UNIDAD CURRICULAR: ALGORITMICA Y PROGRAMACIÓN

UNIDAD XI. PUNTEROS

CONTENIDO:

UNIDAD XI

PUNTEROS

DEFINICIÓN, DECLARACIÓN, OPERADORES Y OPERACIONES.

Un puntero es una variable que contiene una dirección de memoria. Normalmente, esa dirección es la posición de otra variable de memoria.

Si una variable va a contener un puntero, entonces tiene que declararse como tal. Una declaración de un puntero consiste en un tipo base, un ***** y el nombre de la variable. La forma general es:

tipo *nombre;

Donde tipo es cualquier tipo válido y nombre es el nombre de la variable puntero. El tipo base del puntero define el tipo de variables a las que puede apuntar. Técnicamente, cualquier tipo de puntero puede apuntar a cualquier dirección de la memoria, sin embargo, toda la aritmética de punteros esta hecha en relación a sus tipos base, por lo que es importante declarar correctamente el puntero.

Ejemplo:

int *puntero;

float *x;

En este ejemplo ***puntero*** apunta a un int y ***x*** apunta a un float.

Existen dos operadores especiales de punteros: ***&*** y *******. El operador de dirección ***(&)*** devuelve la ***dirección*** de memoria de su operando. El operador de indirección ***(*)*** devuelve el ***contenido*** de la dirección apuntada por el operando.

Después de declarar un puntero, pero antes de asignarle un valor, éste contiene un valor desconocido. Por convenio, se debe asignar el valor nulo a un puntero que no esté apuntando a ningún sitio, aunque ésto tampoco es seguro. Para asignar el valor nulo se utiliza la constante ***NULL***. NULL es una constante, que está definida como cero en varios ficheros de cabecera, como "***cstdio***" o "***iostream***",

Asignación de punteros

Como en el caso de cualquier otra variable, un puntero puede utilizarse a la derecha de una declaración de asignación para asignar su valor a otro puntero.

Ejemplo:

```
#include <iostream>
using namespace std;
int x=15;
int *p1,*p2;  //p1 y p2 son punteros a int
int main()
{
p1=&x;  //se le asigna la dirección de la variable x al puntero p1
p2=p1; //se le asigna a p2 lo que contiene p1, es decir, se le asigna la dirección de la variable x.  Tanto p1 como p2 apuntan a x.
cout<< *p1; //muestra el contenido de la dirección que tiene p1, es decir mostraría el valor de x: 15.
cout<<"La dirección de x es: "<<p1; //muestra la dirección donde se encuentra x.
cout<<"La dirección de x es: "<<p2; //muestra la dirección donde se encuentra x.
return 0;
}
```

PUNTEROS Y FUNCIONES

Ver unidad VI donde se mencionó el uso de punteros para el pase de parámetros por referencia.

PUNTEROS Y ESTRUCTURAS

Ver unidad IX donde se mencionó el uso de punteros para el acceso a estructuras de registros.

EJERCICIOS RESUELTOS

a) Tomando en cuenta el siguiente programa, indique el mensaje que saldrá por pantalla:

```
#include <iostream>
using namespace std;
int x=15,y;
int *p1;
int main()
{
 p1=&x;
 y=*p1;
 cout<<"El valor de Y es: "<<y;
return 0;
}
```

Por pantalla saldrá el mensaje: *El valor de Y es: 15*

b) Tomando en cuenta el siguiente programa, indique el mensaje que saldrá por pantalla:

```
#include <iostream>
using namespace std;
int x=15,y;
int *p1,*p2;
int main()
{
p1=&x;
y=*p1;
p2=&y;
*p2=20;
cout<<"El valor de Y es: "<<y;
return 0;
}
```

Por pantalla saldrá el mensaje: *El valor de Y es: 20*

c) Tomando en cuenta el siguiente programa, indique el mensaje que saldrá por pantalla:

```
#include <iostream>
using namespace std;
int x=20,y=4,z;
int *p1,*p2;
int main()
{
p1=&x;
p2=&y;
z=*p1+*p2;
cout<<"El valor de Z es: "<<z;
```

```
return 0;
}
```
Por pantalla saldrá el mensaje*: El valor de Z es: 24*

d) Tomando en cuenta el siguiente programa, indique el mensaje que saldrá por pantalla:

```
#include <iostream>
using namespace std;
int x=20,y=4,z;
int *p1,*p2;
int main()
{
p1=&x;
p2=&y;
z=*p1+*p2;
cout<<"El valor de Z es: "<<z<<endl;
z=x+y;
cout<<"El valor de Z es: "<<z<<endl;
*p1=10;
z=x+y;
cout<<"El valor de Z es: "<<z<<endl;
return 0;
}
```

Por pantalla saldrá el mensaje:
El valor de Z es: 24
El valor de Z es: 24
El valor de Z es: 14

REFERENCIAS BIBLIOGRÁFICAS

Joyanes, L. (2008). Fundamentos de programación. Algoritmos , Estructuras de datos y objetos. Mc Graw Hill. Tercera edición.

Joyanes, L. y Zahonero, I. (2005). Programación en C. Metodología, algoritmos y Estructura de datos. Mc Graw Hill. Segunda Edición

Martí, N. y Ortega, Y. (2004). Estructuras de datos y Métodos Algorítmicos. Ejercicios Resueltos. Pearson Education.