Programación 3

Facultad de Ingeniería y Ciencias Naturales

Instructor: Ernesto Enrique García Ramos

Contacto: egarcia97.r@gmail.com

gr15i04001@usonsonate.edu.sv

Guía 1: Conceptos básicos de POO

C++ está diseñado para la programación orientada a objetos (POO), y en este paradigma, todas las entidades que podemos manejar son objetos.

Objetivos:

- Recordar al estudiante conceptos básicos de programación 2
- Implementar dichos conceptos en aplicaciones prácticas.

Conceptos que debes conocer:

- Instancia: Es la creación de un objeto.
- Objeto: Son elementos que constan de un estado y de un comportamiento.
- Puntero: Es una dirección. Al contrario que una variable normal, un puntero es una variable almacenada en alguna parte del espacio del programa.
- Parse: Transformar el tipo de dato.

Antes de comenzar con la utilización de punteros, vamos a comenzar con un repaso acerca de los parámetros por valor y referencia, herramientas que nos ayudarán más adelante

Parámetros por valor y por referencia

Cuando se utilizan parámetros por valor, la información de la variable se almacena en una dirección de memoria diferente al recibirla en la función, es decir, el valor de la variable original no es modificado en la función, sólo en la copia.

Por el contrario, al utilizar parámetros por valor la variable que se recibe como parámetro en la función apunta exactamente a la misma dirección de memoria que la variable original por lo que si dentro de la función se modifica su valor también se modifica la variable original.

Ejemplos

- **Parámetros por valor**

```
#include <iostream>
using namespace std;

int funcion(int n, int m);

int main() {
   int a, b;
   a = 30;
   b = 9;

   cout << "Valor original de a: " << a << ", valor original de b: "
<< b << endl;
   cout<<"Sumando con los valores originales: "<<a+b<<endl;
```

```
    cout << "Utilizando funcion: "<< funcion(a, b) << endl;
    cout << "Verificando el valor de a: " << a << ", y el de b: " <<
b << endl;
    return 0;
}

int funcion(int n, int m) {
    n = 5;
    m =5;
    return n+m;
}
```

- **Parametros por referencia**

```
#include <iostream>
using namespace std;

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

int main() {
    int a, b;
    a = 30;
    b = 9;
    cout << "Valor inicial de a: " << a << ", de b: " << b << endl;
    cout << "Aplicamos la funcion...nos da como resultado: " <<
funcion(a, b) << endl;
    cout << "Nuevo valor de a: " << a << ", nuevo valor de b " << b
<< endl;
    return 0;
}

int funcion(int &n, int &m) {
    n =5;
    m =6;
    return m-n;
}
```

Punteros

Consideraciones:

1. Un nombre de variable, precedido por un signo & define la dirección de la variable.

2. Un puntero con un * precediéndole se refiere al valor de la variable señalada por el puntero.

Ejemplo basado en punteros y parámetros.

```
#include "iostream"
#include "stdlib.h"

using namespace std;

int funcion(int valor)
{
      valor = valor + 10; //Paso por valor
      return valor;
```

```
}

int funcionPunteros(int* valor)
{
	*valor = *valor + 10; //Se le suma 10 a la posición en memoria
	return *valor;
}

int main()
{
	int numero = 10;
	cout << "Antes de funcion " << numero << "\n"; //10
	funcion(numero); //Se pasa por valor
	cout << "Despues de funcion " << numero << "\n"; //10
	cout << "Antes de funcionPunteros " << numero << "\n"; //10
	funcionPunteros(&numero); //Se envía la dirección de memoria y
la función resuelve la referencia
	cout << "Despues de funcionPunteros " << numero << "\n"; //20
(10+10)

	system("pause");

	return 0;
}
```

Arreglos dinámicos:

```
#include "iostream"
#include "stdlib.h"
#include "string.h"

using namespace std;

int main()
{
	string* titulos = NULL;
	string* autores = NULL;
	int tamanio ;
	cout << "Cuantos libros y autores desea ingresar: ";
	cin>>tamanio;

	titulos = new string[tamanio];
	autores = new string[tamanio];

	cout << "Por favor ingrese la siguiente informacion de los
Libros: \n";
    for(int i = 0; i < tamanio; i++)
    {
        cout << "\n******* Libro " << i + 1 << "********:\n";
        cout << "Titulo: ";
      cin >> titulos[i];
      cout << "Autor: ";
      cin >> autores[i];
    }

      delete [] titulos;
      delete [] autores;
```

```
        titulos = NULL;
        autores = NULL;

        system("pause");

        return 0;
}
```

Ejercicio

Función que encuentre la posición de un carácter en la cadena, recibirá un puntero con la cadena y el carácter a buscar. Devolver la posición del carácter, repetir el proceso cuantas veces el usuario lo desee y convertir el vector en un arreglo dinámico.