
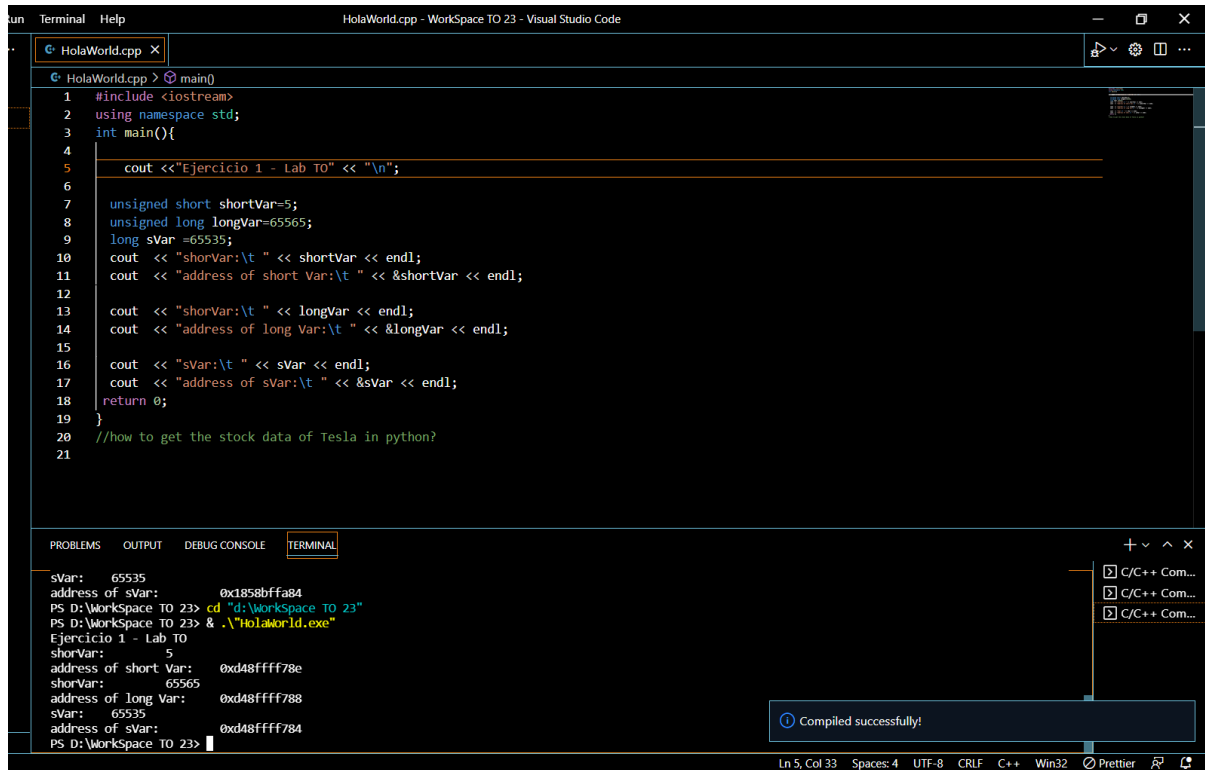
	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE LABORATORIO 01

INFORMACIÓN BÁSICA					
ASIGNATURA:	Tecnologia de Objetos				
TÍTULO DE LA PRÁCTICA:	Punteros				
NÚMERO DE PRÁCTICA:	2	AÑO LECTIVO:	2023A	NRO. SEMESTRE:	
FECHA DE PRESENTACIÓN	22/05/2023	HORA DE PRESENTACIÓN	08:30		
INTEGRANTE (s): FLORES RIVERA Patwil David Chara Quispe, Diego Mauricio Huaman Maqqe Rodrigo Sebastian Montoya Pinto Sneyder Gonzalo				NOTA:	
DOCENTE(s): Mg. William Milton Bornas Rios					

