|  |  |
| --- | --- |
|  | ***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 | Лысенков А.Е.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 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:**

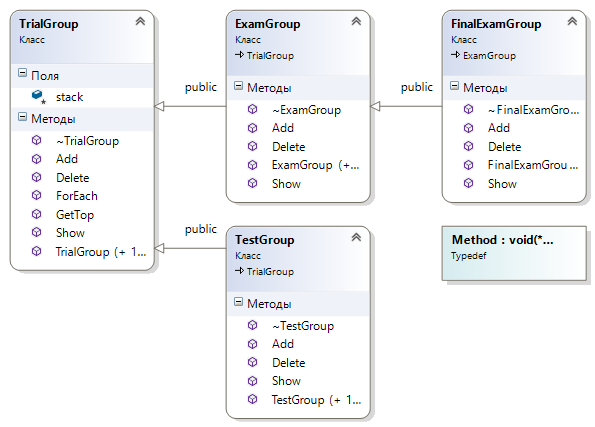
1. Дополнить иерархию классов лабораторной работы № 2 классами “группа”.

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(\*Method)(Trial\*);

class TrialGroup

{

protected:

Stack stack;

public:

TrialGroup();

TrialGroup(Trial\*);

~TrialGroup();

Trial \*GetTop();

void ForEach(Method);

virtual void Add(Trial\*);

virtual void Delete();

void Show();

};

class TestGroup : public TrialGroup

{

public:

TestGroup();

TestGroup(Test\*);

~TestGroup();

virtual void Add(Test\*);

virtual void Delete();

void Show();

};

class ExamGroup : public TrialGroup

{

public:

ExamGroup();

ExamGroup(Exam\*);

~ExamGroup();

virtual void Add(Exam\*);

virtual void Delete();

void Show();

};

class FinalExamGroup : public ExamGroup

{

public:

FinalExamGroup();

FinalExamGroup(FinalExam\*);

~FinalExamGroup();

virtual void Add(FinalExam\*);

virtual void Delete();

void Show(); };

**Файл реализации:**

TrialGroup::TrialGroup()

{}

TrialGroup::TrialGroup(Trial \*obj)

{

Add(obj);

}

TrialGroup::~TrialGroup()

{}

void TrialGroup::Add(Trial \*obj)

{

if (typeid(\*obj) == typeid(Trial))

{

stack.Add(obj);

}

}

void TrialGroup::ForEach(Method method)

{

Element \*tmp = stack.top;

if (tmp == NULL)

{

cout << "No elements";

}

for (int i = 0; tmp != NULL; i++)

{

method(tmp->obj);

tmp = tmp->next;

}

}

void TrialGroup::Show()

{

stack.Show();

}

void TrialGroup::Delete()

{

if (typeid(stack.top) == typeid(Trial))

{

stack.Get();

}

}

Trial \*TrialGroup::GetTop()

{

return stack.top->obj;

}

TestGroup::TestGroup()

{}

TestGroup::TestGroup(Test \*obj)

{

Add(obj);

}

TestGroup::~TestGroup()

{}

void TestGroup::Add(Test \*obj)

{

if (typeid(\*obj) == typeid(Test))

{

stack.Add(obj);

}

}

void TestGroup::Delete()

{

if (typeid(stack.top) == typeid(Test))

{

stack.Get();

}

}

ExamGroup::ExamGroup()

{

stack = NULL;

}

ExamGroup::ExamGroup(Exam \*obj)

{

Add(obj);

}

ExamGroup::~ExamGroup()

{

}

void ExamGroup::Add(Exam \*obj)

{

if (typeid(\*obj) == typeid(Exam))

{

stack.Add(obj);

}

}

void ExamGroup::Delete()

{

if (typeid(stack.top) == typeid(Exam))

{

stack.Get();

}

}

FinalExamGroup::FinalExamGroup()

{

stack = NULL;

}

FinalExamGroup::FinalExamGroup(FinalExam \*obj)

{

Add(obj);

}

FinalExamGroup::~FinalExamGroup()

{}

void FinalExamGroup::Add(FinalExam \*obj)

{

if (typeid(\*obj) == typeid(FinalExam))

{

stack.Add(obj);

}

}

void FinalExamGroup::Delete()

{

if (typeid(stack.top) == typeid(FinalExam))

{

stack.Get();

}

}

**Основная программа:**

int user = 0;

void PrintPassed(Trial \*obj)

{

Test \*work = dynamic\_cast<Test\*>(obj);

if (work->GetScore > work->GetMinScore)

{

work->Print();

}

}

void PrintGreater(Trial \*obj)

{

Test \*work = dynamic\_cast<Test\*>(obj);

if (work->GetScore > user)

{

work->Print();

}

}

int main()

{

TestGroup group(new Test("Math", "1st half", 65, 100, 20));

group.Add(new Test("Prog", "13 test", 75, 100, 20));

group.Add(new Test("Math", "27 test", 30, 100, 20));

group.Add(new Test("Phisics", "0 test", 10, 100, 20));

group.Show();

cout << "-- All who passed: " << endl;

Method method = PrintPassed;

group.ForEach(method);

cout << endl << "-- Enter min one:";

cin >> user;

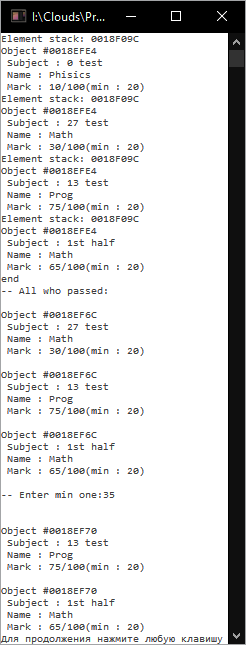
cout << endl;

Method method0 = PrintGreater;

group.ForEach(method0);

}

**Результат выполнения:**



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