**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Создание простых динамических структур данных на языке С++

Студент: Ваньков Д.А.

Группа: 8о-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2018

1. Постановка задачи

Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, в моем случае N-дерево, содержащее одну фигуру (шестиугольник, реализованный в ЛР1).

Классы должны удовлетворять следующим правилам:

· Требования к классу фигуры аналогичны требованиям из лабораторной работы 1.

· Классы фигур должны иметь переопределенный оператор вывода в поток std::ostream (<<). Оператор должен распечатывать параметры фигуры (тип фигуры, длины сторон, радиус и т.д).

· Классы фигур должны иметь переопределенный оператор ввода фигуры из потока std::istream (>>). Оператор должен вводить основные параметры фигуры (длины сторон, радиус и т.д).

· Классы фигур должны иметь операторы копирования (=).

· Классы фигур должны иметь операторы сравнения с такими же фигурами (==).

· Класс-контейнер должен содержать объекты фигур “по значению” (не по ссылке).

· Класс-контейнер должен иметь метод по добавлению фигуры в контейнер.

· Класс-контейнер должен иметь методы по получению фигуры из контейнера (определяется структурой контейнера).

· Класс-контейнер должен иметь метод по удалению фигуры из контейнера (определяется структурой контейнера).

· Класс-контейнер должен иметь перегруженный оператор по выводу контейнера в поток std::ostream (<<).

· Классы должны быть расположены в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

2. Решение задачи

Нужно четко понимать, чего мы хотим от программы, поэтому необходимо составить предварительные тесты, на которые впоследствии нужно опираться.

Пример предварительных нестандартных тестов, которые необходимо проверить:

Попробовать добавить фигуру с отрицательной стороной.

Удалить корень и распечатать дерево.

После написания тестов, опираясь на них, можно писать код.

По итогу, выполнив все шаги, в директории должно оказаться 8 файлов: ***Figure.h, hexagon.cpp, hexagon.h, Tree.cpp, Tree.h, TreeItem.cpp, TreeItem.h, main.cpp*** и ***makefile***.

3. Руководство по использованию программы

Краткий обзор модулей программы:

Figure.h (файл библиотеки, отвечающий за описание общего родительского класса фигур)

Hexagon.h (заголовочный файл, описывающий класс “Шестиугольник”)

Hexagon.cpp (файл, отвечающий за операции над шестиугольником, содержащий все необходимые функции и переменные, для этого класса)

Tree.h (заголовочный файл, описывающий контейнер “N-дерево”)

Tree.cpp (файл, отвечающий за операции над деревом, такие как вставка, удаление, печать и т.д.)

TreeItem.h (заголовочный файла, описывающий элемент дерева)

TreeItem.cpp (файл, отвечающий за связь элемента дерева с фигурами, которыми он будет заполняться)

main.cpp (модуль, отвечающий за вызов программы, содержащий

интерфейс, позволяющий работать с написанным в других файлах кодом)

makefile (файл, производящий сборку всех файлов вместе и создающий исполняемый)

После запуска исполняемого файла, программа предлагает в помощь пользовательское меню работы с кодом, где каждая введенная цифра является командой начала процесса. Пользователь, при взаимодействии с меню может запросить его повторный вызов, вызов необходимых функций и выход из программы. При вводе несуществующих состояний, указанных в меню программа просит ввести корректный запрос.

4. Листинг программы

**main.ccp**

#include <cstdlib>

#include <iostream>

#include "Hexagon.h"

#include "Tree.h"

#include "Item.h"

void help()

{

std::cout << "Press 1 to get help" << std::endl;

std::cout << "Press 2 to add figure in tree" << std::endl;

std::cout << "Press 3 to get height of tree" << std::endl;

std::cout << "Press 4 to print tree" << std::endl;

std::cout << "Press 5 to delete figure from tree" << std::endl;

std::cout << "Press 0 to exit" << std::endl;

}

int main(int argc, const char \* argv[]) {

help();

size\_t act;

TTree s;

while (true) {

std::cin >> act;

if (act == 0) break;

if (act > 6) {

std::cout << "Error: enter another parameter" << std::endl;

continue;

}

switch (act) {

case 1: {

help();

break;