SOLUCIÓN Y RESULTADOS
<p>I. SOLUCIÓN DE EJERCICIOS RESUELTOS</p> <ul style="list-style-type: none"> • Reconocer las direcciones de memoria, operador &. <pre> 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"; </pre>



```
Terminal Help
HolaWorld.cpp - Workspace TO 23 - Visual Studio Code

HolaWorld.cpp > main()
1 #include <iostream>
2 using namespace std;
3 int main(){
4
5     cout << "Ejercicio 1 - Lab TO" << "\n";
6
7     unsigned short shortVar=5;
8     unsigned long longVar=65565;
9     long sVar =65535;
10    cout << "shortVar:\t " << shortVar << endl;
11    cout << "address of short Var:\t " << &shortVar << endl;
12
13    cout << "shortVar:\t " << longVar << endl;
14    cout << "address of long Var:\t " << &longVar << endl;
15
16    cout << "sVar:\t " << sVar << endl;
17    cout << "address of sVar:\t " << &sVar << endl;
18    return 0;
19 }
20 //how to get the stock data of Tesla in python?
21

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
sVar: 65535
address of sVar: 0x1858bffa84
PS D:\Workspace TO 23> cd "d:\Workspace TO 23"
PS D:\Workspace TO 23> & .\HolaWorld.exe
Ejercicio 1 - Lab TO
shortVar:
5
address of short Var: 0xd48ffff78e
shortVar: 65565
address of long Var: 0xd48ffff788
sVar: 65535
address of sVar: 0xd48ffff784
PS D:\Workspace TO 23>

Compiled successfully!
```

- 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";

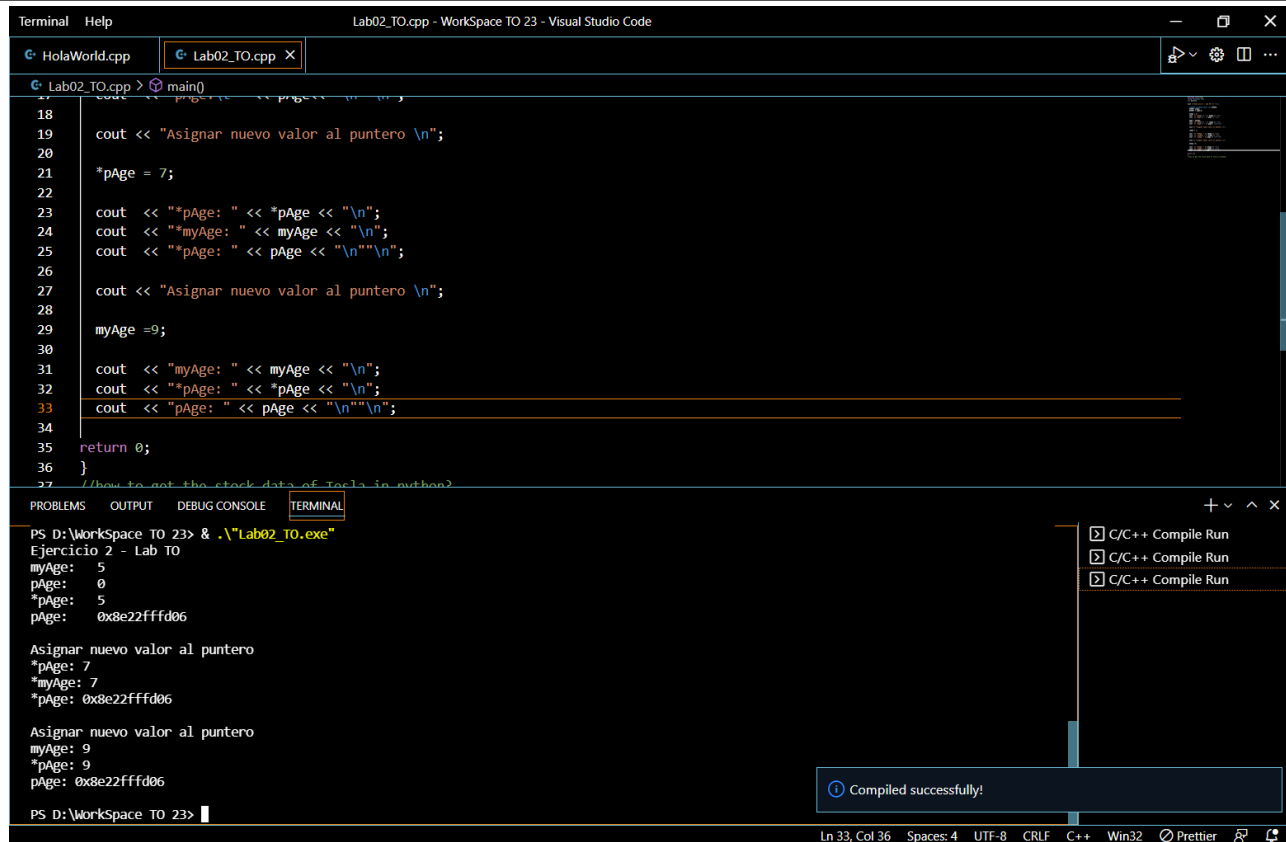
*pAge = 7;

