Учебный курс: Технологии программирования

Кафедра: ИС Факультет: ИТиП

Лабораторная работа №3. Перегрузка операторов

Выполнил: Трофимов В.А. Группа: 1511

Преподаватель: Повышев В.В.

# Задание

Согласно варианту описать указанные типы данных и поместить их в отдельный заголовочный файл, в нем же описать операторы, указанные в варианте. Реализацию функций поместить с отдельный cpp файл.

Квадрат на плоскости. Задается координатой левого верхнего угла, стороной квадрата и углом, на который квадрат повернут относительно оси OX. Равенство площадей квадратов (перегрузите операции ==, !=, <, >). Умножение квадрата на вещественное число (увеличивает сторону квадрата). Прибавление к квадрату вектора (смещение квадрата на указанный вектор).

Стек целых чисел глубиной не более 100. Добавление числа в стек (operator << ). Изъятие числа из стека (operator >>). Не забудьте написать простую функцию (не оператор) для вывода стека на экран – с ней будет удобнее производить отладку.

Написать программу, проверяющую правильность работы – для наглядности и лучшего усвоения материала использовать как явный, так и не явный метод вызова функций операторов (см. пример в конце задания).

# Входные/Выходные данные

## Консоль

>> 0 0 0 0

1

>>2

4

>>2 2

4 4

>>3 2 1

<<3 2 1

# Исходный текст

## Файл main.cpp

// Course: Programming technologies (C Plus Plus)

// Lab 3. Overload.

// Student: Trofimov V.A. Group: 1511

// Teacher: Povishev V.V.

// Created 21.02.2013 Modified: 22.02.2013

// Description: Program entry point.

#include "squarestack.h"

#include <iostream>

using namespace std;

int main() {

Square a, b;

a.x = 0; a.y = 0; a.size = 2;

a.angle = 0;

b.x = 2; b.y = 2; b.size = 2;

b.angle = 120;

cout << operator == (a, b) << endl;

operator \* (a, 2);

cout << a.size << endl;

Vector c;

c.x = 2; c.y = 2;

b = b + c;

cout << b.x << " " << b.y << endl;

Stack stack;

stack.capacity = 100;

Push(stack, 1);

stack << 2;

operator << (stack, 3);

int n;

stack >> n;

cout << n << " ";

cout << Pop(stack) << " ";

cout << Pop(stack);

return 0;

}

## Файл squarestack.h

// Course: Programming technologies (C Plus Plus)

// Lab 3. Overload.

// Student: Trofimov V.A. Group: 1511

// Teacher: Povishev V.V.

// Created 21.02.2013 Modified: 22.02.2013

// Description: Squarestack header file.

#pragma once

#include <iostream>

struct Vector {

double x, y;

};

struct Square {

double x, y;

double size;

double angle;

};

bool operator == (const Square & left, const Square & right);

bool operator != (const Square & left, const Square & right);

bool operator < (const Square & left, const Square & right);

bool operator > (const Square & left, const Square & right);

Square operator \* (Square & left, const double & right);

Square operator \* (const double & left, Square & right);

Square operator + (Square & left, const Vector & right);

Square operator + (const Vector & left, Square & right);

struct tdata {

int value;

tdata \* next;

};

struct Stack {

int size;

int capacity;

tdata \* last;

};

int operator >> (Stack & n, int & value);

int operator << (Stack & n, const int value);

bool isEmpty(const Stack & stack);

bool isFull(const Stack & stack);

int GetSize(const Stack & stack);

int SetCapacity(Stack & stack, const int a);

int Push(Stack & stack, const int a);

int Pop(Stack & stack);

void Print(const Stack & stack);

## Файл squarestack.cpp

// Course: Programming technologies (C Plus Plus)

// Lab 3. Overload.

// Student: Trofimov V.A. Group: 1511

// Teacher: Povishev V.V.

// Created 21.02.2013 Modified: 22.02.2013

// Description: Squarestack source code file.

#include "squarestack.h"

bool operator == (const Square & left, const Square & right) {

return left.size == right.size;

}

bool operator != (const Square & left, const Square & right) {

return left.size != right.size;

}

bool operator < (const Square & left, const Square & right) {

return left.size <= right.size;

}

bool operator > (const Square & left, const Square & right) {

return left.size > right.size;

}

Square operator \* (Square & left, const double & right) {

left.size \*= right;

return left;

}

Square operator \* (const double & left, Square & right) {

right.size \*= left;

return right;

}

Square operator + (Square & left, const Vector & right) {

left.x += right.x;

left.y += right.y;

return left;

}

Square operator + (const Vector & left, Square & right) {

right.x += left.x;

right.y += left.y;

return right;

}

bool isEmpty(const Stack & stack) {

return stack.size == 0;

}

bool isFull(const Stack & stack) {

return stack.size == stack.capacity;

}

int GetSize(const Stack & stack) {

return stack.size;

}

int SetCapacity(Stack & stack, const int a) {

return a > 0 ? 0 == (stack.capacity = a) : -1;

}

int Push(Stack & stack, const int a) {

if (stack.size == stack.capacity) return -1;

if (stack.last == NULL) {

tdata \* curr = (tdata\*)malloc(sizeof(tdata));

curr -> next = NULL; curr -> value = a;

stack.last = curr; stack.size = 1; return 0;

} else {

if (stack.capacity - stack.size > 0) {

tdata \* curr = (tdata\*)malloc(sizeof(tdata));

curr -> next = stack.last; stack.last = curr;

curr -> value = a; stack.size++; return 0;

} return -1;

}

}

int Pop(Stack & stack) {

if (stack.size == 0) {

std::cout << "Stack is empty. Last value is not an element.\n";

return 0;

} else {

tdata \* curr = stack.last;

stack.last = curr -> next; stack.size--;

int res = curr -> value; free(curr);

return res;

}

}

void Print(const Stack & stack) {

if (stack.size == 0) std::cout << "Stack is empty\n";

else {

tdata \* curr;

curr = stack.last;

do {

std::cout << curr -> value << " ";

curr = curr -> next;

} while (curr != NULL);

}

}

int operator >> (Stack & n, int & value) {

value = Pop(n);

return value;

}

int operator << (Stack & n, const int value) {

return Push(n, value);

}