}

case 2: {

std::cout << "Adding the hexagon" << std::endl;

std::cout << "Enter side of hexagon: " << std::endl;

Hexagon obj(std::cin);

s.Push(obj);

break;

}

case 3: {

std::cout << "The height of tree :" << s.GetHeight() << std::endl;

break;

}

case 4: {

std::cout << s << std::endl;

break;

}

case 5: {

std::cout << "Enter side of hexagon to delete him: " << std::endl;

int value;

std::cin >> value;

s.Pop(value);

break;

}

default: {

std::cout << "Error, incorrect data" << std::endl;

break;

}

}

}

return 0;

}

**Tree.h**

#ifndef OOP\_LAB2\_TREE\_H

#define OOP\_LAB2\_TREE\_H

#include "Hexagon.h"

#include "Item.h"

#include <stdbool.h>

class TTree{

public:

TTree();

TTree(const TTree &orig);

int GetNodesOnLevel(int level) {

return GetNodesOnLevel(root, level);

}

void Push(Hexagon &obj) {

Push(root, obj);

}

void Pop(int value){

Pop(&root, value);

}

friend std::ostream &operator<<(std::ostream &os, const TTree &tree);

virtual ~TTree();

private:

int GetHeight(TTreeItem\* item);

void Push(TTreeItem \*&item, Hexagon &hexagon);

void Pop(TTreeItem \*\*item, int value);

int GetNodesOnLevel(TTreeItem\* item, int level);

TTreeItem\* root;

};

#endif //OOP\_LAB2\_TREE\_H

#ifndef OOP\_LAB2\_TREE\_H

#define OOP\_LAB2\_TREE\_H

#include "Hexagon.h"

#include "Item.h"

#include <stdbool.h>

class TTree{

public:

TTree();

TTree(const TTree &orig);

TTreeItem\* GetRoot(){

return root;

}

int GetNodesOnLevel(int level) {

return GetNodesOnLevel(root, level);

}

void Push(Hexagon &obj) {

Push(root, obj);

}

void Pop(int value){

Pop(&root, value);

}

friend std::ostream &operator<<(std::ostream &os, const TTree &tree);

virtual ~TTree();

private:

int GetHeight(TTreeItem\* item);

void Push(TTreeItem \*&item, Hexagon &hexagon);

void Pop(TTreeItem \*\*item, int value);

int GetNodesOnLevel(TTreeItem\* item, int level);

TTreeItem\* root;

};

#endif //OOP\_LAB2\_TREE\_H

**Tree.cpp**

#include <iostream>

#include "Tree.h"

#include "Item.h"

#include "Hexagon.h"

TTree::TTree() : root(nullptr) {

}

TTree::TTree(const TTree &orig) {

root = orig.root;

}

void TreeDel(TTreeItem\* item) {

if(item) {

TreeDel(item->GetBrother());

TreeDel(item->GetSon());

delete item;

}

}

TTree::~TTree() {

TreeDel(root);

}

int TTree::GetNodesOnLevel(TTreeItem \*item, int level) {

if (item == nullptr) {

return 0;

}

if (level <= 0) {

return 0;

}

return GetNodesOnLevel(item->GetSon(), level - 1) + (level == 1) + GetNodesOnLevel(item->GetBrother(), level);

}

void TTree::Push(TTreeItem \*&item, Hexagon &hexagon) {

if (item == nullptr) {

item = new TTreeItem(hexagon);

} else if (hexagon.GetSide() < item->GetHexagon().GetSide() ) {

Push(item->GetSon(), hexagon);

} else {

Push(item->GetBrother(), hexagon);

}

}

TTreeItem \*tree\_find\_parent(TTreeItem \*item, int value) {

if (item == nullptr) {

return nullptr;

}

if(item->GetHexagon().GetSide() == value) {

return item;

}

TTreeItem \*parent\_node = tree\_find\_parent(item->GetSon(), value);

if (parent\_node == nullptr) {

parent\_node = tree\_find\_parent(item->GetBrother(), value);

}

return parent\_node;

}

int tree\_find\_parent1(TTreeItem \*\*item, int value) {

int idx = 0;

if ( (\*item)->GetHexagon().GetSide() == value) {

return 1;

