

# Laboratorio 05

## Punteros

**Nombre:** Gabriel Fernando Rodriguez Cutimbo

**CUI:** 20212157

**Grupo:** B

**Repositorio GitHub:**

[https://github.com/gaco123/EPCC\\_CCII.git](https://github.com/gaco123/EPCC_CCII.git)

### 1. Ejercicios

Resolver los siguientes ejercicios planteados:

1. Asignar valores a dos variables enteras, intercambie estos valores almacenados usando solo punteros a enteros.

**Código:**

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

int main(){
    //Memoria
    int a;
    int b;
    int* swaper;

    //Input
    cout<<"Ingrese el valor del número a: ";
    cin>>a;
    cout<<"Ingrese el valor del número b: ";
    cin>>b;

    //Intercambio
    swaper=new int;
    *swaper=a;
    a=b;
    b=*swaper;

    //Output
    cout<<"Nuevo valor de a: "<<a<<"\n";
    cout<<"Nuevo valor de b: "<<b<<"\n";

    delete swaper;

    return 0;
```

}

**Funcionamiento:**

```
Ingrese el valor del número a: 10
Ingrese el valor del número b: 20
Nuevo valor de a: 20
Nuevo valor de b: 10
```

2. Cree dos vectores con valores flotantes y asígnele valores aleatorios, para esto deberá de asignar memoria a cada vector. Calcule el producto punto de vectores y muestre por pantalla. Una vez finalizado este proceso, retire la memoria asignada a los punteros. Repita este proceso de asignación y retiro de memoria dentro de un for de 1 000 000 veces.

**Código:**

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

int main(){
    //Semilla
    srand(time(NULL));
    //Memoria
    int n;
    float prodpunt;
    int max=10;
    int min=1;

    //Generacion y eliminacion de vectores dinámicos
    for(int i=0; i<1000000; i++){
        prodpunt=0;
        n=rand() % 10 + 1; //el tamaño del vector es de 0 a 10;
        float* vec1=new float[n];
        float* vec2=new float[n];
        for(int j=0; j<n; j++){
            vec1[j]=min+static_cast<float>(rand())*static_cast<float>(max-min)/RAND_MAX;
            vec2[j]=min+static_cast<float>(rand())*static_cast<float>(max-min)/RAND_MAX;
            prodpunt+=vec1[j]*vec2[j];
        }
        cout<<"V_1\n";
        for(int k=0; k<n; k++){
            cout<<vec1[k]<<"\t";
        }
        cout<<"\n";
        cout<<"V_2\n";
        for(int l=0; l<n; l++){
            cout<<vec2[l]<<"\t";
        }
        cout<<"\n";
        cout<<"El producto punto de V_1 y V_2 es: "<<prodpunt;
        cout<<"\n\n";
    }
}
```

```
        delete[] vec1;  
        delete[] vec2;  
    }  
  
    return 0;  
}
```

**Funcionamiento (solo se mostrará el final, ya que, al ser un bucle de 1 000 000, la consola envía demasiados mensajes):**

```
El producto punto de V_1 y V_2 es: 67.668  
  
V_1  
1.16425 6.0654 8.47093 6.11759 5.75173 8.42122 1.75149  
V_2  
8.03751 6.6617 6.83364 3.41844 9.46495 3.05835 1.98935  
El producto punto de V_1 y V_2 es: 212.243  
  
V_1  
6.85315 2.57247 4.80523 6.85589 8.21467 4.28693 9.90661  
V_2  
7.28574 8.30256 4.32594 5.95306 2.43239 6.1522 6.18571  
El producto punto de V_1 y V_2 es: 240.524  
  
V_1  
1.69079 9.77944 3.95157  
V_2  
7.41237 2.89685 3.87356  
El producto punto de V_1 y V_2 es: 56.1689  
  
V_1  
4.59181  
V_2  
3.80956  
El producto punto de V_1 y V_2 es: 17.4928  
  
<< El programa ha finalizado: codigo de salida: 0 >>  
<< Presione enter para cerrar esta ventana >>_
```

3. Construya una lista enlazada simple utilizando solo punteros. Añada las funciones de insertar y eliminar un elemento. En la función insertar se debe asegurar que los números insertados estén en orden creciente. Simule el proceso con 10000 números aleatorios sin que el programa falle.

