UNIVERSIDAD NACIONAL DE INGENIERÍA

Facultad de Ciencias

Escuela Profesional de Ciencia de la Computación

EXAMEN PARCIAL

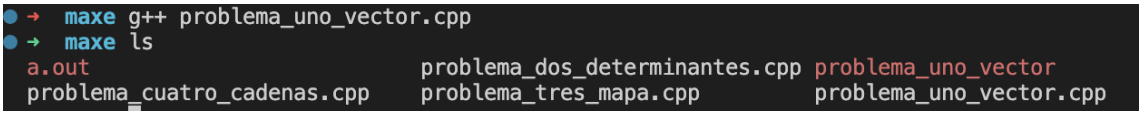
Curso: **Fundamentos de Programación CC112 - A**

Nombres y Apellidos: ELIAZ SEBASTIAN BOBADILLA CAMARENA

Código: 20244697E (c++)

```cpp
#include <iostream>

using namespace std;

void intercambiar(int &a, int &b)
{
  int temp = a;
  a = b;
  b = temp;
}

void PrintArray(int *vec, int tam, ostream &out)
{
  for (int i = 0; i < tam; ++i)
  {
    out << vec[i] << " ";
  }
  out << endl;
}

void DestroyVector(int *&vec)
{
  delete[] vec;
  vec = nullptr;
}

void CreateVector(int *&vec, int tam)
{
  vec = new int[tam];
}

void ReadArray(int *vec, int tam, istream &in)
{
  for (int i = 0; i < tam; ++i)
  {
    in >> vec[i];
  }
}

void Intercambiar(int *&a, int *&b)
{
  int *temp = a;
  a = b;
  b = temp;
}
// ELIAZ SEBASTIAN BOBADILLA CAMARENA 20244697E
```

```cpp
// Función para ordenamiento de burbuja recursivo ascendente
void BurbujaRecursivoAscendente(int *arr, int n)
{
  if (n <= 1)
    return;
  for (int j = 1; j < n; ++j)
  {
    if (arr[j] > arr[0])
    {
      intercambiar(arr[0], arr[j]);
    }
  }
  BurbujaRecursivoAscendente(arr + 1, n - 1);
}

// Función para ordenamiento de burbuja recursivo descendente
void BurbujaRecursivoDescendente(int *arr, int n)
{
  if (n <= 1)
    return;
  for (int j = 1; j < n; ++j)
  {
    if (arr[j] < arr[0])
    {
      intercambiar(arr[0], arr[j]);
    }
  }
  BurbujaRecursivoDescendente(arr + 1, n - 1);
}

int main()
{
  int *vec1 = nullptr, size = 10;
  int *vec2 = nullptr;

  CreateVector(vec1, size);
  ReadArray(vec1, size, cin);
  BurbujaRecursivoAscendente(vec1, size); // Ascendente
  PrintArray(vec1, size, cout);

  CreateVector(vec2, size);
  ReadArray(vec2, size, cin);
  BurbujaRecursivoDescendente(vec2, size); // Descendente
  PrintArray(vec2, size, cout);

  Intercambiar(vec1, vec2); // Intercambiar punteros

  DestroyVector(vec1);
  DestroyVector(vec2);
```


maxe g++ problema_uno_vector.cpp
maxe ls
a.out
problema_dos_determinantes.cpp
problema_uno_vector
problema_cuatro_cadenas.cpp
problema_tres_mapa.cpp
problema_uno_vector.cpp

DOS

```cpp
#include <iostream>

using namespace std;

double valorasbsoluto(double x)
{
  return x * ((x > 0) - (x < 0));
}

int redondeador(double x)
{
  if (x >= 0.0)
    return static_cast<int>(x + 0.5);
  else
    return static_cast<int>(x - 0.5);
}

double determinant(const double *A, int n)
{
  if (n == 1)
  {
    return A[0];
  }

  int bestRow = 0, maxZeros = -1;
  for (int i = 0; i < n; ++i)
  {
    int zeroCount = 0;
    for (int j = 0; j < n; ++j)
    {
      if (A[i * n + j] == 0.0)
        ++zeroCount;
    }
    if (zeroCount == n)
    {
      return 0.0;
    }
    if (zeroCount > maxZeros)
    {
      maxZeros = zeroCount;
      bestRow = i;
    }
  }

  double det = 0.0;
  double *sub = new double[(n - 1) * (n - 1)];
```

```cpp
  for (int j = 0; j < n; ++j)
  {
    double val = A[bestRow * n + j];
    if (val == 0.0)
      continue;

    for (int r = 0, si = 0; r < n; ++r)
    {
      if (r == bestRow)
        continue;
      for (int c = 0, sj = 0; c < n; ++c)
      {
        if (c == j)
          continue;
        sub[si * (n - 1) + sj] = A[r * n + c];
        ++sj;
      }
      ++si;
    }

    double sign = ((bestRow + j) % 2 == 0) ? 1.0 : -1.0;
    det += sign * val * determinant(sub, n - 1);
  }

  delete[] sub;
  return det;
}
// ELIAZ SEBASTIAN BOBADILLA CAMARENA 20244697E

int main()
{
  int n;
  cout << "Ingrese el tamaño de la matriz (máximo 10): ";
  if (!(cin >> n) || n < 1 || n > 10)
  {
    cout << "Tamaño inválido. Debe ser entre 1 y 10.\n";
    return 1;
  }

  double *A = new double[n * n];
  cout << "Ingrese los " << (n * n) << " elementos de la matriz:\n";
  for (int i = 0; i < n * n; ++i)
  {
    cin >> A[i];
  }

  double det = determinant(A, n);
  double absDiff = valorasbsoluto(det - redondeador(det));
```

```cpp
// ELIAZ SEBASTIAN BOBADILLA CAMARENA 20244697E

int main()
{
  int n;
  cout << "Ingrese el tamaño de la matriz (máximo 10): ";
  if (!(cin >> n) || n < 1 || n > 10)
  {
    cout << "Tamaño inválido. Debe ser entre 1 y 10.\n";
    return 1;
  }

  double *A = new double[n * n];
  cout << "Ingrese los " << (n * n) << " elementos de la matriz:\n";
  for (int i = 0; i < n * n; ++i)
  {
    cin >> A[i];
  }

  double det = determinant(A, n);
  double absDiff = valorasbsoluto(det - redondeador(det));
  if (absDiff < 1e-9)
  {
    cout << "Determinante = " << redondeador(det) << "\n";
  }
  else
  {
    cout << "Determinante = " << det << "\n";
  }

  delete[] A;
  return 0;
}
```

```
maxe g++ problema_dos_determinantes.cpp
maxe ./problema_uno_vector
maxe ls
a.out                          problema_dos_determinantes.cpp problema_uno_vector
problema_cuatro_cadenas.cpp    problema_tres_mapa.cpp         problema_uno_vector.cpp
maxe ./a.out
Ingrese el tamaño de la matriz (máximo 10): 3
Ingrese los 9 elementos de la matriz:
1 0 0
2 4 5
5 7 3
Determinante = -23
```

