CAD system

Termín odevzdání:	**12.05.2013 23:59:59**
Hodnocení:	**0.0000**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	0 / 50 Volné pokusy + 50 Penalizované pokusy (-2 % penalizace za každé odevzdání)
Nápovědy:	0 / 0

Úkolem je navrhnout a implementovat sadu tříd, které umožní efektivní práci s vektorovou 2D grafikou.

Realizované třídy mají za úkol umožnit uživateli rychle vybírat 2D tvary, které dříve rozmístil na obrazovku. Předpokládáme, že výkres může obsahovat obdélníky, kruhy, trojúhelníky a konvexní mnohoúhelníky. Tyto geometrické tvary jsou popsané třídami `CRectangle`, `CCircle`, `CTriangle` a `CPolygon`. Geometrické tvary mají svoji jednoznačnou identifikaci (celé číslo `ID`) a informace o svém umístění. Při vytváření objektu - geometrického tvaru jsou informace o velikosti předány konstruktoru:

- `CRectangle ( int ID, int x1, int y1, int x2, int y2 )` reprezentuje obdélník, který je zadaný dvojicí protilehlých rohů,
- `CCircle ( int ID, int x, int y, int r)` reprezentuje kruh zadaný středem a poloměrem,
- `CTriangle ( int ID, CCoord a, CCoord b, CCoord c)` reprezentuje trojúhelník zadaný svými třemi vrcholy (deklarace souřadnice `CCoord` je uvedena níže) a
- `CPolygon ( int ID, int n, const CCoord * v )` reprezentuje konvexní mnohoúhelník, který má `n` vrcholů, souřadnice vrcholů jsou předané v poli `v` (hodnoty si musíte zkopírovat do vytvářeného objektu).

Pro ukládání a zpracování grafických objektů musíte navrhnout a realizovat kontejner `CScreen`, který bude splňovat následující rozhraní:

- konstruktor bez parametrů, který vytvoří prázdnou instanci obrazu,
- destruktor, který uvolní případné alokované prostředky,
- metodu `Add`, která vloží předaný geometrický tvar do kontejneru,
- metodu `Optimize`, která bude zavolaná po vložení posledního obrazce pomocí `Add`, ale před prvním dotazem pomocí `Test` (v této metodě si můžete uložená data vhodně zorganizovat).
- metodu `void Test (int x, int y, int & nr, int * & list )`, která otestuje, které objekty se nacházejí na zadané souřadnici `(x,y)` a vrátí jejích seznam. Délka vráceného seznamu bude předaná ve výstupním parametru `nr`, vlastní nalezené geometrické objekty budou vrácené prostřednictvím svých identifikátorů ID v poli `list`. Metoda je zodpovědná za alokaci pole, volající po použití pole uvolní voláním `delete [] list`. Pokud je seznam vracených objektů prázdný, metoda musí vrátit hodnotu `nr=0` a `list=NULL`.

Při realizaci kontejneru musíte zvolit vhodné vnitřní uspořádání. Požadujeme, aby vyhledávání v geometrických objektech bylo velmi rychlé (odezva myši pro uživatele musí být téměř okamžitá). Dále předpokládáme, že objektů může být v obrazu velmi mnoho (řádově např. statisíce) a že dotazů je řádově více než vkládání.

Úloha je náročnější jednak z hlediska návrhu (nevhodný návrh tříd může délku kódu znásobit) a dále i z hlediska algoritmického (vhodná struktura `CScreen`). Jedná se o soutěžní úlohu. Část hodnocení dostanete za její prosté správné vyřešení, další část hodnocení můžete získat tím, že realizujete řešení efektivnější než řešení Vašich kolegů. Vzhledem k soutěžním charakteru úlohy poskytují pedagogové pouze minimální podporu při hledání chyb.

```
#ifndef __PROGTEST__
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <cctype>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <vector>
#include <set>
#include <map>
#include <list>
#include <algorithm>
#include <memory>
using namespace std;

struct CCoord
 {
   CCoord   ( int x = 0, int y = 0 ) { m_X = x; m_Y = y; }
   int   m_X;
   int   m_Y;
 };
#endif /* __PROGTEST__ */

class CRectangle
 ...
  public:
```

```
   CRectangle ( int ID, int x1, int y1, int x2, int y2 );
   // dalsi Vase implementace

class CCircle
 ...
  public:
   CCircle ( int ID, int x, int y, int r );
   // dalsi Vase implementace

class CPolygon
 ...
  public:
   CPolygon ( int ID, int cnt, const CCoord * v );
   // dalsi Vase implementace

class CTriangle
 ...
  public:
   CTriangle ( int ID, CCoord a, CCoord  b, CCoord c );
   // dalsi Vase implementace

class CScreen
 ...
  public:
   // implicitni konstruktor
   // destruktor (pokud je potreba)
   // metoda/metody Add
   // metoda Test
   void Optimize ( void );
   void Test ( int x, int y, int & len, int * & list ) const;
   // pripadne dalsi Vase metody
```