### **Código:**

```
#include <iostream>  
#include <ctime>  
using namespace std;  
  
struct caja{  
    caja* ant;  
    int num;  
    caja* des;  
};  
  
void imprimir_lista(caja *i){  
    caja* k=i;  
    cout<<k->num<<"-";  
}
```

```
while(k->des!=NULL){
    k=k->des;
    cout<<k->num<<"- ";
}
cout<<"\n";
}

void insertar_elemento(caja* i){
    //Iterador para encontrar la ultima caja de la lista
    caja* k=i;

    //Caja nueva
    caja* j=new caja;
    j->num=rand() % 100 + 1;
    j->ant=NULL;
    j->des=NULL;

    //Busca la ultima caja de la lista
    while(k->des!=NULL){
        k=k->des;
    }
    k->des=j;

    //Hace que la nueva caja apunte a la ultima caja de la lista
    k->des->ant=k;

    //Asegura el orden creciente en la lista
    int temp;
    while(j->ant!=NULL && j->ant->num > j->num){
        //J al estar apuntando a la ultima caja, lo usamos para iterar a travez de la lista y ordenar todos
        los elementos de manera creciente
        temp=j->num;
        j->num=j->ant->num;
        j->ant->num=temp;
        j=j->ant;
    }

    j=NULL;
    delete j;
}

caja* borrar_elemento(caja* i, int a){
    caja* k=i;
    caja* temp=i;
    //El entero "a" es el número a buscar
    //La primera caja tiene un tratamiento diferente que a las demás cajas
    if(a==k->num){
        cout<<"El primer elemento \""<<a<<"\" a sido eliminado.\n";
        temp=k->des;
        k->des->ant=NULL;
        k->des=NULL;
        delete k;
    }
    else{
        //Empieza a bucar el elemento desde el comienzo de la lista enlazada
        while(k->des!=NULL){
            k=k->des;
            if(a==k->num&& k->des!=NULL){
                cout<<"El primer elemento \""<<a<<"\" a sido eliminado.\n";
            }
        }
    }
}
```

```
k->ant->des=k->des;
k->des->ant=k->ant;
k->ant=NULL;
k->des=NULL;
delete k;

break;
}
if(a==k->num&& k->des==NULL){
    cout<<"El primer elemento \""<<a<<"\" a sido eliminado.\n";
    k->ant->des=NULL;
    k->ant=NULL;
    delete k;
    break;
}
}
}
return temp;
}

int main(){
    //Semilla
    srand(time(NULL));

    //Caja inicial
    caja* i = new caja;
    i->ant=NULL;
    i->num=rand() % 100 + 1;
    i->des=NULL;

    //Insertar cajas
    for(int j=0; j<10000; j++){
        insertar_elemento(i);
    }
    cout<<"PRIMERA LISTA\n";
    imprimir_lista(i);
    //Borra el primer elemento buscado (en este caso el número "1"), empezando desde el principio de la
    lista enlazada
    i=borrar_elemento(i,100);
    cout<<"SEGUNDA LISTA\n";
    imprimir_lista(i);

    return 0;
}
```

