and return type of function. By using this information, compiler cross checks function parameters and their data-type with function definition and function call

What happens when a function is called before its declaration in C?

In C, if a function is called before its declaration, the **compiler assumes return type of the function as int**.

- In C, functions can return any type except arrays and functions. We can get around this limitation by returning pointer to array or pointer to function
- If in a C program, a function is called before its declaration then the C compiler automatically assumes the declaration of that function in the following way:
  int function name();
  And in that case if the return type of that function is different than INT ,compiler would show an error.

Static Functions

- 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

How are variables scoped in C – Static or Dynamic?

In C, variables are always statically (or lexically) scoped

Nested functions in C

- Lexical scoping is not valid in C because the compiler cant reach/find the correct memory location of the inner function.
- Nested function is not supported by C because we cannot define a function within another function in C
- We can declare a function inside a function, but it's not a nested function.

Inline Function

- To answer this first we need to talk about stack memory
- When every time a function gets called, it creates a stack and that created memory will be cleared once the function completed
- Now for inline function it won't Create new stack instead it uses the same stack where it's been called from
- E.g.: function n is an inline function gets invoked from function a
- Remember when function a gets called a stack has been created
- But when function n gets called it won't create new stack and used function a stack only
- In kernel its widely used. This is typically done as a way for improving performance, more optimization, although not necessarily the case.
- Its mainly used for small time critical functions and we can see great performance difference by doing so.
- Inline functions are nothing but the normal functions witha keyword inline added to it..inline functions are used because it executes muvh faster than a normal function bt requires more memory..this is because an inline function places its statement block in the call statement..so the program executes faster when the function is called..so an inline function could return values unless it is declared as void..
- Making a function an inline function suggests that calls to the function be as fast as possible.Sure, the run-time performance would increase,thus eliminating the need of additional overhead.

  But, if the function is large, the executable would become large. Be careful to use this keyword. It is suggested to not tag the functions as inline which have more that 3 lines of code.

- The intent of the **inline** specifier is to serve as a hint for the compiler to perform optimizations, such as function inlining, which require the definition of a function to be visible at the call site. The compilers can (and usually do) ignore presence or absence of the inline specifier for the purpose of optimization.
- Any function with internal linkage may be declared static inline with no other restrictions.

Function Pointer

It is similar to the other pointers but the only difference is that it points to a function instead of the variable.
In the other word, we can say, a function pointer is a type of pointer that store the address of a function and these pointed function can be invoked by function pointer in a program whenever required.

Declaration of function pointers in c

The syntax for declaring function pointers are very straightforward. It seems like difficult in beginning but once you are familiar with function pointer then it becomes easy.

The declaration of function pointers is similar to the declaration of a function. That means function pointer also requires a return type, declaration name, and argument list. One thing that you need to remember here is, whenever you declare the function pointer in the program then declaration name is preceded by the * symbol and enclosed in parenthesis.

For example, **void (*fpData)(int)**

For the better understanding, let’s take an example
e.g,

void (*pfDisplayMessage) (const char *);

In above expression, pfDisplayMessage is a pointer to a function taking one argument, const char *, and returns void.

When we declare function pointers in c then there is a lot of importance of the bracket. If in the above example, I remove the bracket, then the meaning of the above expression will be changed and it becomes void *pfDisplayMessage (const char *). It is a declaration of a function which takes the const character pointer as arguments and returns void pointer.

List of some function pointers

A function pointer must have the same signature to the function that it is pointing to. In a simple word, we can say that function pointer and its pointed function should be same in parameters list and return type.

So there can be a lot of possibility of function pointer in c. In below section, I am listing some function pointers and I want you to write the name of these function pointers in a comment box.

Initialization of function pointer in c

We have already discussed that a function pointer is similar to normal pointers. So after the declaration of a function pointer, we need to define it like normal pointers.

A function pointer is initialized to the address of a function but the signature of function pointer should be same as the function.

For example, here I am creating a function pointer taking two arguments, integers and returns an integer. This pointer is pointing to a function AddTwoNumber is used in addition of two number.

