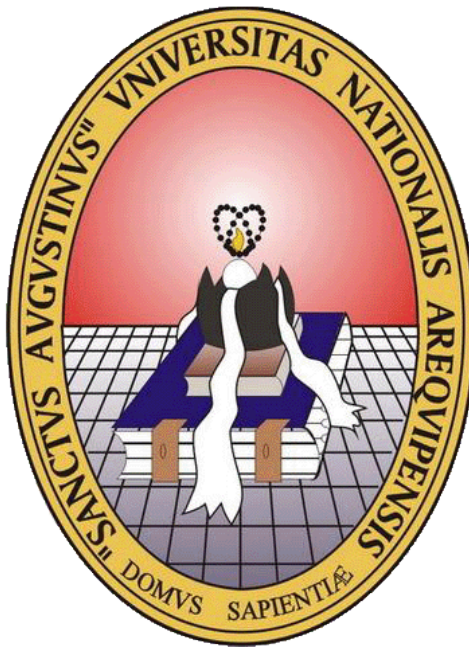


UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA  
FACULTAD DE PRODUCCIÓN Y SERVICIOS  
ESCUELA DE CIENCIAS DE LA COMPUTACIÓN



PRÁCTICA DE LABORATORIO 5  
CURSO DE CIENCIAS DE LA COMPUTACIÓN II

ESTUDIANTE:  
RUIZ MAMANI EDUARDO GERMÁN

EMAIL: [eruizm@unsa.edu.pe](mailto:eruizm@unsa.edu.pe)

CUI: 20193061

TURNO:

C

AREQUIPA- PERÚ

2021

## 1. EJERCICIO

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

- **CÓDIGO**

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

//EDUARDO GERMÁN RUIZ MAMANI
//20193061

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

void cambio (int *x, int *y) {
    int aux;

    aux = *x;
    *x = *y;
    *y = aux;
}

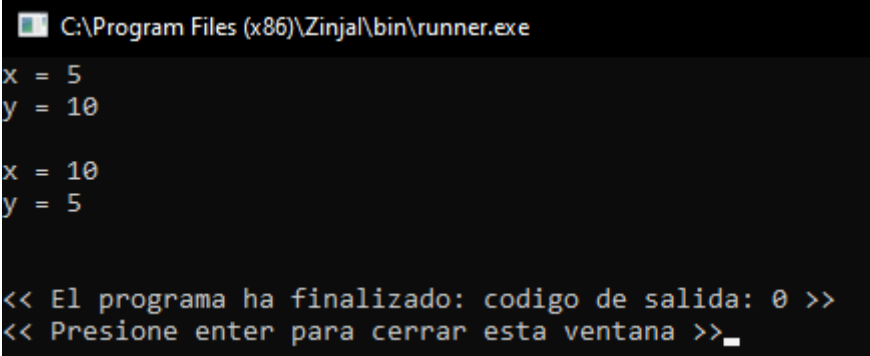
int main(int argc, char *argv[]) {
    int x = 5, y = 10;

    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << endl;

    cambio(&x, &y);

    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    return 0;
}
```

- **CAPTURAS**



```
C:\Program Files (x86)\Zinjal\bin\runner.exe
x = 5
y = 10

x = 10
y = 5

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>_
```

## 2. EJERCICIO

2. Cree dos vectores con valores flotantes y asigne 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<stdlib.h>
#include<time.h>
using namespace std;

//EDUARDO GERMAN RUIZ MAMANI
//20193061

//2. Cree dos vectores con valores flotantes y asigne 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.

void imprimir (float *v, int tam) {
    for (int i = 0; i < tam; i++)
        cout << v[i] << " ";
}

int main(int argc, char *argv[]) {

    float n = 1000000, prod;
    int tam = 3;
    int lim_inf = -5, lim_sup = 5;
    srand(time(NULL));

    float* v1;
    float* v2;
    v1 = new float [tam];
    v2 = new float [tam];

    for (int i = 0; i < n; i++) {

        cout << i+1 << endl;
        prod = 0;

        for (int j = 0; j < tam; j++) {
            v1[j] = lim_inf + rand()%(lim_sup+1-lim_inf);
            v2[j] = lim_inf + rand()%(lim_sup+1-lim_inf);
            prod = prod + v1[j]*v2[j];
        }

        cout << "v1 = [ ";
```

```

        imprimir(v1, tam);
        cout << "]" << endl;
        cout << "v2 = [ ";
        imprimir(v2, tam);
        cout << "]" << endl;

        cout << "Producto punto: " << prod;
        cout << endl;
        cout << endl;
    }

    delete[] v1;
    delete[] v2;
    return 0;
}