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";
```



The screenshot displays the Visual Studio Code interface with a C++ file named `Lab02_TO.cpp` open. The code defines a `main` function that manipulates pointers and prints memory addresses. The `TERMINAL` pane at the bottom shows the output of the program, including the initial memory addresses of `myAge` and `pAge`, followed by the assignment of a new value to the pointer and the resulting memory addresses. A notification at the bottom right indicates that the code was compiled successfully.

```
18 //how to get the stock data of Tesla in python?
19 cout << "Asignar nuevo valor al puntero \n";
20
21 *pAge = 7;
22
23 cout << "myAge: " << *pAge << "\n";
24 cout << "myAge: " << myAge << "\n";
25 cout << "pAge: " << pAge << "\n\n";
26
27 cout << "Asignar nuevo valor al puntero \n";
28
29 myAge = 9;
30
31 cout << "myAge: " << myAge << "\n";
32 cout << "pAge: " << *pAge << "\n";
33 cout << "pAge: " << pAge << "\n\n";
34
35 return 0;
36 }
37
```

PS D:\Workspace TO 23> & .\Lab02_TO.exe

Ejercicio 2 - Lab TO

myAge: 5
pAge: 0
*pAge: 5
pAge: 0x8e22fffd06

Asignar nuevo valor al puntero

*pAge: 7
*myAge: 7
*pAge: 0x8e22fffd06

Asignar nuevo valor al puntero

myAge: 9
*pAge: 9
pAge: 0x8e22fffd06

PS D:\Workspace TO 23>

Compiled successfully!

- 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";

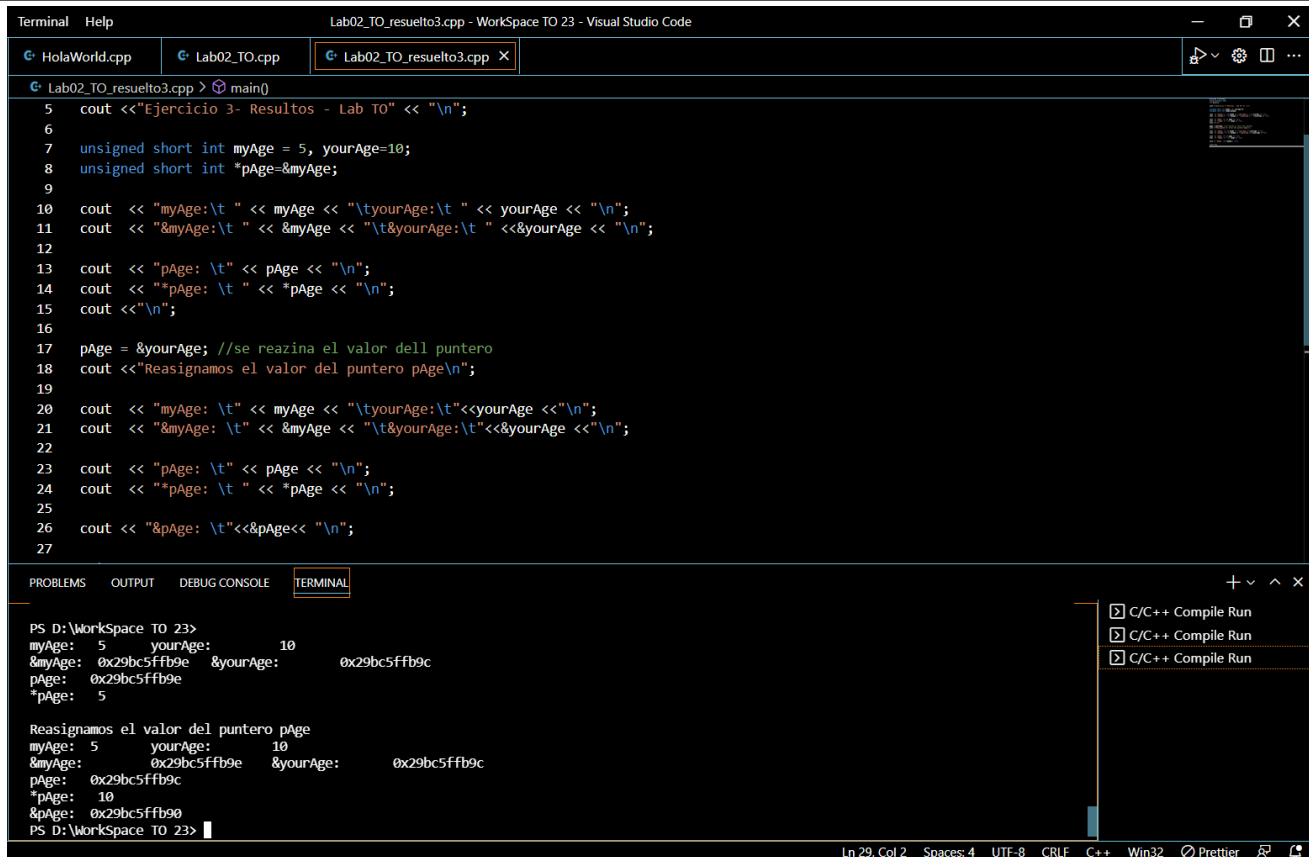
cout << "pAge:\t" << pAge << "\n";
cout << "**pAge:\t" << *pAge << "\n";
cout << "\n";

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";
```