**Funcionamiento (se mostrara una simulación con 100 números, aunque también funciona sin problemas al simular 10000 números):**

```
PRIMERA LISTA
2-3-4-4-5-7-8-10-10-12-13-13-14-15-18-18-20-22-22-23-23-24-24-24-25-25-29-29-30-30-30-31-32-32-32-33-36-37-37-37-38-38-3
9-39-39-43-45-47-48-49-50-50-51-52-52-52-53-53-58-60-61-61-61-63-64-64-67-67-67-68-72-73-74-75-76-77-78-80-81-81-83-84-8
4-84-86-88-88-90-92-93-95-95-95-95-96-97-97-98-99-100-100-
El primer elemento "100" a sido eliminado.
SEGUNDA LISTA
2-3-4-4-5-7-8-10-10-12-13-13-14-15-18-18-20-22-22-23-23-24-24-24-25-25-29-29-30-30-30-31-32-32-32-33-36-37-37-37-38-38-3
9-39-39-43-45-47-48-49-50-50-51-52-52-52-53-53-58-60-61-61-61-63-64-64-67-67-67-68-72-73-74-75-76-77-78-80-81-81-83-84-8
4-84-86-88-88-90-92-93-95-95-95-95-96-97-97-98-99-100-
```

4. Construya una lista enlazada que almacene tanto números como cadenas de texto utilizando punteros. Incluya una función de búsqueda de muestre un dato almacenado además del tipo de dato que se encuentra almacenado (int, float, char, ...)

**Código:**

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

struct caja{
    caja* ant;
    bool pnum=true;
    int num;
    char chr=' ';
    string str=" ";
    caja* des;
};

void imprimir_lista(caja *i){
    caja* k=i;
    cout<<k->num<<"-";
    while(k->des!=NULL){
        k=k->des;
        if(k->pnum!=false){
            cout<<k->num<<"-";
        }
        if(k->chr!=' '){
            cout<<k->chr<<"-";
        }
        if(k->str!=" "){
            cout<<k->str<<"-";
        }
    }
    cout<<"\n";
}

void insertar_elemento(caja* i, int a){
    //Iterador para encontrar la ultima caja de la lista
    caja* k=i;

    //Caja nueva
    caja* j=new caja;
    j->num=a;
    j->ant=NULL;
    j->des=NULL;

    //Busca la ultima caja de la lista
    while(k->des!=NULL){
        k=k->des;
    }
    //Hace que la ultima caja apunte a la nueva caja
    k->des=j;

    //Hace que la nueva caja apunte a la que era la ultima caja de la lista
```

```
k->des->ant=k;

j=NULL;
delete j;
}
void insertar_elemento(caja* i, char a){
    //Iterador para encontrar la ultima caja de la lista
    caja* k=i;

    //Caja nueva
    caja* j=new caja;
    j->pnum=false;
    j->chr=a;
    j->ant=NULL;
    j->des=NULL;

    //Busca la ultima caja de la lista
    while(k->des!=NULL){
        k=k->des;
    }
    //Hace que la ultima caja apunte a la nueva caja
    k->des=j;

    //Hace que la nueva caja apunte a la que era la ultima caja de la lista
    k->des->ant=k;

    j=NULL;
    delete j;
}
void insertar_elemento(caja* i, string a){
    //Iterador para encontrar la ultima caja de la lista
    caja* k=i;

    //Caja nueva
    caja* j=new caja;
    j->pnum=false;
    j->str=a;
    j->ant=NULL;
    j->des=NULL;

    //Busca la ultima caja de la lista
    while(k->des!=NULL){
        k=k->des;
    }
    //Hace que la ultima caja apunte a la nueva caja
    k->des=j;

    //Hace que la nueva caja apunte a la que era la ultima caja de la lista
    k->des->ant=k;

    j=NULL;
    delete j;
}
void buscar(caja* i, int a){
    //Iterador para encontrar la ultima caja de la lista
    caja* k=i;
    //Prueba para que no bote dos mensajes
```

