|  |  |
| --- | --- |
|  | ***Министерство образования и науки Российской Федерации***  *Калужский филиал федерального государственного бюджетного образовательного учреждения высшего профессионального образования*  ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | "Фундаментальные науки" |
| **КАФЕДРА** | "Программное обеспечение ЭВМ, информационные технологии и прикладная математика" |

**О Т Ч Е Т**

Лабораторная работа №3

|  |  |  |
| --- | --- | --- |
| **ДИСЦИПЛИНА:** | | "Программирование" |
| **ТЕМА:** | "Группа и итераторы" | |

|  |  |
| --- | --- |
| Выполнил: студент гр. ИТД.Б-31 | Турченко С.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Проверил: | Пчелинцева Н.И.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

|  |  |  |
| --- | --- | --- |
| Дата сдачи (защиты) работы: | |  |
| Результаты сдачи (защиты):  Количество рейтинговых баллов |  | |
| Оценка | зачтено | |

Калуга, 2015 г.

**Цель:** Получить практические навыки создания групп объектов.

**Постановка задачи:**

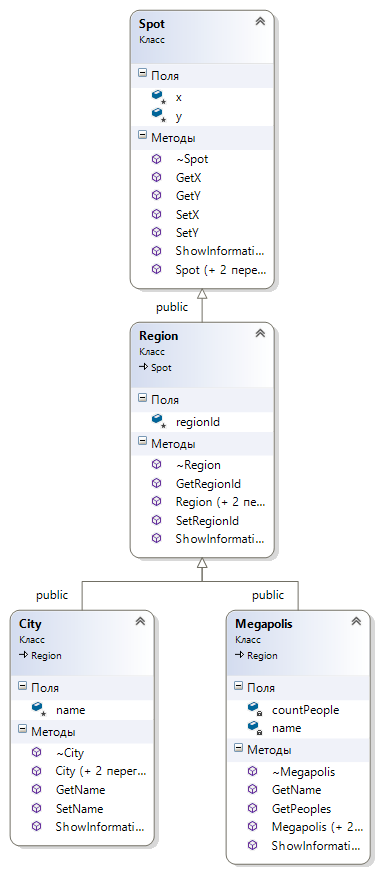
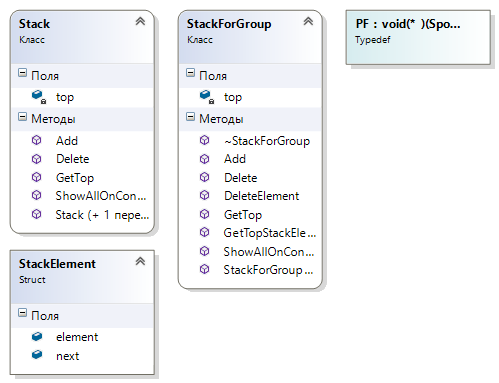
1. Дополнить иерархию классов лабораторной работы № 2 классами “группа”.

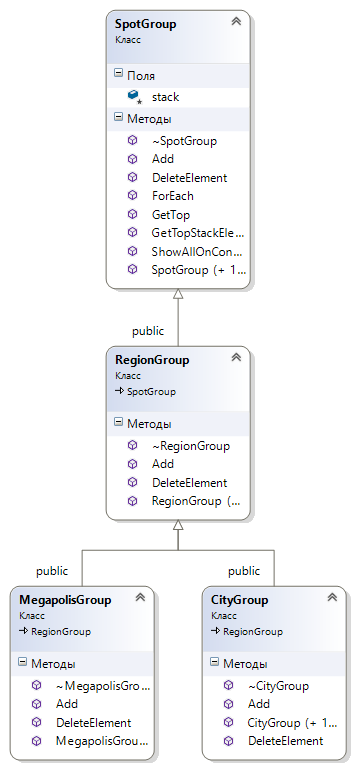
2. Написать для класса-группы метод-итератор.

3. Написать процедуру или функцию, которая выполняется для всех объектов, входящих в группу.

4. Написать демонстрационную программу, в которой создаются, показываются и разрушаются объекты-группы, а также демонстрируется использование итератора.

**Иерархия классов:**





**Иерархия классов была дополнена “группами**”:

typedef void(\*PF)(Spot\*);

class SpotGroup

{

protected:

StackForGroup stack;

public:

SpotGroup();

SpotGroup(Spot \*);

~SpotGroup();

Spot\* GetTop();

StackElement\* GetTopStackElement();

void ForEach(PF);

virtual void Add(Spot \*);

virtual void DeleteElement(Spot\*);

void ShowAllOnConsole();

};

class RegionGroup : public SpotGroup

{

public:

RegionGroup();

RegionGroup(Spot \*);

~RegionGroup();

virtual void Add(Spot \*);

virtual void DeleteElement(Spot\*);

};

class CityGroup : public RegionGroup

{

public:

CityGroup();

CityGroup(Spot \*);

~CityGroup();

void Add(Spot \*);

void DeleteElement(Spot\*);

};

class MegapolisGroup : public RegionGroup

{

public:

MegapolisGroup();

MegapolisGroup(Spot \*);

