Pointer

Pointer:-**A pointer is a variable which contains the memory address of another variable. We can have a pointer to any variable type.**

OR

A pointer, also called a link or a reference.

Use of pointer:-

Every time you declare a variable, the compiler puts it somewhere, which you can now refer to as an address. Once you know that address, you can use it.

Syntax:-

datatype *ptr_name;

Example:-

int *ptr, *qt, *b;

float *var, *length;

char *ch;

& operator:- ***& oprator is used to get the address of any type of variable(like int, struct, union etc).***

Initializing a pointer:-

Int *ptr;

Int * pointer=&variable;

Example:- ***Write a program to initialization of pointer variable.***

Solution:-

```
#include<stdio.h>
using namespace std;
int main()
{
    int a;
    printf("Enter one value ");
    scanf("%d",&a);
    int *ptr=&a;
    printf("Value lives at: %d\n",a);
    printf("Pointer lives at: %d", *ptr);
    return 0;
}
```

Output:-

```
Enter one value 4
Value lives at: 4
Pointer lives at: 4
```

Example 2:- ***Write a program to operation on pointers.***

Solution:-

```
#include<stdio.h>
using namespace std;
int main()
```

```
{
        int boys,girls,total;
        int *ptrboys;
        int *ptrgirls;
        int *ptrtotal;
        ptrboys=&boys;
        ptrgirls=&girls;
        ptrtotal=&total;
        printf("Enter no. of boys");
        scanf("%d",ptrboys);
        printf("Enter no. of girls ");
        scanf("%d",ptrgirls);
        printf("Number of students");
        printf("\nBoys: \t %d",boys);
        printf("\nGirls: \t %d",girls);
        total=boys+girls;
        *ptrtotal=*ptrboys+*ptrgirls;
printf("\nTotal no. of students: \t %d",total);
        return 0;
}
```

Output:-

Enter no. of boys 45

Enter no og girls 56

Number of students

Boys:45

Girls:56

Total no. of students: 101

Pointer and Arrays:- *The concept of array is very much bound to the one of pointer In fact, the identifier of an array is equivalent to the address of its first element, as a pointer is equivalent to the address of the first element that it points to, so in fact they are the same concept. Elements of an array are accessed through pointers internally. Let us now try to understand the bonding between them.*

Pointers and One-Dimensional Arrays:-

Consider the following two declarataions

```
int number[20];

int *ptr;

ptr=&numbers[0];
```

Example:- ***Write a program to display the contents of an array using array name and also a pointer variable.***

Solution:-

```
#include<stdio.h>

using namespace std;

int main()
```

```
{
int arr[20];
int p,num;
int *ptr;
printf("Enter no of elements in the array\n");
scanf("%d", &num);
printf("Enter array elements\n");
for(p=0;p<num;p++)
{
        scanf("%d",&arr[p]);
}
ptr=arr;
printf("Array elements are:\n");
for(p=0;p<num;p++)
{
        printf("\nValue using array name
arr[%d]=%d",p,arr[p]);
        printf("Value using pointer arr[%d] =%d",p,*(ptr+p));
}
        return 0;
}
```

Output:-

```
Enter no of elements in the array
5
Enter array elements
1
2
3
34
5
Array elements are:
Value using array name arr[0]=1 Value using pointer arr[0]=1
Value using array name arr[1]=2 Value using pointer arr[0]=2
Value using array name arr[2]=3 Value using pointer arr[0]=3
Value using array name arr[3]=34 Value using pointer arr[0]=34
Value using array name arr[4]=5 Value using pointer arr[0]=5
```

Pointer Assignment:-

Example:-Write a program for pointer assignment.

Solution:-

```
#include<stdio.h>
using namespace std;
int main()
{
    int i,*p1,*p2;
```

```
        printf("Enter one value ");
        scanf("%d",&i);
        p1=&i;
        p2= p1;          //pointer assignment
        printf("*p1=%d and *p2=%d",*p1,*p2);
        return 0;
}
```

Output:-

```
Enter one value 5
*p1=5 and *p2=5
```

Array of pointers:-

Example:- ***Write a program to use of array of pointers.***

Solution:-

```
#include<stdio.h>
using namespace std;
int main()
{ int *arr[4];
    int i,j,k,l;
    printf("Enter i value");
    scanf("%d",&i);
    printf("Enter j value");
    scanf("%d",&j);
```

```
        printf("Enter k value");
        scanf("%d",&k);
        printf("Enter l value");
        scanf("%d",&l);
    arr[0]=&i;
    arr[1]=&j;
    arr[2]=&k;
    arr[3]=&l;
    for(int m=0;m<4;m++)
    printf("\n%d",*arr[m]);
        return 0;
    }
```

Output:-

```
Enter I value 5
Enter j value 51
Enter l value 15
Enter k value 56


5
51
15
56
```