The screenshot shows the Visual Studio Code interface with a C++ file named `Lab02_TO_resuelto3.cpp` open. The code is a C++ program that demonstrates pointer manipulation and memory addresses. It includes several `cout` statements to display the values of variables and pointers. The terminal output shows the execution results, including memory addresses for variables and pointers.

```
5  cout << "Ejercicio 3- Resultados - Lab T0" << "\n";
6
7  unsigned short int myAge = 5, yourAge=10;
8  unsigned short int *pAge=&myAge;
9
10 cout << "myAge:\t " << myAge << "\tyourAge:\t " << yourAge << "\n";
11 cout << "&myAge:\t " << &myAge << "\t&yourAge:\t " <<&yourAge << "\n";
12
13 cout << "pAge: \t" << pAge << "\n";
14 cout << "*pAge: \t " << *pAge << "\n";
15 cout << "\n";
16
17 pAge = &yourAge; //se reasigna el valor del puntero
18 cout << "Reasignamos el valor del puntero pAge\n";
19
20 cout << "myAge: \t" << myAge << "\tyourAge:\t"<<yourAge << "\n";
21 cout << "&myAge: \t" << &myAge << "\t&yourAge:\t"<<&yourAge << "\n";
22
23 cout << "pAge: \t" << pAge << "\n";
24 cout << "*pAge: \t " << *pAge << "\n";
25
26 cout << "&pAge: \t"<<&pAge<< "\n";
27
```

Terminal Output:

```
PS D:\Workspace TO 23>
myAge: 5      yourAge:      10
&myAge: 0x29bc5ffb9e  &yourAge:      0x29bc5ffb9c
pAge: 0x29bc5ffb9e
*pAge: 5

Reasignamos el valor del puntero pAge
myAge: 5      yourAge:      10
&myAge: 0x29bc5ffb9e  &yourAge:      0x29bc5ffb9c
pAge: 0x29bc5ffb9c
*pAge: 10
&pAge: 0x29bc5ffb90
PS D:\Workspace TO 23>
```

