Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.4**

Дисциплина: основы алгоритмизации и программирования

Тема: “Классы и объекты. Принцип подстановки.”

Вариант 10

Выполнил:

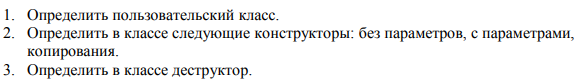
студент группы ИВТ-20-2Б Галинов О.Ю.

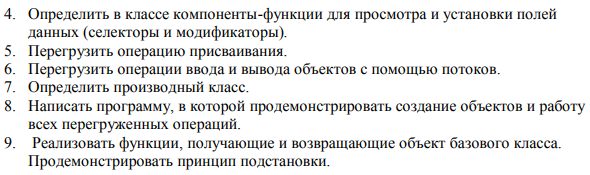
Проверила: доцент кафедры ИТАС

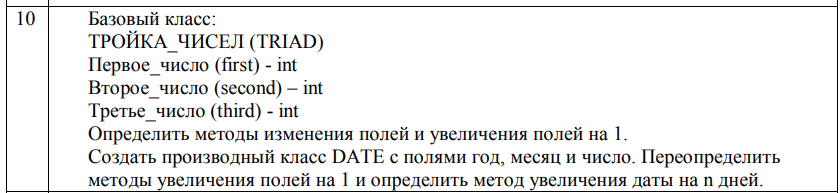
Полякова О.А.

Пермь, 2021

**Постановка задачи**







**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать класс Triad для хранения первого числа (first), второго числа (second),третьего числа (third)

**1.2.** Организовать класс Date с полями типа int: day, month,year, который является производным класса Triad.

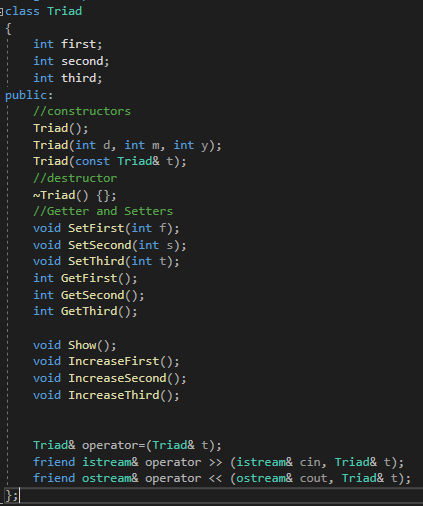
**1.3.** Организовать необходимые методы для ввода данных в поля first, second, third: гетторы, сетторы, конструкторы, деструктор.

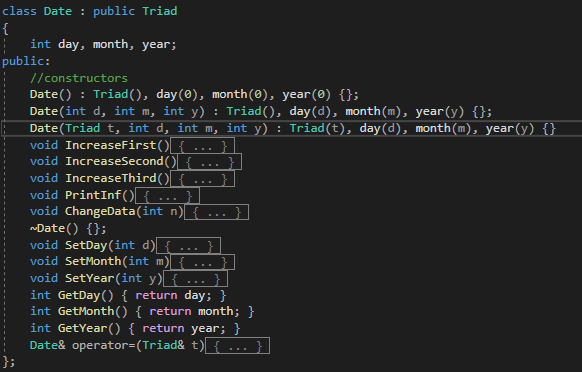
**1.4.** Организовать перегрузку оператора >> дружественной классу Triad.

**1.5.** Организовать перегрузку оператора << дружественной классу Triad.

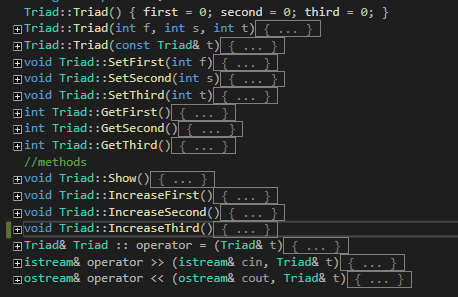
**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Организовать класс Triad для хранения первого числа (first), второго числа (second), третьего числа (third). в заголовочном файле Triad.h.

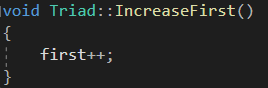
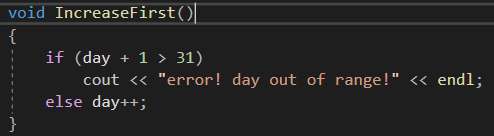


****

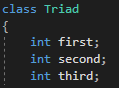
**2.2.** Организовать соответствующий файл Triad.cpp, в котором будут описаны методы, конструкторы и дружественные функции класса Triad.



**2.3.** Для реализация также необходимо описать методы увеличения полей на 1, в классе Date будет выполняться проверка, чтобы день, месяц и год не могли выходить за рамки.

**2.4.** Переменные типа int: first, second, third, day, month, year, где first, second, third – объекты класса Triad, day, month, year - объекты класса Date.

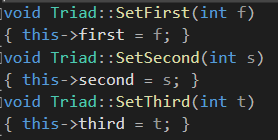
 

**2.5.** Объект A класса Triad. Объект D принадлежит классу Date.

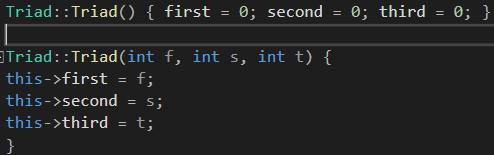
 ****

**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Данные вводятся через объекты класса Triad через cетторы (и присваивается значение, находящееся в скобках).