Pointers and Strings:-

Example:- ***Write a program to use concept of pointers and string.***

Solution:-

```
#include<stdio.h>
using namespace std;
    char strA[80]="A String to be used for demonstration purpose";
    char strB[80];
int main(void)
{
char *pA;
char *pB;
puts(strA);
pA=strA;
puts(pA);
pB=strB;
putchar('\n');
while(*pA!='\0')
{
    *pB++=*pA++;
}
*pB='\0';
```

```
    puts(strB);
        return 0;
    }
```

Output:-

```
A String to be used for demonstration purpose
A String to be used for demonstration purpose

A String to be used for demonstration purpose
```

Pointer as Function Arguments:-*In C language a function can be called by the calling program in two ways.*

1. *Call by Value*
2. *Call by Reference*

1.Call by Value:-

Example:-*Write a program to use of call by Value.*

Solution:-

```
    #include<stdio.h>
    using namespace std;
    int main()
    {
        int x,y;
        void fun(int,int);
        x=20;
```

```
        y=80;
        printf("\n Value of x and y before function call=%d, %d",x,y);
        fun(x,y);
        printf("\n Value of x and y after function call=%d, %d",x,y);
        return 0;
}
void fun(int p,int q)
{
        p=p+p;
        q=q+q;
}
```

Output:-

```
Value of x and y before function call=20, 80
Value of x and y after function call=20, 80
```

2.call by Reference:-

Example:-Write a program to use Call by reference.

Solution:-

```
#include<stdio.h>
using namespace std;
int main()
```

```
{
        int x=25,y=10;
        void fun(int *p,int *q);
        printf("\n Before function call a=%d, b=%d",x,y);
        fun(&x,&y);
        printf("\n After function call a=%d, b=%d",x,y);
        return 0;
}
void fun(int *p,int *q)
{
        *p=*p+10;
        *q=*q+10;
        printf("\nInside the function a=%d, b=%d",*p, *q);
}
```

Output:-

```
Before function call a=25, b=10
Inside the function a=35, b=20
After function call a=35, b=20
```

Pointers and Functions:-

Here we explore the possibility of the following.

- *Passing pointers as arguments to functions*
- *Returning a pointer from a function*

➢ *Pointer to a function*

Passing pointers as arguments to functions:- *As we know that when an array is passed to a function as an argument then only the address of the first element of the array is passed, but not the actual values of the array elements. The function uses this address for manipulating the array elements. Similarly, we can pass the address of a variable as an argument to a function in the normal fashion.*

Example 1:- ***write a program to use pointers as arguments to function.***

Solution:-

```
#include<stdio.h>
void swap(int *,int *);
int main()
{
        int a,b;
        printf("Enter two number for swap value");
        scanf("%d%d",&a,&b);
        printf("a=%d,b=%d\n",a,b);
        swap(&a,&b);
        printf("a=%d,b=%d",a,b);
        return 0;
}
void swap(int *x,int *y)
```

```
{
        int temp;
        temp=*x;
        *x=*y;*y=temp;
        return;
}
```

Output:-

```
Enter two number for swap value 5
6
a=5,b=6
a=6,b=5
```

Example 2:- ***write a program to use pointers as arguments to function.***

Solution:-

```
#include<stdio.h>
void change(int *);
int x=5,y=6;
int main(void)
{
        int *p=&x;
        printf("Pointer points to value %d\n",*p);
        change(p);
```

```
        printf("Pointer points to value %d\n",*p);
        return 1;
}
void change(int *p)
{
        p=&y;
        printf("Pointer points to value %d\n",*p);
        return;
}
```

Output:-

```
Pointer points to value 5
Pointer points to value 6
Pointer points to value 5
```

Returning a pointer from a Function:- *Under some circumstances, we require functions to return a pointer.*

Syntax:-

```
Data_type *function_name(arg);
```

Example:- ***write a program to use function returning a pointer.***

Solution:-

```
        #include<stdio.h>
        int *mul(int, int, int);
```

```
int main()
{
        int a,b,c,*s;
        printf("Enter three number");
        scanf("%d%d%d",&a,&b,&c);
        printf("A=%d\nB=%d\nc=%d",a,b,c);
        s=mul(a,b,c);
        printf("\nMultiplication is %d",*s);
        return 0;
}
int *mul(int a,int b,int c)
{
        int res;
        res=a*b*c;
        return(&res);
}
```

Output:-

```
Enter three numbers 3 4 6
A=3
B=4
C=6
Multiplication is 72
```

Pointer to a function:- *Function also gives its starting address. Thus, instead of using the name of a function to call it, we can use a pointer to invoke it.*

Example:- ***Write a program to use pointer to a function.***

Solution:-

```
#include<stdio.h>
#include<conio.h>
int (*fpointer)(void); //Define a pointer to a function
int fun1(void); // define few functions
int fun2(void);
int main()
{
    fpointer=fun1; //put the address of 'func1' in fpointer
    fpointer();
    fpointer=fun2;  ////repeat for 'func2
    fpointer();
    getch();
}
int fun1(void)
{
    puts("Hello function first");
    return 0;
```

```
}
int fun2(void)
{
        puts("Hello function second");
return 0;
}
```

Output:-

```
Hello function first
Hello function second
```

Created by:-Ajay Kumar Verma