EJERCICIO 01:Asignar valores a dos variables enteras, intercambie estos valores almacenados usando solo punteros hacia enteros.

```
19  void intercambio(int *_a,int *_b){
20      int temp = *_a;
21      *_a=*_b;
22      *_b=temp;
23  }
24
25  int main(){
26      int a,b;
27      cout<<"Ingrese primer valor";cin>>a;
28      cout<<"ingrese segundo valor";cin>>b;
29
30      intercambio(&a,&b);
31      cout<<endl<<"el valor de a es de: "<<a<<" y el valor de b: "<<b;
32
33      return 0;
34  }
```

```
PS C:\Users\olmer\Documents\UNIVERSIDAD\2022\A\CIENCIAS
-II\LAB_05> & .\"ejercicio_01.exe"
Ingrese primer valor360
ingrese segundo valor400

el valor de a es de: 400 y el valor de b: 360
```

EJERCICIO 02: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.

```
20  void punto(int a[],int b[]){
21      int producto=0, punto=0;
22      for (int i = 0; i < 2; i++){
23          producto=((*(a+i))*((*(b+i))));
24          punto+=producto;
25      }
26
27      cout<<"el prodcuto punto es de: "<<punto<<endl;
28  }
```

```
29  int main(){
30      int vec1[2],vec2[2];
31      srand(time(0));
32      for (int j = 0; j < 1000; j++)
33      {
34          for (int i = 0; i < 10000; i++){
35          int *a,*b;
36          a=vec1;
37          b=vec2;
38
39          for (int i = 0; i < 2; i++){
40              *(a+i)=1+rand()%(9-1);
41              *(b+i)=1+rand()%(9-1);
42          }
43          /*for (int i = 0; i < 2; i++){
44              cout<<*(vec1+i)<<" ";
45          }
46          cout<<endl;
47          for (int i = 0; i < 2; i++){
48              cout<<*(vec2+i)<<" ";
49          }
50          */
51          punto(a, b);
52          delete [] a;
53          delete [] b;
54          a=NULL;
55          b=NULL;
56
57          }
58      }
```

```
el prodcuto punto es de: 54
el prodcuto punto es de: 45
el prodcuto punto es de: 42
el prodcuto punto es de: 97
el prodcuto punto es de: 26
el prodcuto punto es de: 49
el prodcuto punto es de: 75
el prodcuto punto es de: 36
el prodcuto punto es de: 50
el prodcuto punto es de: 80
el prodcuto punto es de: 51
el prodcuto punto es de: 100
el prodcuto punto es de: 30
el prodcuto punto es de: 73
el prodcuto punto es de: 84
el prodcuto punto es de: 52
el prodcuto punto es de: 49
el prodcuto punto es de: 38
el prodcuto punto es de: 25
el prodcuto punto es de: 8
el prodcuto punto es de: 84
```

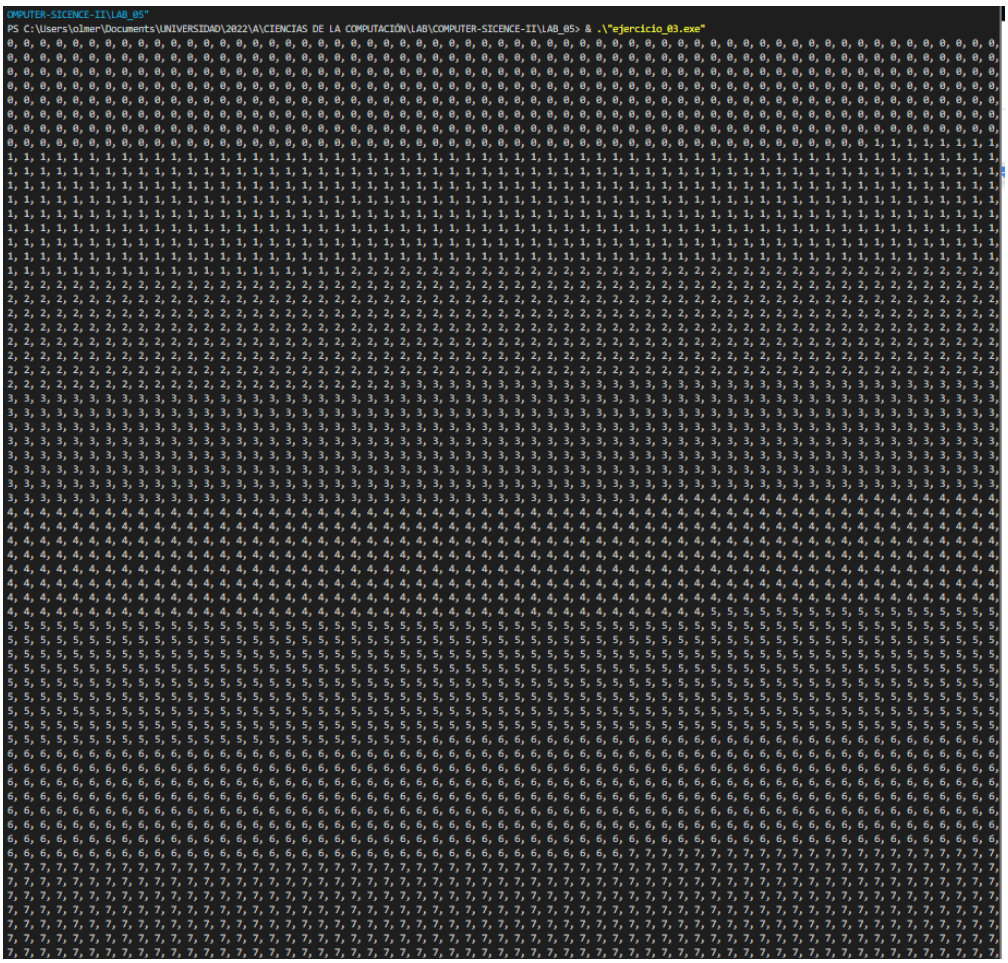
//No hay forma de capturar 1 000 000 de datos en una imagen :(

