


	UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 1

GUÍA DE LABORATORIO

(formato docente)

INFORMACIÓN BÁSICA					
ASIGNATURA:	TECNOLOGIA DE OBJETOS				
TÍTULO DE LA PRÁCTICA:	Puntero en C++				
NÚMERO DE PRÁCTICA:	02	AÑO LECTIVO:	2023-B	NRO. SEMESTRE:	
TIPO DE PRÁCTICA:	INDIVIDUAL				
	GRUPAL	X	MÁXIMO DE ESTUDIANTES	3	
FECHA INICIO:	18/09/2023	FECHA FIN:	25/09/2023	DURACIÓN:	50 min
RECURSOS A UTILIZAR: <i>Código C++, lab PC, presentaciones, explicación por casuística.</i>					
DOCENTE(s): <i>Mg. William Bornas Rios</i> <i>Mg. Karen Quispe Vergaray</i>					

OBJETIVOS/TEMAS Y COMPETENCIAS	
OBJETIVOS: <ul style="list-style-type: none"> Clases en C++ El alumno deberá de manipular diferentes tipos de estructuras utilizando punteros (pointers). 	
TEMAS: <ul style="list-style-type: none"> Manejo de punteros en C++ Administración de memoria en C++ Lista enlazada 	
COMPETENCIAS	<ul style="list-style-type: none"> Diseña responsablemente sistemas, componentes o procesos para satisfacer necesidades dentro de restricciones realistas: económicas, medio ambientales, sociales, políticas, éticas, de salud, de seguridad, manufacturación y sostenibilidad. Aplica de forma flexible técnicas, métodos, principios, normas, estándares y herramientas de ingeniería necesarias para la construcción de software e implementación de sistemas de información.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 2</p>

CONTENIDO DE LA GUÍA

I. MARCO CONCEPTUAL

Punteros

Es una forma para poder usar o consumir la memoria del computador.

La memoria de la computadora se divide en ubicaciones de memoria numeradas secuencialmente.

Cada variable está ubicada en una ubicación única en la memoria, conocida como su dirección.

STACK

Es la memoria que recibe las declaraciones de variables, es el local del computador, esta crece hacia direcciones mas bajas.

HEAP

Es la memoria dinámica que se accede con punteros (normalmente declarados con asterisco *) y utilizando el operador NEW. El crecimiento de esta memoria es hacia direcciones mas altas.

Por lo general, los programadores no necesitan saber la dirección particular de ninguna variable declarada, ya que el compilador lo maneja automáticamente. Sin embargo, podemos capturar la dirección del valor de la variable con el operador (&).

Cuando declaramos variables (int, char, etc) es el compilador que asigna la memoria para las variables declarando el tipo de variable; el compilador le asigna automáticamente una dirección. Por ejemplo, un entero largo suele tener cuatro bytes, lo que significa que la variable tiene una dirección para cuatro bytes de memoria.

En C++ podemos almacenar las direcciones de memoria en variables tipo punteros (ó apuntadores), los punteros pueden hacer referencia a direcciones de memoria desde simples variables declaradas hasta estructuras complejas.

```
unsigned short int Edad = 50;  
unsigned short int * pEdad = 0;  
pEdad = &Edad;
```

El operador (*) también es llamado operador de indirección.

II. EJERCICIO/PROBLEMA RESUELTO POR EL DOCENTE

- Reconocer las direcciones de memoria, operador &.

```
unsigned short shortVar = 5;  
unsigned long longVar = 65535;  
long sVar = -65535;  
  
cout << "shortVar:\t" << shortVar<<endl;  
cout << " Address of shortVar:\t"<<&shortVar << "\n";  
  
cout << "longVar:\t" << longVar <<endl;  
cout << " Address of longVar:\t" <<&longVar << "\n";  
  
cout << "sVar:\t" << sVar <<endl;  
cout << " Address of sVar:\t"<< &sVar << "\n";
```

- Analizar la asignación de valores y direcciones de memoria

```
typedef unsigned short int USHORT;  
USHORT myAge;  
USHORT * pAge = 0;  
myAge = 5;  
cout << "myAge: " << myAge << "\n";  
cout << "pAge: " << pAge<< "\n";  
  
pAge = &myAge;  
cout << "*pAge: " << *pAge << "\n";  
cout << "pAge: " << pAge << "\n\n";  
  
cout << "Asignar nuevo valor al puntero\n";
```

```
*pAge = 7;

cout << "*pAge: " << *pAge << "\n";
cout << "myAge: " << myAge << "\n";
cout << "pAge: " << pAge << "\n\n";

cout << "Asignar nuevo valor al puntero\n";

myAge = 9;

cout << "myAge: " << myAge << "\n";
cout << "*pAge: " << *pAge << "\n";
cout << "pAge: " << pAge << "\n";
```