```
bool prueba=true;
//Recorre todo la lista enlazada para buscar el elemento requerido
if(a==k->num){
    cout<<"SI, se encontro el (int) con valor: "<<a<<"\n";
}
else{
    while(k->des!=NULL){
        k=k->des;
        if(a==k->num){
            prueba=false;
            cout<<"SI, se encontro el (int) con valor: "<<a<<"\n";
            break;
        }
    }
    if(prueba==true){
        cout<<"NO, se encontro el (int) con valor: "<<a<<"\n";
    }
}
}

void buscar(caja* i, char a){
    //Iterador para encontrar la ultima caja de la lista
    caja* k=i;
    //Prueba para que no bote dos mensajes
    bool prueba=true;
    //Recorre todo la lista enlazada para buscar el elemento requerido
    if(a==k->chr){
        cout<<"SI, se encontro el (char) con valor: "<<a<<"\n";
    }
    else{
        while(k->des!=NULL){
            k=k->des;
            if(a==k->chr){
                prueba=false;
                cout<<"SI, se encontro el (char) con valor: "<<a<<"\n";
                break;
            }
        }
        if(prueba==true){
            cout<<"NO, se encontro el (char) con valor: "<<a<<"\n";
        }
    }
}

}

void buscar(caja* i, string a){
    //Iterador para encontrar la ultima caja de la lista
    caja* k=i;
    //Prueba para que no bote dos mensajes
    bool prueba=true;
    //Recorre todo la lista enlazada para buscar el elemento requerido
    if(a==k->str){
        cout<<"SI, se encontro el (string) con valor: "<<a<<"\n";
    }
    else{
        while(k->des!=NULL){
            k=k->des;
            if(a==k->str){
                prueba=false;
            }
        }
    }
}
```



```
        cout<<"SI, se encontro el (string) con valor: "<<a<<"\n";
        break;
    }
}
if(prueba==true){
    cout<<"NO, se encontro el (string) con valor: "<<a<<"\n";
}
}

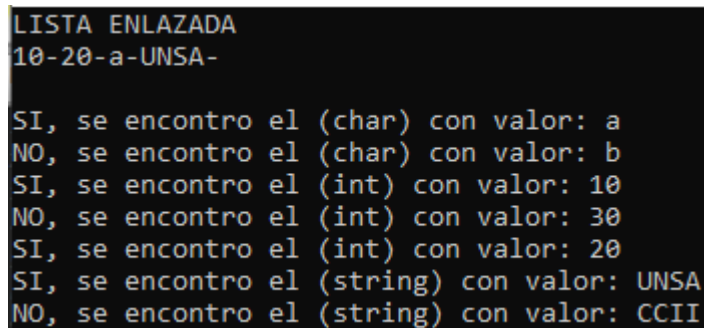
int main(){
    //Caja inicial
    caja* i = new caja;
    i->ant=NULL;
    i->num=10;
    i->des=NULL;

    //Insertar elementos a la lista enlazada
    insertar_elemento(i,20);
    insertar_elemento(i,'a');
    insertar_elemento(i,"UNSA");
    cout<<"LISTA ENLAZADA\n";
    imprimir_lista(i);
    cout<<"\n";

    //Búsqueda de elemetos
    buscar(i,'a');
    buscar(i,'b');
    buscar(i,10);
    buscar(i,30);
    buscar(i,20);
    buscar(i,"UNSA");
    buscar(i,"CCII");

    return 0;
}
```

### Funcionamiento:



```
LISTA ENLAZADA
10-20-a-UNSA-

SI, se encontro el (char) con valor: a
NO, se encontro el (char) con valor: b
SI, se encontro el (int) con valor: 10
NO, se encontro el (int) con valor: 30
SI, se encontro el (int) con valor: 20
SI, se encontro el (string) con valor: UNSA
NO, se encontro el (string) con valor: CCII
```

5. Implemente su propia función de concatenación de cadenas de texto especial (¡no es la función ordinaria de concatenación!), recibirá como parámetro dos punteros de caracteres y tendrá que asignar el contenido del segundo puntero al INICIO del primer puntero. La función no retorna ningún valor.

**Código:**

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

void concant_Cadenas(string* i, string* j){
    *i=*j+*i;
}

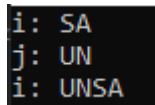
int main(){
    //Puntero I
    string* i = new string ("SA");

    //Puntero J
    string* j = new string ("UN");

    //Imprimir
    cout<< "i: "<<*i<<"\n";
    cout<< "j: "<<*j<<"\n";
    //Función
    concant_Cadenas(i,j);
    cout<< "i: "<<*i<<"\n";

    delete i;
    delete j;

    return 0;
}
```