```

- **CAPTURAS**

```

C:\Program Files (x86)...
Producto punto: 18

1228
v1 = [ -4 1 -4 ]
v2 = [ 2 3 -2 ]
Producto punto: 3

1229
v1 = [ 1 -4 2 ]
v2 = [ 3 4 -3 ]
Producto punto: -19

```

### 3. EJERCICIO

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 <stdlib.h>
#include <time.h>
using namespace std;

//EDUARDO GERMAN RUIZ MAMANI
//20193061

//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.

struct casilla {
    int num;
    casilla *sig;
};

struct lista {
    casilla *cabeza;
};

void insertaraleatorio (int v, lista *e) {
    casilla *nuevo = new casilla;
    nuevo->num = v;
    if (e->cabeza == NULL) {
        nuevo->sig = NULL;
        e->cabeza = nuevo;
    } else {
        nuevo->sig = e->cabeza;
        e->cabeza = nuevo;
    }
}

void ordenar (lista *e, int tam) {
    casilla *temp1;
    casilla *temp2;
    int a = 0;

    while (tam != 1) {
        temp1 = e->cabeza;
        temp2 = (e->cabeza)->sig;

/*      cout << temp1->num << " " << temp2->num << endl;*/

        for (int i = 1; i < tam; i++) {
            if((temp1->num) > (temp2->num)){
                temp1->sig = temp2->sig;
                temp2->sig = temp1;

                if(temp1 == e->cabeza)
                    e->cabeza = temp2;

                if((temp1->sig != NULL) && ((temp1->num) > ((temp1->sig)->num)))
                    temp2->sig = temp1->sig;

                a=1;
            } else {
                if((temp2->sig != NULL) && ((temp2->num) > ((temp2->sig)->num)))
                    temp1->sig = temp2->sig;
                temp1 = temp2;
            }
            temp2 = temp1->sig;
        }
    }
}

```

```

        if (a == 0)
            break;
        a=0;

        tam--;
    }
}

void imprimir (lista *e) {
    casilla *temp = e->cabeza;

    while (temp != NULL) {
        cout << temp->num << " ";
        temp = temp->sig;
    }

    cout << endl;
}

void insertarelem (lista *e) {
    int v;

    cout << "LISTA ACTUAL:" << endl;
    imprimir(e);

    cout << "Numero a insertar: ";
    cin >> v;

    casilla *nuevo = new casilla;
    nuevo->num = v;

    if (e->cabeza == NULL) {
        nuevo->sig = NULL;
        e->cabeza = nuevo;
    } else if ((e->cabeza)->num >= nuevo->num) {
        nuevo->sig = e->cabeza;
        e->cabeza = nuevo;
    } else {
        casilla *temp1 = e->cabeza;
        casilla *temp2 = (e->cabeza)->sig;

        while (true)
            if ((nuevo->num) <= (temp2->num)) {
                nuevo->sig = temp2;
                temp1->sig = nuevo;
                break;
            } else if (temp2 == NULL) {
                nuevo->sig = NULL;
                temp1->sig = nuevo;
                break;
            } else {
                temp1 = temp2;
                temp2 = temp1->sig;
            }
    }
}

```

```

        }
    }

    cout << "LISTA ACTUALIZADA:" << endl;
    imprimir(e);
}

void eliminarelem (lista *e) {
    cout << "LISTA ACTUAL:" << endl;
    imprimir(e);

    int v;

    cout << "Numero a eliminar: ";
    cin >> v;

    casilla *temp1;
    casilla *temp2 = e->cabeza;

    while (true) {
        if (temp2 == NULL)
            cout << "El valor no se encuentra en la
lista";
        else
            if (temp2->num == v) {
                if (temp2 == e->cabeza)
                    e->cabeza = temp2->sig;
                else
                    temp1->sig = temp2->sig;

                delete temp2;

                cout << "\nLISTA ACTUALIZADA:" <<
endl;

                imprimir(e);

                break;
            } else {
                temp1 = temp2;
                temp2 = temp1->sig;
            }
        }
    }

