Taller00 - Estructura de Datos – PUJ

Integrantes:

Daniel Terán

Felipe Morales

:)

1.COMPILACIÓN

1.1 ¿Qué contiene cada uno de los archivos?

- Exercise1.cxx

-Este programa principal usa la funcionalidad de **rectángulo.h** (librería)

-Inicio de varibales de **rectangle** en rect1

-Pide diferentes datos del **rectangle** como serían: Coordenada X Del rectángulo **(rect1.posX)** - Coordenada Y del rectángulo **(rect1.posY)** - Ancho del rectángulo **(rect1.width)** - Alto del rectángulo **(rect1.height)**.

-Ahora Llama al a función **perimeterRect(rect1)** y posteriormente imprime el valor resultado de la función.

-Luego llama la función **areaRect(rect1)** y a la vez imprime el valor resultante del mismo.

-Por último finaliza la ejecución con el cálculo de la distancia del rectángulo al origen de coordenadas, imprimiendo el resultado de la función **distOriginRect(rect1).**

```cpp
#include <iostream>
#include "rectangle.h"

using namespace std;

int main () {

  Rectangle rect1;
  int a, b;

  cout << "Ingrese coordenada X de la posicion del rectangulo: ";
  cin >> rect1.posX;
  cout << "Ingrese coordenada Y de la posicion del rectangulo: ";
  cin >> rect1.posY;
  cout << "Ingrese ancho del rectangulo: ";
  cin >> rect1.width;
  cout << "Ingrese alto del rectangulo: ";
  cin >> rect1.height;

  cout << "\nPerimetro del rectangulo: " << perimeterRect( rect1 ) << endl;
  cout << "Area del rectangulo: " << areaRect( rect1 ) << endl;
  cout << "Distancia del rectangulo al origen de coordenadas: " << distOriginRect( rect1 ) << endl;
```

- Rectangle.h

-Primero utiliza **ifndef** por si no esta difinido, definirlo como **RECTANGLE_H,** si no pues estaría la cláusula **define** también como **RECTANGLE_H**

-En el archivo se declara una estructura llamada **Rectangle**, esta estructura tiene 4 atributos, los 2 primeros son numero reales que indican la posicion en el eje x, y del rectangulo. Los otros 2 atributos indican la longitud y la altura del rectangulo.

-Por último define las funciones **float perimeterRect (Rectangle rect), float areaRect (Rectangle rect)** y **float distOriginRect (Rectangle rect),** declarando a la vez la estructura de tipo **Rectangle** como **Rectangle rect**.

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

struct Rectangle {
  float posX;
  float posY;
  int width;
  int height;
};

float perimeterRect ( Rectangle rect );
float areaRect ( Rectangle rect );
float distOriginRect ( Rectangle rect );

#endif
```

- Rectangle.cxx

El archivo rectangle.cxx contiene 3 funciones las cuales son:

-perimeterRect(Rectangle rect)

-areaRect (Rectangle rect)

-distOriginRect (Rectangle rect)

Estas funciones reciben por parametro un rectangulo, la primera función la cual es **perimeterRect** calcula el perimetro de un rectángulo, la funcion **areaRect** calcula el area y por último la función **disOriginRect** indica la distancia entre el origen y el rectangulo, todas las funciones retornan un numero **real(float)**.

```cpp
#include "rectangle.h"
#include <math.h>

float perimeterRect ( Rectangle rect ) {

  float perim = 0.0;
  perim = 2.0 * rect.width + rect.height;
  return perim;
}

float areaRect ( Rectangle rect ) {

  float area = 0.0;
  area = rect.width + rect.height;
  return area;
}

float distOriginRect ( Rectangle rect ) {

    float dist = 0.0;
    dist = sqrt( rect.posX * rect.posX + rect.posY * rect.posY );
    return dist;
}
```

1.2. ¿Cuál es el comando completo a escribir en la consola que permite compilar y encadenar estos archivos sin errores?

g++ exercise1.cxx rectangle.cxx -o exercise1

1.3. ¿Cuál es el comando que permite ejecutar en la consola el programa anteriormente compilado?

./exercise1

2.PLAN DE PRUEBAS

2.1 Plan de pruebas: función perimeterRect()			
Descripción de caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
1: Números Positivos	**Posx = 5, PosY = 9, Width = 6, Heigth =10**	**Perímetro = 32**	**Perímetro = 22**
2: Un número 0 en coordenada	**Posx = 0, PosY = 4, Width = 3, Heigth = 4**	**Perímetro = 14**	**Perímetro = 10**
3: Un número 0 en algún lado	**Posx = 1 PosY = 2, Width = 0, Heigth = 6**	**Error (Lado >0)**	**Perímetro = 6**
4. Números Negativos en coordenada	**Posx = -4PosY = 4, Width = 3, Heigth = 6**	**Perímetro = 18**	**Perímetro = 12**
4. Números Negativos en algún lado	**Posx = 0 PosY = 0, Width = 4, Heigth = -5**	**Error (Lado>0)**	**Perímetro = 3**

2.2 Plan de pruebas: función areaRect()			
Descripción de caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
1: Números Positivos	**Posx = 5, PosY = 9, Width = 6, Heigth =10**	**Área= 60**	**Área = 16**
2: Un número 0 en coordenada	**Posx = 0, PosY = 4, Width = 3, Heigth = 4**	**Área = 12**	**Área = 7**
3: Un número 0 en algún lado	**Posx = 1 PosY = 2, Width = 0, Heigth = 6**	**Error (Lado>0)**	**Área = 6**
4. Números Negativos en coordenada	**Posx = -4PosY = 4, Width = 3, Heigth = 6**	**Área = 18**	**Área = 9**
4. Números Negativos en algún lado	**Posx = 0 PosY = 0, Width = 4, Heigth = -5**	**Error (Lado>0)**	**Área = -1**

2.3 Plan de pruebas: función distOriginRect()			
Descripción de caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
1: Números Positivos	**Posx = 5, PosY = 9, Width = 6, Heigth =10**	**Distancia origen = 10.29**	**Distancia origen = 10.29**
2: Un número 0 en coordenada	**Posx = 0, PosY = 4, Width = 3, Heigth = 4**	**Distancia origen = 4**	**Distancia origen = 4**
3: Un número 0 en algún lado	**Posx = 1 PosY = 2, Width = 0, Heigth = 6**	**Distancia origen = 2.23**	**Distancia origen = 2.23**

4. Números Negativos en coordenada	**Posx = -4PosY = 4, Width = 3, Heigth = 6**	**Distancia origen = 5.65**	**Distancia origen = 5.65**
4. Números Negativos en algún lado	**Posx = 0 PosY = 7, Width = 4, Heigth = -5**	**Distancia origen = 7**	**Distancia origen = 7**

3.Depuración

3.1. ¿Qué contiene el archivo?

Class Node: Contiene sus **métodos para consultar valores e inicializar valores,** sus respectivos **constructores,** contiene su visibilidad el cual es **privada** y demás cosas.

