Pointer to functions

“A pointer to a function points to the address of the executable code of the function. You can use pointers to call functions and to pass functions as arguments to other functions. You cannot perform pointer arithmetic on pointers to functions.”

The type of a pointer to a function is based on both the return type and parameter types of the function.

A declaration of a pointer to a function must have the pointer name in parentheses.

The syntax for declaring a function pointer might seem messy at first, but in most cases it's really quite straight-forward once you understand what's going on. Let's look at a simple **example:**

**void** (\*ptr)(**int**);

**ptr** is a pointer to a function taking one argument, an integer, and that returns void. It's as if you're declaring a function called "\***ptr**", which takes an int and returns void; now, if \***ptr** is a function, then **ptr** must be a pointer to a function.

To call the function pointed to by a function pointer, you treat the function pointer as though it were the name of the function you wish to call.

**void** **hello**()

{

printf("Hello**\n**");

}

**int** **main**()

{

**void** (\*ptr)();

ptr = hello; //ptr references function hello

**return** **0**;

}

* Like normal pointers, we can have an array of function pointers.

**Function pointer** can be used in place of switch case. For example, in below program, user is asked for a choice between 0 and 2 to do different tasks.

#include <stdio.h>

void add(int a, int b)

{

printf("Addition is %d\n", a+b);

}

void subtract(int a, int b)

{

printf("Subtraction is %d\n", a-b);

}

void multiply(int a, int b)

{

printf("Multiplication is %d\n", a\*b);

}

int main()

{

// fun\_ptr\_arr is an array of function pointers

void (\*fun\_ptr\_arr[])(int, int) = {add, subtract, multiply};

unsigned int ch, a = 15, b = 10;

printf("Enter Choice: 0 for add, 1 for subtract and 2 "

"for multiply\n");

scanf("%d", &ch);

if (ch > 2) return 0;

(\*fun\_ptr\_arr[ch])(a, b);

return 0;

}

**Benefits of Function Pointers**

* Function pointers provide a way of passing around instructions for how to do something
* You can write flexible functions and libraries that allow the programmer to choose behavior by passing function pointers as arguments
* This flexibility can also be achieved by using classes with virtual functions