}

if ((\*item)->GetSon() != nullptr){

idx = tree\_find\_parent1(&((\*item)->GetSon()) , value);

}

if ((\*item)->GetBrother() != nullptr){

idx = tree\_find\_parent1(&((\*item)->GetBrother()), value);

}

if (idx == 1) {

delete((\*item)->GetBrother());

(\*item)->GetBrother() = nullptr;

idx = 0;

}

return 0;

}

void split(TTreeItem \*\*item) {

TTreeItem \*tmp = (\*item)->GetBrother();

(\*item)->GetSon() = tmp->GetSon();

(\*item)->GetHexagon() = tmp->GetHexagon();

(\*item)->GetBrother() = tmp->GetBrother();

delete(tmp);

tmp = nullptr;

}

TTreeItem \*destroy\_Tree(TTreeItem \*\*pointer) {

if((\*pointer) == nullptr) {

return nullptr;

}

if ((\*pointer)->GetSon() != nullptr) {

destroy\_Tree(&((\*pointer)->GetSon()));

}

if ((\*pointer)->GetBrother() != nullptr) {

destroy\_Tree(&((\*pointer)->GetBrother()));

}

if ((\*pointer)->GetSon() == nullptr && (\*pointer)->GetBrother() == nullptr) {

delete((\*pointer));

\*pointer = nullptr;

}

return nullptr;

}

void TTree::Pop(TTreeItem \*\*item, int value) {

TTreeItem \*pointer = tree\_find\_parent(\*item, value);

destroy\_Tree(&(pointer->GetSon()));

if (pointer->GetBrother() != nullptr) {

split(&pointer);

} else if (pointer->GetBrother() == nullptr) {

if ((\*item)->GetSon() == nullptr) {

delete(\*item);

\*item = nullptr;

} else {

TTreeItem \*\*son = &((\*item)->GetSon());

if ((\*son)->GetHexagon().GetSide() == value) {

delete(\*son);

\*son = nullptr;

} else {

tree\_find\_parent1(item, value);

}

}

}

}

void TSpace(size\_t n){ // Функция расставляющая пробелы

for (size\_t i=0;i<=n;i++)

std::cout << " ";

}

void TreeRun(std::ostream &os, TTreeItem \*item,size\_t space){ //Функция обхода дерева для печати

if (item) {

TSpace(space);

os << \*item << std::endl;

if (item->GetBrother() != nullptr) {

TreeRun(os, item->GetBrother(),space);

}

if (item->GetSon() != nullptr) {

TreeRun(os, item->GetSon(),space+1);

}

}

}

std::ostream& operator<<(std::ostream &os, const TTree &tree) {

TTreeItem \*obj = tree.root;

os << "Printed tree:" << std::endl;

TreeRun(os, obj,1);

return os;

}

**TreeItem.h**

#ifndef OOP\_LAB2\_ITEM\_H

#define OOP\_LAB2\_ITEM\_H

#include "Hexagon.h"

class TTreeItem {

public:

TTreeItem(const Hexagon& hexagon);

TTreeItem(const TTreeItem &orig);

friend std::ostream& operator<<(std::ostream &os, const TTreeItem &obj);

TTreeItem\*& GetSon();

TTreeItem\*& GetBrother();

Hexagon GetHexagon() const;

virtual ~TTreeItem();

private:

Hexagon hexagon;

TTreeItem \*son;

TTreeItem \*brother;

};

#endif //OOP\_LAB2\_ITEM\_H

**TreeItem.cpp**

#include <iostream>

#include <cstdlib>

#include "Item.h"

#include "Hexagon.h"

TTreeItem::TTreeItem(const Hexagon& hexagon) {

this->hexagon = hexagon;

this->son = nullptr;

this->brother = nullptr;

std::cout << "Tree item: created" << std::endl;

}

TTreeItem::TTreeItem(const TTreeItem &orig) {

this->hexagon = orig.hexagon;

this->son = orig.son;

this->brother = orig.brother;

std::cout << "Tree item: copied" << std::endl;

}

Hexagon TTreeItem::GetHexagon() const {

return this->hexagon;

}

TTreeItem\*& TTreeItem::GetSon() {

return this->son;

}

TTreeItem\*& TTreeItem::GetBrother() {

return this->brother;

