FUNDAMENTOS DE PROGRAMACIÓN

Inicio Clase 11

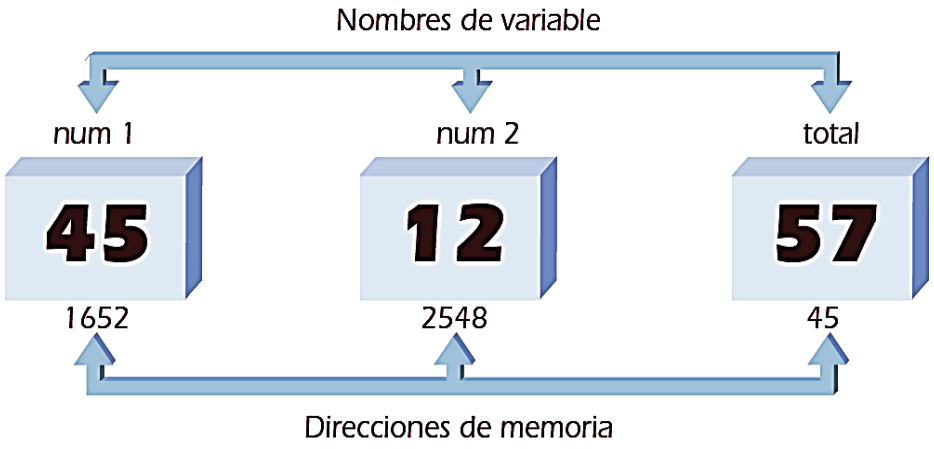
Profesor: Carlos Díaz

Clase 11: Punteros (Parte 1)

- Definición de puntero
- Diferentes usos de & en C++
- Operador de indirección *
- Punteros NULL y void
- Punteros y arrays

Definición de puntero

- Una variable puntero (o puntero, como se llama normalmente) es una variable que contiene la dirección de otra variable.
- Cuando una variable se declara, se asocian tres atributos a la misma: su nombre, su tipo y su dirección en memoria.
- Al valor o contenido de una variable se accede por medio de su nombre. A la dirección de la variable se accede por medio del operador de dirección &.
- Una referencia es un alias de otra variable. Se declara utilizando el operador de referencia (&) que se añade al tipo de la referencia.



Ejemplo 1

- Obtener el valor y la dirección de una Variable y una Referencia.
- Los dos identificadores Variable y Referencia son nombres diferentes para la misma variable, cuyo contenido y dirección son respectivamente 75 y 0x6ffe34.

```
Contenido de Variable = 75
Direccion &Variable = 0x6ffe34
Contenido de Referencia = 75
Direccion &Referencia = 0x6ffe34
```

```
#include <iostream>

using namespace std;

int main(){
        int Variable= 75; // Declaración de variable

        int& Referencia = Variable; //Referencia e inicialización

        cout << " Contenido de Variable = " << Variable << endl;

        cout << " Direccion &Variable = " << &Variable << endl;

        cout << " Contenido de Referencia = " << Referencia << endl;

        cout << " Direccion &Referencia = " << &Referencia << endl;

}
```

Diferentes usos de & en C++

- El carácter & tiene diferentes usos en C++:
- Cuando se utiliza como prefijo de un nombre de una variable, devuelve la dirección de esa variable.

```
cout << " Direccion &Variable = " << &Variable << endl;
```

- Cuando se utiliza como un sufijo de un tipo en una declaración de una variable, declara la variable como sinónimo de la variable que se ha inicializado.

```
int& Referencia = Variable; //Referencia e inicialización
```

- Cuando se utiliza como sufijo de un tipo en una declaración de parámetros de una función, declara el parámetro referencia de la variable que se pasa a la función.

```
int funcion(int &n, int &m);
```

Operador de indirección *

- Un puntero es una variable que contiene una dirección de una posición de memoria que puede corresponder o no a una variable declarada en el programa.
- La declaración de una variable puntero debe indicar el tipo de dato al que apunta; para ello se hace preceder a su nombre con un asterisco (*):

  <tipo de dato apuntado> *<identificador de puntero>

- C++ no inicializa los punteros cuando se declaran y es preciso inicializarlos antes de su uso.
- Después de la inicialización, se puede utilizar el puntero para referenciar los datos direccionados.
- Para asignar una dirección de memoria a un puntero se utiliza el operador &. Este método de inicialización, denominado estático, requiere:

  * Asignar memoria estáticamente definiendo una variable y, a continuación, hacer que el puntero apunte al valor de la variable.

  * Asignar un valor a la dirección de memoria.