Let’s take a look at the below declaration.

declaration of function pointer

int (* pfAddTwoNumber) (int, int);

Initialize the function pointer with the address of AddTwoNumber

pfAddTwoNumber = & AddTwoNumber;

or

because the function name also represents the beginning address of the function, so we can also use the function name to initialize the function pointer.

pfAddTwoNumber = AddTwoNumber;

We can also initialize the function pointer in a single line.

int (* pfAddTwoNumber) (int, int) = AddTwoNumber;

Some Important points for the function pointer

Memory allocation and deallocation for function pointer

We cannot allocate or deallocate memory for the function pointer as like the normal pointers. The aim of the creation of function pointer is to point the beginning address of the function. So if you allocate the memory for the function pointer then there is no importance to create the function pointer.

void (*pfData) (int) = malloc(sizeof**(pfData));** // Not useful expression

Assigning function address to a function pointer

The function name is also used to get the address of the function, that means in a program we can also use the function name to get the address of the function there is no need to write & operators with the function name.

pfAddTwoNumber = AddTwoNumber;
or
pfAddTwoNumber = &AddTwoNumber;

Calling a function using the function pointer

After the initialization of function pointer, we can call the function using the initialized function pointer. For your understanding, I am breaking the function calling through a pointer in a few steps.

- First, write the name of a function pointer.
  pfAddTwoNumber;
- We know that function pointer is also similar to another pointer but the only difference is that function pointer pointed to a function except for the variable. So we put the * as a prefix to the name of function pointer to take the value of the function pointer.
  ***pfAddTwoNumber;**
- Finally, we have a function. For the function calling just simply add the arguments list with extra parenthesis to the above expression.
  (*pfAddTwoNumber)(10, 20);

Function pointer as arguments

In the c language, we can pass the function pointer as an argument into the function like the other pointers. Let's take an example to understand how to pass a function pointer in a function. I am writing an example code.

In below code, ArithMaticOperation is a function that takes three arguments two integers, and one function pointer. The function pointer is pointed to a function that takes two integers as arguments and returns an integer.

In the example code, there are three functions that perform the addition, subtraction, and multiplication operation. We can pass these function in ArithMaticOperation as the argument to perform the operation on integer as per the user choice.

Use of typedef with the function pointer

Using typedef, we can make the declaration of an array of function pointers easy.
For example,
In above example, I have used a complex expression for the declaration of the array of function pointers. Using typedef, we can make it easier.

// typedef of array of function pointers
typedef int (*apfArithmatics[3])(int,int);

Now, apfArithmatics is a new data type and it is able to store the address of three functions.

apfArithmatics aArithmaticOperation = **{AddTwoNumber,SubTwoNumber,MulTwoNumber};**

Declare function pointers in the structure

C is not an object-oriented language, so it does not contain the member functions like C++. In the c language, a structure does not contain the functions, only contains the data. We can store a function within the structure using the function pointers and these function pointers are treated like a member function in C++.

Applications

Call-back function:

In C language, we can implement the call-back using the function pointer. A call-back function is an important feature of any programing language, it keeps the reference to executable code that is passed as an argument to the other function and where it is used to call the referenced executable code.

Generally, the call-back function is used in library API because the library API is not only used by you but it is also used by the third person. If you use hard-coded function name in the library API then it will create issues.

The advantage of the call-back is that in library API caller does not need to know about the callee function, it only knows the prototypes of the callee function.

To replace a nested switch using the array of a function pointer

One of the uses of function pointer is to replace the nested switch. Let's understand it with below the example is shown below., In this code, I have a nested switch case, and I will remove the nested switch case using the array of the function pointer.

In this code, every state has three sub-states and each sub-states has a function which performs the mathematical operation. If the user selects the arithmetic state (state == Arithmetic), the user can select any of the three sub-states, each sub-state has a function which performs a mathematical operation. If user select first sub-state (substate1) than addition will be performed.

To display the result of the mathematical operation user needs to select the message state (state == Message) and after that corresponding sub-state.

Function Pointer in C Struct