~MegapolisGroup();

void Add(Spot \*);

void DeleteElement(Spot\*);

};

**Файл реализации:**

#include "lists.h";

#include "groups.h";

#include <iostream>;

using namespace std;

SpotGroup::SpotGroup()

{

}

SpotGroup::SpotGroup(Spot \* spot)

{

Add(spot);

}

SpotGroup::~SpotGroup()

{

}

void SpotGroup::Add(Spot \* spot)

{

if (typeid(\*spot) == typeid(Spot))

{

stack.Add(spot);

}

else

{

cout << "Type of element does not match with type of group!" << endl;

}

}

Spot \* SpotGroup::GetTop()

{

return stack.GetTop();

}

StackElement \* SpotGroup::GetTopStackElement()

{

return stack.GetTopStackElement();

}

void SpotGroup::ForEach(PF procedure)

{

int count = 0;

StackElement \* tmp = stack.GetTopStackElement();

while (tmp != NULL)

{

procedure(tmp->element);

count++;

tmp = tmp->next;

}

if (count == 0) cout << "Not found elements!";

cout << endl;

}

void SpotGroup::DeleteElement(Spot \* spot)

{

stack.DeleteElement(spot);

}

void SpotGroup::ShowAllOnConsole()

{

stack.ShowAllOnConsole();

}

RegionGroup::RegionGroup()

{

}

RegionGroup::RegionGroup(Spot \* spot)

{

Add(spot);

}

RegionGroup::~RegionGroup()

{

}

void RegionGroup::Add(Spot \* spot)

{

if (typeid(\*spot) == typeid(Region))

{

stack.Add(spot);

}

else

{

cout << "Type of element does not match with type of group!" << endl;

}

}

void RegionGroup::DeleteElement(Spot \* spot)

{

if (typeid(\*spot) == typeid(Region))

{

stack.DeleteElement(spot);

}

else

{

cout << "Type of element does not match with type of group!" << endl;

}

}

CityGroup::CityGroup()

{

}

CityGroup::CityGroup(Spot \* spot)

{

Add(spot);

}

CityGroup::~CityGroup()

{

}

void CityGroup::Add(Spot \* spot)

{

if (typeid(\*spot) == typeid(City))

{

stack.Add(spot);

}

else

{

cout << "Type of element does not match with type of group!" << endl;

}

}

void CityGroup::DeleteElement(Spot \* spot)

{

if (typeid(\*spot) == typeid(City))

{

stack.DeleteElement(spot);

}

else

{

cout << "Type of element does not match with type of group!" << endl;

}

}

MegapolisGroup::MegapolisGroup()

{

}

MegapolisGroup::MegapolisGroup(Spot \* spot)

{

Add(spot);

}

MegapolisGroup::~MegapolisGroup()

{

}

void MegapolisGroup::Add(Spot \* spot)

{

if (typeid(\*spot) == typeid(Megapolis))

{

stack.Add(spot);

}

else

{

cout << "Type of element does not match with type of group!" << endl;

}

}

void MegapolisGroup::DeleteElement(Spot \* spot)

{

if (typeid(\*spot) == typeid(Megapolis))

{

stack.DeleteElement(spot);

}

else

{

cout << "Type of element does not match with type of group!" << endl;

}

}

**Основная программа:**

void main()

{

CityGroup cityGroup(new City(1, 1, 42, "Kaluga"));

cityGroup.Add(new City(1, 1, 32, "Bryansk"));

cityGroup.Add(new City(1, 1, 32, "Klimovo"));

cityGroup.Add(new City(1, 1, 77, "Moscow"));

cityGroup.Add(new City(1, 1, 0, "Unknown"));

cityGroup.ShowAllOnConsole();

cout << "Type region id>";

cin >> REGION;

PF procedure = GetByRegion;

cout << "Cites:" << endl;

cityGroup.ForEach(procedure);

}

**АТД Место**

Данные

Значения, которые содержат информацию о месте, ее расположение.

Операции

*Конструктор*

Начальные значения: Нули

Процесс: Инициализировать начальные значения.

SetX

Вход: Значение X.

Предусловия: Нет.

Процесс: Установить значение поля x.

Выход: Нет.

Постусловия: Нет.

SetY

Вход: Значение Y.

Предусловия: Нет.

Процесс: Установить значение поля y.

Выход: Нет.

Постусловия: Нет.

GetY

Вход: Нет.

Предусловия: Нет.

Процесс: Получить значение поля y.

Выход: Значение Y.

Постусловия: Нет.

GetX

Вход: Нет.

Предусловия: Нет.

Процесс: Получить значение поля x.

Выход: Значение X.

Постусловия: Нет.

ShowInformationOnConsole

Вход: Нет.

Предусловия: Нет.

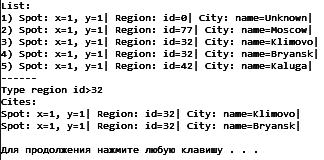
Процесс: Показать всю информацию о месте на консоль.

Выход: Нет.

Постусловия: Нет.

**Конец АТД Животное.**

**Результат выполнения:**



**Вывод:** Получены практические навыки создания групп объектов и применения метода-итератора.