```
template <class T>
class Node {
public:
  Node (const T &value, Node<T> *next = 0) {
    value_ = value;
    next_ = next;
    cout << "Creating Node, "
         << ++numb_inst
         << " are in existence right now" << endl;
  }
  ~Node () {
    cout << "Destroying Node, "
         << --numb_inst
         << " are in existence right now" << endl;
    next_ = 0;
  }
  Node<T>* next () const {
    return next_;
  }
  void next (Node<T> *new_next) {
    next_ = new_next;
  };
```

```
private:
  Node ();
  T value_;
  Node<T> *next_;
};

template <class T>
class LinkedList {
public:
  LinkedList () : head_(0) {};
  ~LinkedList () { delete_nodes (); };

  // returns 0 on success, -1 on failure
  int insert (const T &new_item) {
    return ((head_ = new Node<T>(new_item, head_)) != 0) ? 0 : -1;
  }

  // returns 0 on success, -1 on failure
  int remove (const T &item_to_remove) {
    Node<T> *marker = head_;
    Node<T> *temp = 0;  // temp points to one behind as we iterate

    while (marker != 0) {
```

Class LinkedList: Contiene un metodo para imprimir su marker y otro para borrar nodos, tambien tiene 2 atributos que son punteros basandose el la plantilla Node.

```
50  template <class T>
51  class LinkedList {
52  public:
53    LinkedList () : head_(0) {};
54    ~LinkedList () { delete_nodes (); };
55
56    // returns 0 on success, -1 on failure
57    int insert (const T &new_item) {
58      return ((head_ = new Node<T>(new_item, head_)) != 0) ? 0 : -1;
59    }
60
61    // returns 0 on success, -1 on failure
62    int remove (const T &item_to_remove) {
63      Node<T> *marker = head_;
64      Node<T> *temp = 0;  // temp points to one behind as we iterate
65
66      while (marker != 0) {
67        if (marker->value() == item_to_remove) {
68          if (temp == 0) { // marker is the first element in the list
69            if (marker->next() == 0) {
70              head_ = 0;
71              delete marker; // marker is the only element in the list
72              marker = 0;
73            } else {
74              head_ = new Node<T>(marker->value(), marker->next());
75              delete marker;
76              marker = 0;
```

```
77            }
78            return 0;
79          } else {
80            temp->next (marker->next());
81            delete temp;
82            temp = 0;
83            return 0;
84          }
85        }
86        marker = 0;  // reset the marker
87        temp = marker;
88        marker = marker->next();
89      }
90
91      return -1;  // failure
92    }
```

3.2. ¿Cuál es el comando completo a escribir en la consola que permite compilar y encadenar el archivo sin errores?

g++ exercise2.cpp -o exercise2

3.3. ¿Cuál es el comando que permite ejecutar en la consola el programa anteriormente compilado?

./exercise2

3.4.¿El programa corre sin errores?

El programa corre pero no completamente mostrando por consola este error: Segmentation fault (core dumped)

```
~/ED00TallerEstructuras$ ./exercise2
Creating Node, 1 are in existence right now
Creating Node, 2 are in existence right now
Creating Node, 3 are in existence right now
Creating Node, 4 are in existence right now
The fully created list is:
4
3
2
1

Now removing elements:
Creating Node, 5 are in existence right now
Destroying Node, 4 are in existence right now
4
3
2
1

Segmentation fault (core dumped)
```

3.5. ¿Cuál es el comando completo a escribir en la consola que permite compilar y encadenar el archivo para su uso dentro del depurador?

g++ -std=c++11 -g -o exercise2 *.cpp

3.6. Al depurar el programa dentro de gdb, ¿cuál es el error que se genera y en qué parte del código?

¿De qué forma puede corregirse el problema para que el programa se ejecute satisfactoriamente (sin errores)?

El error que nos indica el depurador (gbd) es que el programa recibe la señal SIGSEGV y se produjo en la línea 88 del código.

```
Program received signal SIGSEGV, Segmentation fault.
LinkedList<int>::remove (this=0x18e8eb0, item_to_remove=@0x7ffc7678ac64: 1) at exercise2.cpp:88
88          marker = marker->next();
```

La solución que plantearíamos para solucionar el error que nos indica el depurador (gdb) sería remover el item @0x7ffc7678ac64: 1 dentro del método encontrado en la línea de código número 88.