Ejemplo 2

- Asignar a una variable puntero una dirección, y a su contenido un valor.

```
#include <iostream>
using namespace std;
int main(){
        int var; // define una variable entera var
        int *pun; //define un puntero a un entero pun
        pun = &var; //asigna la dirección de var a pun
        *pun = 60; // asigna al contenido de p 60
        cout << " &var. Direccion de var = " << &var << endl;
        cout << " pun. Contenido de pun es la misma direccion de var ";
        cout << pun << endl;
        cout <<" var. Contenido de var = " << var << endl;
        cout << " *pun. El contenido de *pun es el mismo que el de var: ";
        cout << *pun << endl;
}
```

```
&var. Direccion de var = 0x6ffe34
pun. Contenido de pun es la misma direccion de var 0x6ffe34
var. Contenido de var = 60
*pun. El contenido de *pun es el mismo que el de var: 60
```

Punteros NULL y void

- Un puntero nulo no apunta a ningún dato válido, se utiliza para proporcionar a un programa un medio de conocer cuando una variable puntero no direcciona a un dato válido.
- Los punteros void pueden apuntar a cualquier tipo de dato.
- Ejemplo:

```
int x, *px = &x, &rx = x;

char* c = "Cadena larga";

float *z = NULL;

void *r = px, *s = c, *t = z;
```

- x es una variable entera; px es un puntero a una variable entera inicializado a la dirección de x; rx es una referencia a un entero inicializada a x.
- c (puntero a carácter) es una cadena de caracteres de longitud 10.
- z es un puntero a un real inicializado a NULL.
- r es un puntero void incializado a un puntero a entero; s es un puntero void, inicializado a un puntero a char; t es un puntero void inicializado a un puntero a float.

Punteros y arrays

- Los arrays y los punteros están fuertemente relacionados en el lenguaje C++.
- El nombre de un array es un puntero que contiene la dirección en memoria de comienzo de la secuencia de elementos que forman el array.
- Este nombre del array es un puntero constante ya que no se puede modificar, sólo se puede acceder para indexar a los elementos del array.
- Para visualizar, almacenar o calcular un elemento de un array, se puede utilizar notación de subíndices o notación de punteros, ya que a un puntero p se le puede sumar un entero n, desplazándose el puntero tantos bytes como ocupe el tipo de dato.
- Si se tiene la siguiente declaración de array int V[6] = {1, 11, 21, 31, 41, 51};, su almacenamiento en memoria será el siguiente:

	V[0]	V[1]	V[2]	V[3]	V[4]	V[5]
memoria	1	11	21	31	41	51
	*V	*(V + 1)	*(V + 2)	*(V + 3)	*(V + 4)	*(V + 5)

Ejemplo 3

- Inicialización y visualización de un array con punteros.
- El programa inicializa un array de reales y visualiza las direcciones de cada una de las posiciones así como sus contenidos.

```
Direccion Contenido
V+0 = 0x6ffe20 V[0] = 11
V+1 = 0x6ffe24 V[1] = 21
V+2 = 0x6ffe28 V[2] = 31
V+3 = 0x6ffe2c V[3] = 41
V+4 = 0x6ffe30 V[4] = 51
V+5 = 0x6ffe34 V[5] = 61
```

```
#include <iostream>
using namespace std;
int main(){
        float V[6];
        for (int j = 0; j < 6; j++)
                *(V+j) = (j + 1) * 10 + 1;
        cout << " Direccion Contenido" << endl;
        for (int j= 0; j < 6; j++)
        {
                cout << " V+" << j << " = "<< V + j;
                cout << " V[" << j <<"] = " << *(V+j)<< "\n";
        }
}
```

Ejemplo 4

- Se puede declarar un array de punteros, como un array que contiene punteros como elementos, cada uno de los cuales apuntará a otro dato específico.
- El siguiente programa inicializa el array de reales V, así como el array de punteros a reales P, con las direcciones de las sucesivas posiciones del array V. Luego, visualiza las direcciones y los contenidos de V usando el array de punteros P.

```
#include <iostream>
using namespace std;
int main(){
        float V[6], *P[6];
        for (int j = 0; j < 6; j++)
        {
                *(V+j) = (5-j) * 10 + 1;
                *(P+j) = V+j; // inicialización de array de punteros
        }
        cout << " Direccion Contenido" << endl;
        for (int j = 0; j<6; j++)
        {
                cout << " V+" << j << " = " << *(P+j) << " = *(P+" << j << ")";
                cout << " V[" << j <<"] = " << **(P+j) << "\n";
        }
}
```

```
Direccion Contenido
V+0 = 0x6ffe20 = *(P+0) V[0] = 51
V+1 = 0x6ffe24 = *(P+1) V[1] = 41
V+2 = 0x6ffe28 = *(P+2) V[2] = 31
V+3 = 0x6ffe2c = *(P+3) V[3] = 21
V+4 = 0x6ffe30 = *(P+4) V[4] = 11
V+5 = 0x6ffe34 = *(P+5) V[5] = 1
```

FUNDAMENTOS DE PROGRAMACIÓN

Fin Clase 11

Profesor: Carlos Díaz