}

void borrarcasillas (casilla *ref) {
    if (ref != NULL) {
        borrarcasillas(ref->sig);
        delete ref;
    }
}

int main(int argc, char *argv[]) {
    int tam, v, op;

    do {

```

```

        cout << "Tamaño de la lista: ";
        cin >> tam;

        if (tam < 0) {
            cout << "ERROR! Ingrese solo valores
positivos\n" << endl;
        }
    } while (tam < 0);

    int lim_inf = 0, lim_sup = 100;
    srand(time(NULL));

    lista *A = new lista;
    A->cabeza = NULL;
    for (int i = 0; i < tam; i++) {
        v = lim_inf + rand()%(lim_sup+1-lim_inf);
        insertaraleatorio (v, A);
    }
    cout << "\nLISTA ALEATORIA GENERADA:" << endl;
    imprimir(A);

    ordenar(A,tam);

    cout << "LISTA ORDENADA:" << endl;
    imprimir(A);

    do {
        op = 0;
        do {
            cout << "\nOperaciones:\n";
            cout << "1. Insertar elemento\n";
            cout << "2. Eliminar elemento\n";
            cout << "3. Salir\n";
            cout << "Operacion? ";
            cin >> op;

            if (op != 1 && op != 2 && op != 3) {
                cout << "ERROR! Ingrese un numero de
operacion valido\n" << endl;
            }
        } while (op != 1 && op != 2 && op != 3);

        if (op == 1)
            insertarelem(A);
        else if (op == 2)
            eliminarelem(A);
        else
            cout << "Gracias por confiar en nosotros!";
    } while (op != 3);

    borrarcasillas(A->cabeza);
    delete A;
    return 0;
}

```



- CAPTURAS

```
C:\Program Files (x86)\Zinjal\bin\runner.exe
Tamaño de la lista: 10

LISTA ALEATORIA GENERADA:
64 66 7 77 45 48 33 43 34 1
LISTA ORDENADA:
1 7 33 34 43 45 48 64 66 77

Operaciones:
1. Insertar elemento
2. Eliminar elemento
3. Salir
Operacion? 1
LISTA ACTUAL:
1 7 33 34 43 45 48 64 66 77
Numero a insertar: 64
LISTA ACTUALIZADA:
1 7 33 34 43 45 48 64 64 66 77

Operaciones:
1. Insertar elemento
2. Eliminar elemento
3. Salir
Operacion? 2
LISTA ACTUAL:
1 7 33 34 43 45 48 64 64 66 77
Numero a eliminar: 1

LISTA ACTUALIZADA:
7 33 34 43 45 48 64 64 66 77

Operaciones:
1. Insertar elemento
2. Eliminar elemento
3. Salir
Operacion? 4
ERROR! Ingrese un numero de operacion valido

Operaciones:
1. Insertar elemento
2. Eliminar elemento
3. Salir
Operacion? 3
Gracias por confiar en nosotros!

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>_
```

#### 4. EJERCICIO

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

- **CÓDIGO (no pude culminar, se me complicó el void)**

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

//EDUARDO GERMAN RUIZ MAMANI
//20193061

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

struct dato {
    void *d;
    dato *sig;
};

struct lista {
    dato *cabeza;
};

void insertar (void *v, lista *e) {
    dato *nuevo = new dato;
    nuevo->d = v;
    if (e->cabeza == NULL) {
        nuevo->sig = NULL;
        e->cabeza = nuevo;
    } else {
        nuevo->sig = e->cabeza;
        e->cabeza = nuevo;
    }
}

void imprimir (lista *e) {
    dato *temp = e->cabeza;

    while (temp != NULL) {
        cout << temp->d << " ";
        temp = temp->sig;
    }

    cout << endl;
}

void borrar_datos (dato *ref) {
    if (ref != NULL) {
```

```

        borrardatos(ref->sig);
        delete ref;
    }
}

int main(int argc, char *argv[]) {
    lista *A = new lista;
    A->cabeza = NULL;

    insertar("dia", A);
    imprimir(A);

    borrardatos(A->cabeza);
    delete A;
    return 0;
}