}

TTreeItem::~TTreeItem() {

std::cout << "Tree item: deleted" << std::endl;

son = nullptr;

brother = nullptr;

}

std::ostream& operator<<(std::ostream &os, const TTreeItem &obj) {

os << "[" << obj.GetHexagon().GetSide() << "]";

return os;

}

**Hexagon.h**

#ifndef OOP\_LAB2\_HEXAGON\_H

#define OOP\_LAB2\_HEXAGON\_H

#include <iostream>

#include <cstdlib>

#include "Figure.h"

class Hexagon : public Figure{

public:

Hexagon();

Hexagon(std::istream &is);

Hexagon(size\_t i);

size\_t GetSide();

double Square() override;

void Print() override;

virtual ~Hexagon();

private:

size\_t side\_a;

};

#endif //OOP\_LAB2\_HEXAGON\_H

**Hexagon.cpp**

#include "Hexagon.h"

#include <istream>

#include <cmath>

Hexagon::Hexagon() :Hexagon(0) {

}

Hexagon::Hexagon(std::istream &is) {

int ptr;

std::cout << "Insert side a:" << std::endl;

is >> ptr;

while (ptr < 0) {

std:: cout << "Please, insert correct data" << std:: endl;

is >> ptr;

}

side\_a = ptr;

}

Hexagon::Hexagon(size\_t i) : side\_a(i){

std::cout << "Hexagon created: " << side\_a << std::endl;

}

double Hexagon::Square() {

return 1.5 \* pow(side\_a,2) \* sqrt(3);

}

void Hexagon::Print() {

std::cout << "Side: a = " << side\_a << std::endl;

}

size\_t Hexagon::GetSide(){

return side\_a;

}

Hexagon::~Hexagon() {}

**Тесты:**

C:\Users\dvankov\CLionProjects\OOP\_lab2\cmake-build-debug\OOP\_lab2.exe

Press 1 to get help

Press 2 to add figure in tree

Press 3 to get a number of nodes on level

Press 4 to print tree

Press 5 to delete figure from tree

Press 0 to exit

2

2

Adding the hexagon

Enter side of hexagon:

Insert side a:

12

12

Hexagon created: 0

Tree item: created

2

2

Adding the hexagon

Enter side of hexagon:

Insert side a:

5

5

Hexagon created: 0

Tree item: created

2

2

Adding the hexagon

E

Enter side of hexagon:

Insert side a:

6

6

Hexagon created: 0

Tree item: created

2

2

Adding the hexagon

Enter side of hexagon:

Insert side a:

7

7

Hexagon created: 0

Tree item: created

2

2

Adding the hexagon

Enter side of hexagon:

Insert side a:

3

3

Hexagon created: 0

Tree item: created

4

4

Printed tree:

[12]

[5]

[6]

[7]

[3]

1

1

Press 1 to get help

Press 2 to add figure in tree

Press 3 to get a number of nodes on level

Press 4 to print tree

Press 5 to delete figure from tree

Press 0 to exit

3

3

Enter level to get a number of nodes

2

2

On level 2 nodes: 3

5

5

Enter side of hexagon to delete him:

5

5

Tree item: deleted

Tree item: deleted

4

4

Printed tree:

[12]

[5]

[7]

0

0

Tree item: deleted

Tree item: deleted

Tree item: deleted

Tree deleted

Process finished with exit code 0

5. Выводы

Главной целью данной лабораторной работы было закрепление навыков работы с классами и объектами, а также закрепить знания работы с динамическими структурами.

В C++ удобно работать с объектами: можно реализовать виртуальный класс родитель, в котором описать основные принципы объектов и создать его подклассы в виде конкретных объектов с реализованными для них процедурами. Можно создавать процедуры с одним и тем же именем, который для данных различных типов будут работать по-разному. Для каждого объекта можно легко самому написать конструктор и деструктор. Все это значительно упрощает работу с объектами и написание кода в целом.

СПИСОК ЛИТЕРАТУРЫ

1. Дьюхерст C. Скользкие места С++.

2. Ханспетер Мессенбок, "Плюсы и минусы объектно-ориентированного программирования", ИнфоАрт, 1995.

3.Бьерн Страуструп — “Язык программирования С++” (Издание 3 специальное).