- Analizar la asignación de valores y direcciones de memoria

```
unsigned short int myAge = 5, yourAge = 10;
unsigned short int * pAge = &myAge;

cout << "myAge:\t" << myAge << "\tyourAge:\t" << yourAge << "\n";
cout << "&myAge:\t" << &myAge << "\t&yourAge:\t" << &yourAge << "\n";

cout << "pAge:\t" << pAge << "\n";
cout << "*pAge:\t" << *pAge << "\n";
cout << "\n";

pAge = &yourAge;          // reasignar el pointer

cout << "myAge:\t" << myAge << "\tyourAge:\t" << yourAge << "\n";
cout << "&myAge:\t" << &myAge << "\t&yourAge:\t" << &yourAge << "\n";

cout << "pAge:\t" << pAge << "\n";
cout << "*pAge:\t" << *pAge << "\n";

cout << "&pAge:\t" << &pAge << "\n";
```

- Analizar la asignación de memoria Heap con punteros

```
int localVariable = 5;
int * pLocal = &localVariable;
int * pHeap = new int;
//int *pHeap = NULL;
if (pHeap == NULL)
{
    cout << "Error! No memory for pHeap!!";
    return 0;
}
*pHeap = 7;
cout << "localVariable: " << localVariable << "\n";
cout << "**pLocal: " << *pLocal << "\n";
cout << "**pHeap: " << *pHeap << "\n";
delete pHeap;
pHeap = new int;
if (pHeap == NULL)
{
    cout << "Error! No memory for pHeap!!";
    return 0;
}
*pHeap = 9;
cout << "**pHeap: " << *pHeap << "\n";
delete pHeap;
```

- Puntero a clase

Antes del main

```
class C {
public: int x;
       int* p;

       void fun() { cout << "Valor miembro x == " << x << endl; }
       C() { // constructor por defecto
           x = 13;
           p = &x;
       }
};

void f1(C* cpt); //prototipo de función
```

En el main

```
C c1;           // instancia de C
C* cptr;        // puntero a clase
cptr = &c1;     // asignado al objeto c1

cout << "1 c1.x == " << c1.x << endl; //
cout << "2 c1.p == " << *c1.p << endl; //

c1.fun();           //
f1(cptr);           // puntero se utiliza como argumento de f1
```

Función implementada:

```
void f1(C* cp) { // definición de función
    cout << "3 c1.x == " << (*cp).x << endl;
    cout << "4 c1.x == " << cp->x << endl;
    cout << "5 c1.p == " << *(*cp).p << endl;
    cout << "6 c1.p == " << *cp->p << endl;

    (*cp).fun();
    cp->fun();
}
```



Lista enlazada con punteros: Revisar algoritmo enviado por classroom.

III. EJERCICIOS/PROBLEMAS PROPUESTOS

1. Implementar una calculadora con 3 clases en el lenguaje c++, donde la primera analizará la operación matemática (suma, resta....), la segunda administrará las operaciones matemáticas (el núcleo de la calculadora), y la tercera procesará la operación ingresada.
El programa recibirá de entrada una cadena de texto con la operación a realizar ("10+37") ("45+14-42") ("1+2+3+4+5+6")
Como máximo el programa recibe 6 números a operar.
2. Implementar con punteros una lista doblemente enlazada, utilizar clases o struct.

IV. REFERENCIAS Y BIBLIOGRAFÍA RECOMENDADAS:

- [1] C++ reference, online: <https://en.cppreference.com/w/>
- [2] Laaksonen, A. (2017). Guide to Competitive Programming. Springer.
- [3] <https://paginas.matem.unam.mx/pderbf/images/mprogintc++.pdf>

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 7</p>

TÉCNICAS E INSTRUMENTOS DE EVALUACIÓN	
<p>TÉCNICAS: <i>Observación y retroalimentación in-situ.</i></p>	<p>INSTRUMENTOS: <i>Organización y resultado del trabajo en equipo.</i> <i>Desarrollo y sustentación de las actividades propuestas –grupal</i></p>
<p>CRITERIOS DE EVALUACIÓN</p> <ul style="list-style-type: none"> • Interiorizar y comprender el paradigma de la programación orientada a objetos (POO) y su utilidad. • Diseñar y producir código en C++ para resolver problemas. 	