```

- **CAPTURAS  
ERROR**

## 5. EJERCICIO

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.h>
using namespace std;

//EDUARDO GERMAN RUIZ MAMANI
//20193061

//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.

void concat (string *cadena_1, string *cadena_2) {
    *cadena_1 = *cadena_2 + *cadena_1;
}

int main(int argc, char *argv[]) {
    string c1, c2;

    c1 = "Ruiz";
    c2 = "Eduardo";

    concat(&c1, &c2);
}

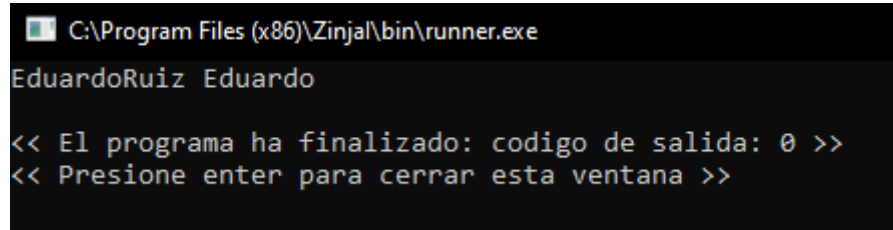
```

```

        cout << c1 << " " << c2;
        return 0;
    }

```

- **CAPTURAS**



```

C:\Program Files (x86)\Zinjal\bin\runner.exe
EduardoRuiz Eduardo
<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>

```

## 6. EJERCICIO

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;

//EDUARDO GERMAN RUIZ MAMANI
//20193061

//6. Utilizando punteros a funciones, implemente las funciones:
//  a. void sumar (int a, int b);
//  b. void restar (int a, int b);
//  c. void multiplicar (int a, int b);
//  d. 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

void resul(void(*func)(int,int), int a, int b){
    (*func)(a,b);
}

void sumar (int a, int b){
    cout << "SUMA: " << a+b << endl;
}

void restar (int a, int b){
    cout << "RESTA: " << a-b << endl;
}

void multiplicar (int a, int b){
    cout << "MULTIPLICACION: " << a*b << endl;
}

```

```

void dividir (int a, int b){
    cout << "DIVISION: " << (float)a/(float)b << endl;
}

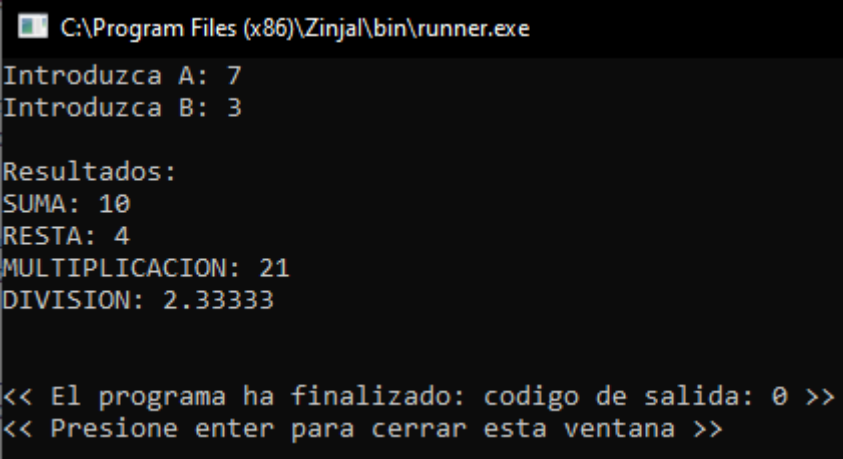
int main(int argc, char *argv[]) {
    int a, b;
    cout << "Introduzca A: ";
    cin >> a;
    cout << "Introduzca B: ";
    cin >> b;

    cout<<"\nResultados:\n";
    resul(sumar, a, b);
    resul(restar, a, b);
    resul(multiplicar, a, b);
    resul(dividir, a, b);

    return 0;
}

```

- **CAPTURAS**



```

C:\Program Files (x86)\Zinjal\bin\runner.exe
Introduzca A: 7
Introduzca B: 3

Resultados:
SUMA: 10
RESTA: 4
MULTIPLICACION: 21
DIVISION: 2.33333

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>

```