**Funcionamiento:**

```
i: SA
j: UN
i: UNSA
```

6. Utilizando punteros a funciones, implemente las funciones:

- void sumar (int a, int b);
- void restar (int a, int b);
- void multiplicar (int a, int b);
- void dividir (int a, int b);

Cree un vector de punteros a funciones e implemente un programa que permita la invocación de cada función, pero a través del puntero.

### Código:

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

void Sum(int num1, int num2){
    cout<<"La suma de los dos números es: "<<num1+num2;
}
void Rest(int num1, int num2){
    cout<<"La resta de los dos números es: "<<num1-num2;
}
void Mult(int num1, int num2){
    cout<<"La multiplicación de los dos números es: "<<num1*num2;
}
void Div(int num1, int num2){
    cout<<"La división de los dos números es: "<<num1/num2;
}

int main(){
    int num1;
    int num2;
    int opc;
    void (*p[4])(int,int)={Sum,Rest,Mult,Div};

    cout<<"MENU DE OPERACIONES ARITMETICAS BASICAS"<<endl;
    cout<<"1. Suma"<<endl;
    cout<<"2. Resta"<<endl;
    cout<<"3. Multiplicación"<<endl;
    cout<<"4. División"<<endl;
    do{
        cout<<"Ingrese el número de la opción a elegir: ";
        cin>>opc;
    } while(opc<1||opc>4);

    cout<<"Ingrese el primer número: ";
    cin>>num1;
    cout<<"Ingrese el segundo número: ";
    cin>>num2;

    if(opc==1){
        (*p[opc-1])(num1,num2);
    }
    else if(opc==2){
        (*p[opc-1])(num1,num2);
    }
    else if(opc==3){
```

```
        (*p[opc-1])(num1,num2);  
    }  
    else if(opc==4){  
        (*p[opc-1])(num1,num2);  
    }  
  
    return 0;  
}
```

### Funcionamiento:

```
MENU DE OPERACIONES ARITMETICAS BASICAS  
1. Suma  
2. Resta  
3. Multiplicación  
4. División  
Ingrese el número de la opción a elegir: 1  
Ingrese el primer número: 10  
Ingrese el segundo número: 20  
La suma de los dos números es: 30
```

## 5. Entregables

Al final estudiante deberá:

1. Compactar el código elaborado y subirlo al aula virtual de trabajo. Agregue sus datos personales como comentario en cada archivo de código elaborado.
2. Elaborar un documento que incluya tanto el código como capturas de pantalla de la ejecución del programa. Este documento debe de estar en formato PDF.
3. El nombre del archivo (comprimido como el documento PDF), será su LAB05\_GRUPO\_A/B/C\_CUI\_1erNOMBRE\_1erAPELLIDO.

(Ejemplo: LAB05\_GRUPO\_A\_2022123\_PEDRO\_VASQUEZ).

4. Debe remitir el documento ejecutable con el siguiente formato:

LAB05\_GRUPO\_A/B/C\_CUI\_EJECUTABLE\_1erNOMBRE\_1erAPELLIDO

(Ejemplo: LAB05\_GRUPO\_A\_EJECUTABLE\_2022123\_PEDRO\_VASQUEZ).

En caso de encontrarse trabajos similares, los alumnos involucrados no tendrán evaluación y serán sujetos a sanción.