Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования  
«Пермский национальный исследовательский политехнический университет»

(ПНИПУ)

**Факультет:** электротехнический

**Направление:** Промышленная мехатроника и робототехника

ОТЧЁТ

По лабораторной работе №4

«Классы»

Семестр 1

Выполнил:

студент группы ПРТ-21-1б

Поздняков Е.С.

Пермь 2022

**Задание №1**

**Постановка задачи**

Базовый класс:

ТРОЙКА ЧИСЕЛ (Triad)

First – int

Second – int

Third – int

Определить методы изменения полей и увеличения полей на 1.

Создать производный класс DATE с полями год, месяц и число. Переопределить методы увеличения полей на 1 и определить метод увеличения даты на n дней.

**Анализ**

1. Пользователь вводит значение длин двух катетов.
2. Создается экземпляр класса A, класс А содержит поля first и second – значения двух катетов, функции инициализации переменных и функцию hypotenuse, которая возвращает значение гипотенузы по заданных значениям катетов.
3. При помощи функций SetFirst и SetSecond устанавливаются значения длин катетов. Далее переменной h присваивается результат работы функции hypotenuse.
4. Полученный результат выводится.

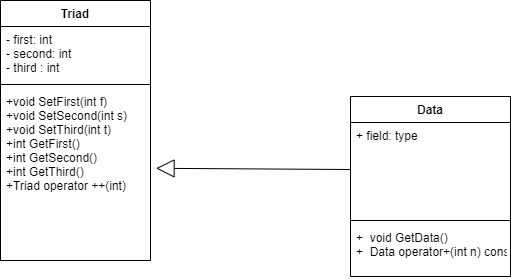
**Используемые переменные**

int n – количество дней;

Triad m1 – тройка чисел;

Data d1 – дата;

**UML-диаграмма**



**Код C++**

#include <iostream>

using namespace std;

class Data;

class Triad

{

protected:

int first;

int second;

int third;

public:

Triad() : first(0), second(0), third() {}

Triad(int f, int s, int t) : first(f), second(s), third(t) {}

Triad(const Triad &t)

{

first = t.first;

second = t.second;

third = t.third;

}

~Triad(){}

void SetFirst(int f){first = f;}

void SetSecond(int s){second = s;}

void SetThird(int t){third = t;}

int GetFirst(){return first;}

int GetSecond(){return second;}

int GetThird(){return third;}

Triad operator +()

{

Triad t;

t.first = this->first + 1;

t.second = this->second + 1;

t.third = this->third + 1;

return t;

}

Triad operator ++(int)

{

Triad t;

t.first = this->first + 1;

t.second = this->second + 1;

t.third = this->third + 1;

return t;

}

friend istream& operator >> (istream&in, Triad&t);

friend ostream& operator << (ostream&out, const Triad&t);

};

istream& operator>>(istream&in, Triad&t)

{

cout << "Input first: ";

in >> t.first;

cout << "Input second: ";

in >> t.second;

cout << "Input third: ";

in >> t.third;

return in;

}

ostream& operator<<(ostream&out, const Triad&t)

{

return (out << "{" << t.first << ", " << t.second << ", " << t.third << "}");

}

class Data : public Triad

{

public:

Data() : Triad() {}

Data(int f, int s, int t) : Triad(f, s, t)

{}

Data(const Data &t)

{

first = t.first;

second = t.second;

third = t.third;

}

~Data(){}

void GetData()

{

cout << first << "."<< second << "." << third << endl;

}

Data operator+(int n) const

{

Data t;

t.first = this->first + n;

if (t.first > 31)

{

t.first = this->first + n - 31;

t.second = this->second + 1;

if (t.second > 12)

{

t.second = 1;

t.third = this->third + 1;

}

else

{

t.third = this->third;

}

}

else

{

t.second = this->second;

t.third = this->third;

}

if (t.first > 31)

{

t.first = t.first - 31;

t.second++;

}

return t;

}

};

int main()

{

int n;

Triad m1(1, 2, 2003);

cout << "m1 = " << m1 << endl;

m1 = m1++;

cout << "m1 + 1 = " << m1 << endl;

Data d1(28, 12, 2003);

cout << "d1 = ";

d1.GetData();

cout << "Input amount of days: ";

cin >> n;

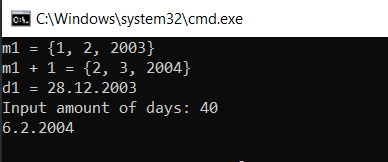
d1 = d1 + n;

d1.GetData();

return 0;

}

**Скриншоты результатов**

****

**Анализ результатов**

Изначально заданы значения тройки чисел. К каждому числу прибавляется 1. Полученная тройка выводится на экран. Выводится значение даты d1. Пользователь вводит количество дней, которые нужно прибавить к дате. Полученный результат выводится на экран.

**Контрольные вопросы**

1. Для чего используется механизм наследования?

Для создания новых классов на основе уже существующих.

1. Каким образом наследуются компоненты класса, описанные со спецификатором public?

Если тип наследования protected, то protected, если public – public, если private – private.

1. Каким образом наследуются компоненты класса, описанные со спецификатором private?

Не наследуются.

1. Каким образом наследуются компоненты класса, описанные со спецификатором protected?

При типе наследования public\protected – protected, при типе наследования private наследуются как private.

1. Каким образом описывается производный класс?

Class name1 : <тип наследования> name2

1. Наследуются ли конструкторы?

Не наследуются, но могут быть вызваны для передачи параметров.

1. Наследуются ли деструкторы?

Не наследуются

1. В каком порядке конструируются объекты производных классов?

Компоненты базового класса, базовый класс, компоненты производного класса, производный класс.

1. В каком порядке уничтожаются объекты производных классов?

В обратном порядке конструирования.

1. Могут ли быть виртуальными конструкторы? Деструкторы?

Конструкторы не могут, деструкторы могут.

1. Что представляют собой виртуальные функции и механизм позднего связывания?

Виртуальные функции – функции, которые могут быть переопределены в производном классе.

1. Наследуется ли спецификатор virtual?

Наследуется

1. Какое отношение устанавливает между классами открытое наследование?

Производный класс – частный случай базового класса.

1. Какое отношение устанавливает между классами закрытое наследование?

Производный имеет доступ ко всему функционалу базового класса, кроме private полей, но ни наследники, ни экземпляры класса не имеют доступ к этому функционалу.

1. В чем заключается принцип подстановки?

Если класс B является частным случаем класса A, то везде, где был использован класс A, можно использовать класс B.

1. String name

17.

Student ()

{

this->name = "-";

}

Employee() : Student()

{}

Teacher() : Employee() {}

18.

Student (string a)

{

this->name = a;

}

Employee(string a)

{

this->name = a;

}

Teacher(string a)

{

this->name = a;

}

19.

Student(const Student& a)

{

this->name = a.name;

}

Employee (const Employee& a)

{

this->name = a.name;

}

Teacher (const Teacher& a)

{

this->name = a.name;

}

20.

Student operator=(const Student& a)

{

Student temp;

temp.name = a.name;

return temp;

}

Employee operator=(const Employee& a)

{

Employee temp;

temp.name = a.name;

return temp;

}

Teacher operator=(const Teacher & a)

{

Teacher temp;

temp.name = a.name;

return temp;

}