Nápověda:

- Deklaraci třídy `CCoord` nechte v bloku podmíněného překladu. Deklarace je uvedena již v testovacím prostředí. Odstraněním podmíněného překladu byste způsobili duplicitní deklaraci a chybu při překladu.
- Neukládejte objekty pouze do pole. Při lineárním uložení nebude vyhledávání zvládnutelné v časovém limitu.
- Nemá cenu zkoušet ukládat rastrovanou podobu geometrických tvarů. Dotazy metodou `Test` mají souřadnice `x,y` v rozsahu od -1048576 do +1048576. Celkově máte paměťový limit cca 50MB.
- Pro uložení použijte nějakou stromovou strukturu - K-D strom, intervalový strom, segmentový strom, quad-tree, BVH strom, ...
- Využijte při návrhu OOP se všemi jeho výhodami. Snažte se šetřit klávesnici.
- Pokud to ve Vaší implementaci nebudete využívat, nemusíte ve třídách geometrických tvarů ani `CScreen` deklarovat kopírující konstruktory a přetěžovat operátory=. V této úloze nebudou tyto funkce testované.
- Objekt považujeme v metodě `Test` za vybraný, pokud je zadaná souřadnice uvnitř nebo na jeho hraně. Při testování uvnitř/vně kružnice se vyhněte odmocňování. Porovnávejte druhé mocniny (zde jsou to celá čísla), tedy nebude docházet k zaokrouhlení.
- Při testování nejsou zadávané nesmyslné tvary (nekonvexní, nulové šířky, ...).

```
int   * res, resLen;
CScreen  S0;
S0 . Add ( CRectangle ( 1, 10, 20, 30, 40 ) );
S0 . Add ( CRectangle ( 2, 20, 10, 40, 30 ) );
S0 . Add ( CTriangle ( 3, CCoord ( 10, 20 ), CCoord ( 20, 10 ), CCoord ( 30, 30 ) ) );
S0 . Optimize();
S0 . Test ( 0, 0, resLen, res );
 // resLen = 0, res = [ ]
delete [] res;
S0 . Test ( 21, 21, resLen, res );
 // resLen = 3, res = [ 1 2 3 ]
delete [] res;
S0 . Test ( 16, 17, resLen, res );
 // resLen = 1, res = [ 3 ]
delete [] res;
S0 . Test ( 30, 22, resLen, res );
 // resLen = 2, res = [ 1 2 ]
delete [] res;
S0 . Test ( 35, 25, resLen, res );
 // resLen = 1, res = [ 2 ]
delete [] res;

CScreen  S1;
S1 . Add ( CCircle ( 1, 10, 10, 15 ) );
S1 . Add ( CCircle ( 2, 30, 10, 15 ) );
S1 . Add ( CCircle ( 3, 20, 20, 15 ) );
S1 . Optimize();
```

```
S1 . Test ( 0, 0, resLen, res );
 // resLen = 1, res = [ 1 ]
delete [] res;
S1 . Test ( 15, 15, resLen, res );
 // resLen = 2, res = [ 1 3 ]
delete [] res;
S1 . Test ( 20, 11, resLen, res );
 // resLen = 3, res = [ 1 2 3 ]
delete [] res;
S1 . Test ( 32, 8, resLen, res );
 // resLen = 1, res = [ 2 ]
delete [] res;

CScreen  S2;
CCoord  vertex1[4] = { CCoord ( 10, 0 ), CCoord ( 20, 20 ), CCoord ( 30, 20 ), CCoord ( 40, 0 ) };
S2 . Add ( CPolygon ( 1, 4, vertex1 ) );
CCoord  vertex2[5] = { CCoord ( 20, 10 ), CCoord ( 10, 20 ), CCoord ( 25, 30 ), CCoord ( 40, 20 ), CCoord ( 30, 10 ) };
S2 . Add ( CPolygon ( 2, 5, vertex2 ) );
S2 . Optimize();
S2 . Test ( 25, 15, resLen, res );
 // resLen = 2, res = [ 1 2 ]
delete [] res;
S2 . Test ( 25, 25, resLen, res );
 // resLen = 1, res = [ 2 ]
delete [] res;
S2 . Test ( 15, 3, resLen, res );
 // resLen = 1, res = [ 1 ]
delete [] res;
S2 . Test ( 11, 10, resLen, res );
 // resLen = 0, res = [ ]
delete [] res;
```

V této úloze lze získat dodatečné bonusové body za řešení, které je kvalitnější (spolehlivější nebo rychlejší) než řešení referenční, případně než řešení ostatních studentů.

Výsledky

Skóre v soutěži: -- (0.00 % referenčního řešení)

Pořadí: --

Bonus: -- (soutěž je již uzavřená)

Referenční řešení