PROBLEMA 3

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

const int MAX = 100;

void mappa(int **mapa, bool **visitado, int filas, int columnas, int x, int y)
{
  if (x < 0 || x >= filas || y >= columnas || visitado[x][y] || mapa[x][y] == 1 || y < 0)
  {
    return;
  }

  visitado[x][y] = true;

  mappa(mapa, visitado, filas, columnas, x - 1, y);
  mappa(mapa, visitado, filas, columnas, x + 1, y);
  mappa(mapa, visitado, filas, columnas, x, y - 1);
  mappa(mapa, visitado, filas, columnas, x, y + 1);
}

int main()
{
  int filas, columnas;
  cout << "Ingrese el número de filas y columnas del mapa: ";
  cin >> filas >> columnas;

  if (filas > MAX || columnas > MAX)
  {
    cout << "Dimensiones exceden el máximo permitido de " << MAX << "x" << MAX << endl;
    return 1;
  }

  int **mapa = new int *[filas];
  for (int i = 0; i < filas; ++i)
  {
    mapa[i] = new int[columnas];
  }

  bool **visitado = new bool *[filas];
  for (int i = 0; i < filas; ++i)
  {
    visitado[i] = new bool[columnas];
    for (int j = 0; j < columnas; ++j)
      visitado[i][j] = false;
  }
```

```
    cout << "Ingrese los valores del mapa (0 = camino, 1 = obstáculo):\n";
    for (int i = 0; i < filas; ++i)
    {
      for (int j = 0; j < columnas; ++j)
      {
        cin >> mapa[i][j];
      }
    }

    int regiones = 0;

    for (int i = 0; i < filas; ++i)
    {
      for (int j = 0; j < columnas; ++j)
      {
        if (mapa[i][j] == 0 && !visitado[i][j])
        {
          mappa(mapa, visitado, filas, columnas, i, j);
          regiones++;
        }
      }
    }

    cout << "Mapa ingresado:\n";
    for (int i = 0; i < filas; ++i)
    {
      for (int j = 0; j < columnas; ++j)
      {
        cout << mapa[i][j];
      }
      cout << endl;
    }
    // ELIAZ SEBASTIAN BOBADILLA CAMARENA 20244697E

    cout << "Número de regiones transitables conectadas: " << regiones << endl;

    for (int i = 0; i < filas; ++i)
    {
      delete[] mapa[i];
      delete[] visitado[i];
    }
    delete[] mapa;
    delete[] visitado;

    return 0;
  }
```

```
maxe g++ problema_tres_mapa.cpp
maxe ./a.out
Ingrese el número de filas y columnas del mapa: 4 5
Ingrese los valores del mapa (0 = camino, 1 = obstáculo):
1 0 0 1 1
1 0 1 1 0
1 1 1 0 0
0 1 1 1 1
Mapa ingresado:
10011
10110
11100
01111
Número de regiones transitables conectadas: 3
```

cuatro

```cpp
// ELIAZ SEBASTIAN BOBADILLA CAMARENA 20244697E

#include <iostream>
using namespace std;

bool isAlnumChar(char c)
{
  return (c >= '0' && c <= '9') || (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z');
}

int main()
{
  const int MAXLEN = 1000;
  char *input = new char[MAXLEN + 1];

  cout << "Ingrese una cadena:\n";
  cin.getline(input, MAXLEN);

  char *clean = new char[MAXLEN + 1];
  char *src = input;
  char *dst = clean;
  while (*src)
  {
    if (isAlnumChar(*src))
    {
      *dst++ = *src;
    }
    ++src;
  }
  *dst = '\0';
  delete[] input;

  int len = 0;
  for (char *p = clean; *p; ++p)
  {
    ++len;
  }
```

```
  char *left = clean;
  char *right = clean + (len - 1);
  while (left < right)
  {
    char tmp = *left;
    *left++ = *right;
    *right-- = tmp;
  }

  int index = 0;
  for (int chunk = 1; index < len; ++chunk)
  {
    int remain = len - index;
    int take = (chunk < remain ? chunk : remain);
    for (int k = 0; k < take; ++k)
    {
      cout << clean[index + k];
    }
    cout << '\n';
    index += take;
  }

  delete[] clean;
  return 0;
}
```

```
→  maxe g++ problema_cuatro_cadenas.cpp
→  maxe ./a.out
Ingrese una cadena:
Hola, mundo feliz!
z
il
efo
dnum
aloH
```