- 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

Clase_puntero.cpp X

```
Clase_puntero.cpp > C > C()
1  #include<iostream>
2  using namespace std;
3  class C {
4  public: int x;
5         int* p;
6
7         void fun() { cout << "Valor miembro x == " << x << endl; }
8         C() { // constructor por defecto
9             x = 13;
10            p = &x;
11        }
12    };
13    void f1(C* cpt); //prototipo de función
14    int main()
15    {
16        C c1; // instancia de C
17        C* cptr; // puntero a clase
18        cptr = &c1; // asignado al objeto c1
19
20        cout << "1 c1.x == " << c1.x << endl;
21        cout << "2 c1.p == " << *c1.p << endl;
22        c1.fun();
23        f1(cptr); // puntero se utiliza como argumento de f1
24        return 0;
25    }
26    void f1(C* cp) { // definición de función
27        cout << "3 c1.x == " << (*cp).x << endl;
28        cout << "4 c1.x == " << cp->x << endl;
29        cout << "5 c1.p == " << *(*cp).p << endl;
30        cout << "6 c1.p == " << *cp->p << endl;
31        (*cp).fun();
32        cp->fun();
33    }
```

PROBLEMAS SALIDA TERMINAL SQL CONSOLE

▼ TERMINAL

```
Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\rodri\OneDrive\Documentos\Teoria de Objetos\Lab02>ejer2.exe
1 c1.x == 13
2 c1.p == 13
Valor miembro x == 13
3 c1.x == 13
4 c1.x == 13
5 c1.p == 13
6 c1.p == 13
Valor miembro x == 13
Valor miembro x == 13
```


- Lista enlazada con punteros

```

Lista_Enlazada.cpp
Lista_Enlazada.cpp > ...
1  #include<iostream>
2  using namespace std;
3  struct Nodo {
4      int valor;
5      Nodo *siguiente;
6      ~Nodo() {
7          cout << "\nNodo eliminado " << valor;
8      }
9  };
10
11 void copiaArreglo(int arreglo[], int size, Nodo **inicio);
12 void imprimeLista(Nodo *inicio);
13 void destruyeLista(Nodo *inicio);
14
15 int main()
16 {
17     int arreglo[] = { 1,2,3,4,5,6,7,8,9,10 };
18     const int tama = sizeof(arreglo) / sizeof(*arreglo);
19     Nodo *inicio = nullptr;
20
21     copiaArreglo(arreglo, tama, &inicio);
22     imprimeLista(inicio);
23     destruyeLista(inicio);
24
25     int espera;
26     cin >> espera;
27
28     return 0;
29 }
30
31 void copiaArreglo(int arreglo[], int size, Nodo **inicio) {
32     Nodo *iNodo;
33     int i = 0;
34     *inicio = new Nodo;