EJERCICIO 03: 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.

```
20  struct Nod{
21      int num;
22      Nod *sig;
23  };
24
25  void insertar(Nod *&, int);
26  void mostrar(Nod *);
27  void eliminar(Nod *&, int);
28
29
30  Nod *list = NULL;
31
32  int main(){
33      int numero, cont=0, num;
34
35      while (cont<10000){
36          num=0+rand()%(20-0);
37          insertar(list,num);
38          cont++;
39      }
40      mostrar(list);
41
42      return 0;
43  }
```

```
46  void insertar(Nod *&list, int _num){
47      Nod * new_nod=new Nod();
48      new_nod->num=_num;
49      Nod *tmp1 = list;
50      Nod *tmp2;
51
52      while ((tmp1 != NULL)&&(tmp1->num <_num)){
53          tmp2 = tmp1;
54          tmp1 = tmp1->sig;
55      }
56
57      if(list ==tmp1){
58          list = new_nod;
59      }
60      else{
61          tmp2->sig = new_nod;
62      }
63      new_nod->sig = tmp1;
64  }
```

```
77  void eliminar(Nod *&list, int buscar){
78      if(list != NULL){
79          Nod *temp;
80          Nod *ant=NULL;
81
82          temp=list;
83
84          while ((temp != NULL) && (temp->num !=buscar)){
85              ant=temp;
86              temp= temp->sig;
87          }
88
89          if(temp==NULL)cout<<"Dato no encontrado";
90
91          if(ant==NULL){
92              list=list->sig;
93              delete temp;
94          }
95          else{
96              ant->sig=temp->sig;
97              delete temp;
98          }
99
100     }
101 }
```



EJERCICIO 04: 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.

EJERCICIO 05: 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.

EJERCICIO 06: 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.

```
20  void sumar(int a, int b){
21      cout<<"La suma-> "<<a<<" + "<<b<<" = "<<a+b<<endl<<endl;
22  }
23  void restar(int a, int b){
24      cout<<"La resta-> "<<a<<" - "<<b<<" = "<<a-b<<endl<<endl;
25  }
26  void multiplicar(int a, int b){
27      cout<<"La multiplicacion-> "<<a<<" * "<<b<<" = "<<a*b<<endl<<endl;
28  }
29  void dividir(int a, int b){
30      cout<<"La dicision-> "<<a<<" + "<<b<<" / "<<a/b<<endl<<endl;
31  }
32
```

```
int main(){
    void (*ope[4]) (int,int) = {sumar,restar,multiplicar, dividir};
    int opc,a,b;
    do
    {
        cout<<"    OPERACIONES:"<<endl<<endl;
        cout<<"1. SUMAR"<<endl;
        cout<<"2. RESTAR"<<endl;
        cout<<"3. MULTIPLICAR"<<endl;
        cout<<"4. DIVIDIR"<<endl;
        cout<<"5. SALIR"<<endl;
        cin>>opc;
    } while (opc>5 || opc<1);

    switch (opc){
    case 1:
        system ("cls");
        cout<<"SUMAR: "<<endl<<endl;
        cout<<"Ingrese el primer elemento: ";cin>>a;
        cout<<"Ingrese el segundo elemento: ";cin>>b;
        (ope[0])(a,b);
        main();
        break;

    case 2:
        system ("cls");
        cout<<"RESTAR: "<<endl<<endl;
        cout<<"Ingrese el primer elemento: ";cin>>a;
        cout<<"Ingrese el segundo elemento: ";cin>>b;
        (ope[1])(a,b);
        main();
        break;
```

```
66      case 3:
67          system ("cls");
68          cout<<"MULTIPLICAR: "<<endl<<endl;
69          cout<<"Ingrese el primer elemento: ";cin>>a;
70          cout<<"Ingrese el segundo elemento: ";cin>>b;
71          (ope[2])(a,b);
72          main();
73          break;
74
75      case 4:
76          system ("cls");
77          cout<<"DIVIDIR: "<<endl<<endl;
78          cout<<"Ingrese el primer elemento: ";cin>>a;
79          cout<<"Ingrese el segundo elemento: ";cin>>b;
80          (ope[3])(a,b);
81          main();
82          break;
83
84      case 5:
85          return 0;
86          break;
87      }
88
89
90
91  }
```

```
SUMAR:

Ingrese el primer elemento: 13
Ingrese el segundo elemento: 16
La suma-> 13 + 16 = 29

   OPERACIONES:

1. SUMAR
2. RESTAR
3. MULTIPLICAR
4. DIVIDIR
5. SALIR
```

```
RESTAR:

Ingrese el primer elemento: 40
Ingrese el segundo elemento: 10
La resta-> 40 - 10 = 30

   OPERACIONES:

1. SUMAR
2. RESTAR
3. MULTIPLICAR
4. DIVIDIR
5. SALIR
```

```
MULTIPLICAR:

Ingrese el primer elemento: 15
Ingrese el segundo elemento: 10
La multiplicacion-> 15 * 10 = 150

   OPERACIONES:

1. SUMAR
2. RESTAR
3. MULTIPLICAR
4. DIVIDIR
5. SALIR
```

```
DIVIDIR:

Ingrese el primer elemento: 100
Ingrese el segundo elemento: 25
La dicision-> 100 / 25 = 4

   OPERACIONES:

1. SUMAR
2. RESTAR
3. MULTIPLICAR
4. DIVIDIR
5. SALIR
```