Chapter - 9 : Pointers

Pointers and Functions
- When the program is executed, the code for each function is loaded into memory starting at a specific address.
- A pointer to a function holds the starting address of a function - its entry point.
- The code for each function is loaded into memory starting at a specific address.
- Like other pointers, one must declare a pointer to a function.
- The general form of the declaration is as follows:
  
  \[
  \text{type (*ptr_to_func)(parameter_list);} 
  \]

Function Pointer
A pointer which keeps address of a function is known as function pointer.

Steps to use pointer to call function:
- Declare Pointer which is capable of storing address of function.
- Initialize Pointer Variable.
- Call function using Pointer Variable.

This table show the calling \texttt{display( )} function with the pointer to function.

<table>
<thead>
<tr>
<th>Return Type :</th>
<th>Parameter :</th>
<th>void (*ptr)( );</th>
<th>ptr=&amp;display;</th>
<th>(*ptr)( );</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>None</td>
<td>int (*ptr)( );</td>
<td>ptr=&amp;display;</td>
<td>int result;</td>
</tr>
<tr>
<td>Float</td>
<td>None</td>
<td>float (*ptr)( );</td>
<td>ptr=&amp;display;</td>
<td>float result;</td>
</tr>
<tr>
<td>Char</td>
<td>None</td>
<td>char (*ptr)( );</td>
<td>ptr=&amp;display;</td>
<td>char result;</td>
</tr>
</tbody>
</table>

Example

```c
#include <stdio.h>
void display(int a, int b)
{
    printf("Value of sum \%d", a+b);
}

void main()
{
    void (*f_ptr)(int, int);
    //Assign the address of function to pointer
    f_ptr=&display;
    //Call the function through pointer
    f_ptr(10,20);
}
```
Array of Function Pointer
- Array of function pointer contains the pointer to multiple functions.
- It can be declared in a similar ways a normal pointer to function
- Syntax
  \[
  \text{type} \ ( \ast \text{ptr	extunderscore to	extunderscore func}[ \text{array	extunderscore size}]) (\text{parameter	extunderscore list}) ;
  \]

Example

```c
// This program define a function pointer and call the different function.
#include <stdio.h>
void add(int a, int b)
{    printf("Value of sum \%d \n", a+b);
}

void multiply(int a, int b)
{    printf("Value of multiply \%d\n", a*b);
}

void subtract(int a, int b)
{    printf("Value of subtract \%d\n", a-b);
}

void main()
{    void (*f_ptr[3])(int, int);
    //Assign the address of function to pointer
    f_ptr[0]=&add;
    f_ptr[1]=&multiply
    f_ptr[2]=&subtract;

    //Call the function through pointer
    f_ptr[0](10,20);   //call the add ( )
    f_ptr[1](10,20);   //call the subtract ( )
    f_ptr[2](10,20);   //call the multiply ( )
}
```

Output

Value of sum 30
Value of multiply 300
Value of subtract -20
Pointers to Structures

- A structure is a collection of variables within one variable.
- Structures can be termed as a variable that aggregates variables of different or similar types.
- The correlation of structures and pointers is very strong.
- A structure provides a very intuitive way to model user defined entities (records, packet formats, image headers, etc.).

```c
struct name
{
    Member 1;
    Member 2;
    ....
};

struct name *ptr;
```

Accessing Structure Member through Pointer

- In normal structure, to access the member of structure, we can use the dot(.) operator.
- In case of pointer to structure, 2 ways are specified
  1. (*struct_ptr).member
  2. struct_ptr -> member

Example

```c
#include <stdio.h>

struct Book
{
    char name[20];
    int price;
};

void main()
{
    struct Book a="C Prog", 200);
    struct Book *ptr;

    ptr=&a;

    printf("Book Name =%s\n", ptr->name);
    printf("Book Price =%d\n", ptr->price);

    printf("Book Name =%s\n", (*ptr).name);
    printf("Book Price =%d\n", (*ptr).price);
}
```
Example-2

```
#include <stdio.h>

struct Book
{
    char name[20];
    int price;
}

void main()
{
    struct Book *ptr;
    ptr = &a;

    /* print the Book details – Method-1 */
    printf("Book Name =\\%s\\n", ptr[0].name);
    printf("Book Price =\\%d\\n", ptr[0].price);
    printf("Book Name =\\%s\\n", ptr[1].name);
    printf("Book Price =\\%d\\n", ptr[1].price);

    /* print the Book details - Method -2 */
    printf("Book Name =\\%s\\n", (ptr+0)->name);
    printf("Book Price =\\%d\\n", (ptr+0)->price);
    printf("Book Name =\\%s\\n", (ptr+1)->name);
    printf("Book Price =\\%d\\n", (ptr+1)->price);

    /* print the Book details - Method -3 */
    printf("Book Name =\\%s\\n", ( *(ptr+0) ).name);
    printf("Book Price =\\%d\\n", ( *(ptr+0) ).price);
    printf("Book Name =\\%s\\n", ( *(ptr+1) ).name);
    printf("Book Price =\\%d\\n", ( *(ptr+1) ).price);
}
```