```

PROBLEMAS SALIDA **TERMINAL** SQL CONSOLE

▼ **TERMINAL**

```

Microsoft Windows [Versión 10.0.19045.2965]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\rodri\OneDrive\Documentos\Teoria de Objetos\Lab02>g++ Lista_Enlazada.cpp -o ejer1.exe

C:\Users\rodri\OneDrive\Documentos\Teoria de Objetos\Lab02>ejer1.exe
1 2 3 4 5 6 7 8 9 10
Nodo eliminado 1
Nodo eliminado 2
Nodo eliminado 3
Nodo eliminado 4
Nodo eliminado 5
Nodo eliminado 6
Nodo eliminado 7
Nodo eliminado 8
Nodo eliminado 9
Nodo eliminado 10

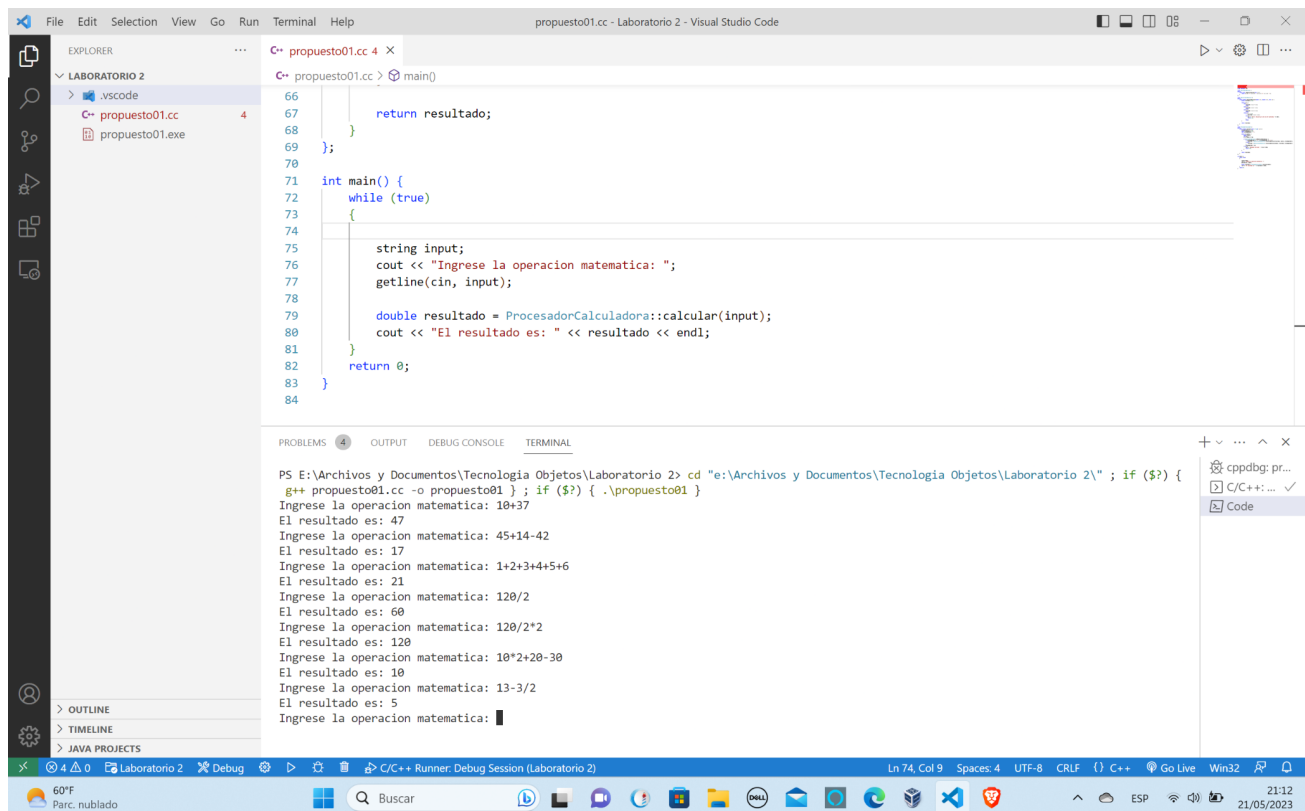
```

II. SOLUCIÓN EJERCICIOS 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.

En la función main, se solicita al usuario que ingrese la operación matemática como una cadena. Luego, se llama al método calcular de la clase ProcesadorCalculadora para obtener el resultado y se muestra en la salida.

Esta estructura de clases proporciona una separación de responsabilidades y facilita el mantenimiento y la extensibilidad del código. Cada clase se encarga de una parte específica del proceso de cálculo, lo que mejora la legibilidad y la organización del programa.



The screenshot shows the Visual Studio Code interface with a C++ file named `propuesto01.cc` open. The code implements a calculator with a `main` function that uses a `ProcesadorCalculadora` class. The terminal output shows the program running and calculating various expressions.

```

66         return resultado;
67     }
68 }
69 };
70
71 int main() {
72     while (true)
73     {
74
75         string input;
76         cout << "Ingrese la operacion matematica: ";
77         getline(cin, input);
78
79         double resultado = ProcesadorCalculadora::calcular(input);
80         cout << "El resultado es: " << resultado << endl;
81     }
82     return 0;
83 }
84

```

Terminal Output:

```

PS E:\Archivos y Documentos\Tecnologia Objetos\Laboratorio 2> cd "e:\Archivos y Documentos\Tecnologia Objetos\Laboratorio 2\"; if ($?) {
g++ propuesto01.cc -o propuesto01 } ; if ($?) { .\propuesto01 }
Ingrese la operacion matematica: 10+37
El resultado es: 47
Ingrese la operacion matematica: 45+14-42
El resultado es: 17
Ingrese la operacion matematica: 1+2+3+4+5+6
El resultado es: 21
Ingrese la operacion matematica: 120/2
El resultado es: 60
Ingrese la operacion matematica: 120/2*2
El resultado es: 120
Ingrese la operacion matematica: 10*2+20-30
El resultado es: 10
Ingrese la operacion matematica: 13-3/2
El resultado es: 5
Ingrese la operacion matematica:

```

```
41 class ProcesadorCalculadora {
42 public:
43     static double calcular(const string& input) {
44         stringstream ss(input);
45         double resultado = 0.0;
46         char ultimoOperador = '+';
47         while (!ss.eof()) {
48             double numero;
49             char op;
50             ss >> numero >> op;
51             if (AnalizadorOperacion::esOperacionValida(op)) {
52                 if (ultimoOperador == '*' || ultimoOperador == '/') {
53                     resultado = AdministradorOperacion::realizarOperacion(resultado, numero, ultimoOperador);
54                 } else {
55                     resultado = AdministradorOperacion::realizarOperacion(numero, resultado, ultimoOperador);
56                 }
57                 ultimoOperador = op;
58             } else {
59                 cout << "Operador inválido: " << op << endl;
60                 return 0.0;
61             }
62         }
63         return resultado;
64     }
65 };
```

Esta clase es responsable de procesar la cadena de entrada y realizar las operaciones matemáticas. Contiene un método estático llamado `calcular`, que recibe la cadena de operación matemática como parámetro. Utiliza un `stringstream` para leer los números y operadores de la cadena de entrada. Luego, realiza las operaciones matemáticas siguiendo las reglas de precedencia adecuadas. Retorna el resultado final de la operación.

2. Implementar con punteros una lista doblemente enlazada, utilizar clases o struct.

```
Lab2_propuesto2.cpp x
Lab2_propuesto2.cpp > DoubleLinkedList > mostrar()
1  #include <iostream>
2
3  struct Node {
4      int data;
5      Node* prev;
6      Node* next;
7  };
8
9  class DoubleLinkedList {
10 private:
11     Node* head;
12     Node* tail;
13
14 public:
15     DoubleLinkedList() : head(nullptr), tail(nullptr) {}
16
17 void insertFrente(int value) {
18     Node* newNode = new Node;
19     newNode->data = value;
20     newNode->prev = nullptr;
21     newNode->next = head;
22
23     if (head != nullptr) {
24         head->prev = newNode;
25     } else {
26         tail = newNode;
27     }
}
```

```
28
29     head = newNode;
30 }
31
32 void insertAtras(int value) {
33     Node* newNode = new Node;
34     newNode->data = value;
35     newNode->prev = tail;
36     newNode->next = nullptr;
37
38     if (tail != nullptr) {
39         tail->next = newNode;
40     } else {
41         head = newNode;
42     }
43
44     tail = newNode;
45 }
46
47 void mostrar() {
48     Node* current = head;
49     while (current != nullptr) {
50         std::cout << current->data << " ";
51         current = current->next;
52     }
53     std::cout << std::endl;
54 }
55 };
```

```
56
57 int main() {
58     DoubleLinkedList dll;
59     dll.insertFrente(3);
60     dll.insertFrente(2);
61     dll.insertFrente(1);
62     dll.insertAtras(4);
63     dll.insertAtras(5);
64
65     dll.mostrar(); // Salida: 1 2 3 4 5
66
67     return 0;
68 }
```

La implementación se realizó utilizando una estructura "Node" con los campos para almacenar datos "Data" y dos punteros, uno para el nodo anterior "prev" y otro para el siguiente nodo "next". Además se crea una clase "DoubleLinkedList" la cual contiene los métodos para insertar elementos al frente "insertFrente" y atrás "insertAtras" de la lista así como también el método para mostrar "mostrar", la lista contendrá dos punteros "head" y "tail" que apuntan al primer y último nodo. En el método "insertFrente" se creará un nuevo nodo el cual actualiza el puntero del nuevo nodo y el actual, si es necesario actualizará el "head", con "insertAtras" el proceso es similar pero actualiza los punteros del nodo nuevo y el anterior, si es necesario actualizará el "tail". Para finalizar "mostrar" recorre la lista desde el "head" hasta el "tail" e imprimirá los valores de cada nodo.