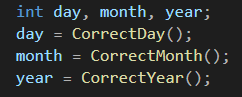


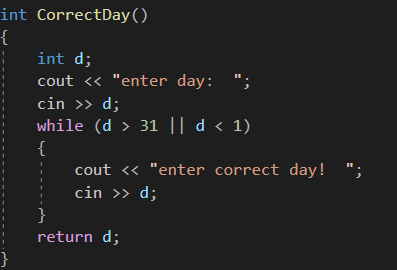
**3.2.** Данные также вводятся через объекты класса Triad через конструктор с параметрами или без.



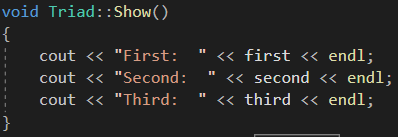
**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Ввод чисел происходит в главной функции, где (если это необходимо, выполняется проверка на ввод (день/месяц/год)).

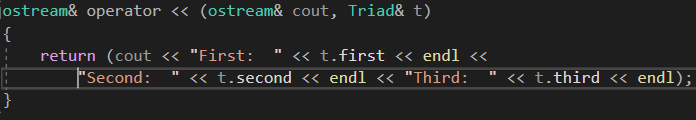




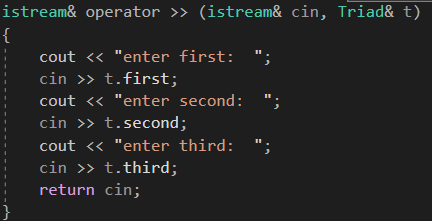
**4.2.** Для вывода на консоль значений переменных first, second, third используется метод Show.



**4.3.** Так же вывод на консоль может быть осуществлён с помощью перегрузки оператора <<.

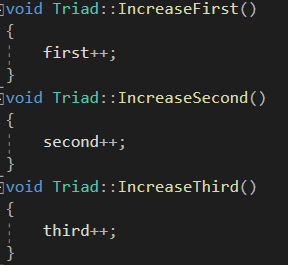


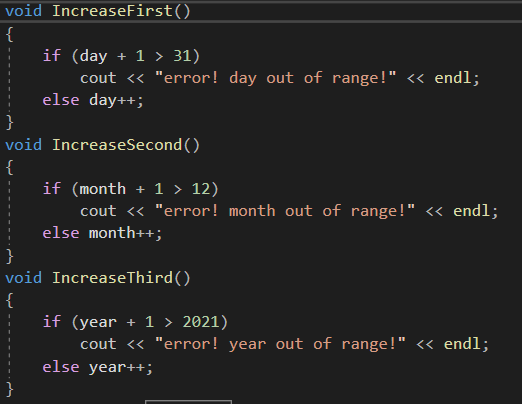
**4.4.** Так же ввод пар чисел может быть осуществлён с помощью перегрузки оператора >>.



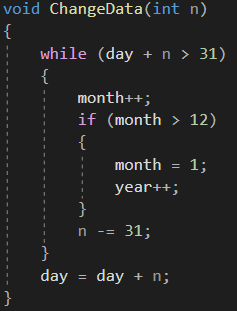
**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** В методах IncreaseFirst, IncreaseSecond , IncreaseThird описываются правила увеличения числа на 1(для класса Triad никаких ограничений нет, а для Date – выполняется проверка (см. п. 4.1.).

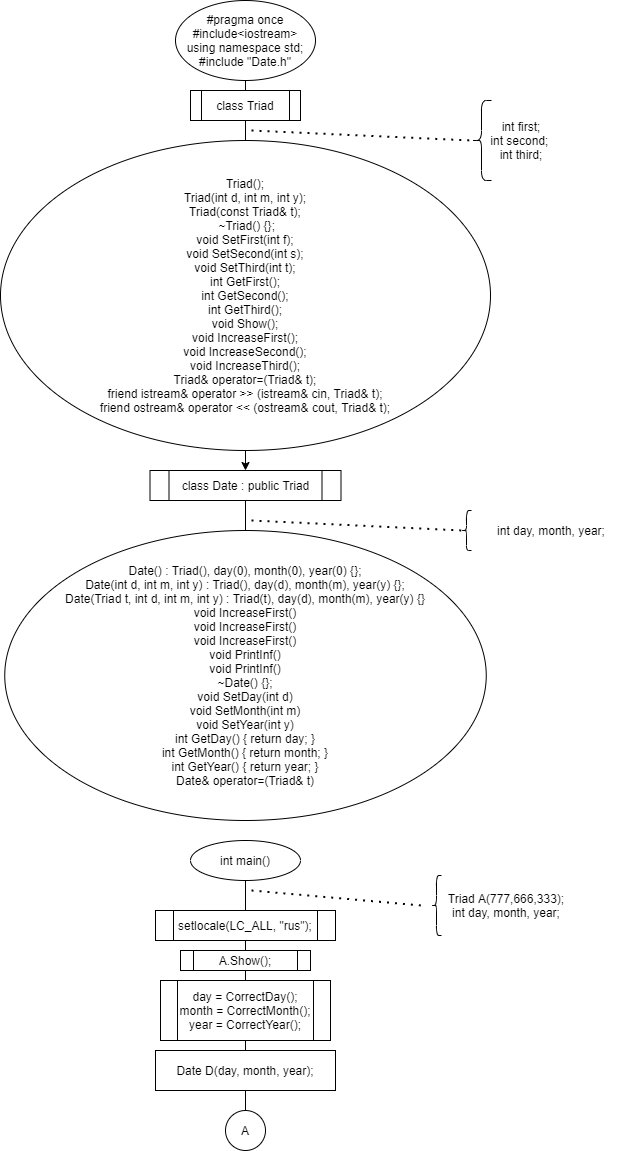


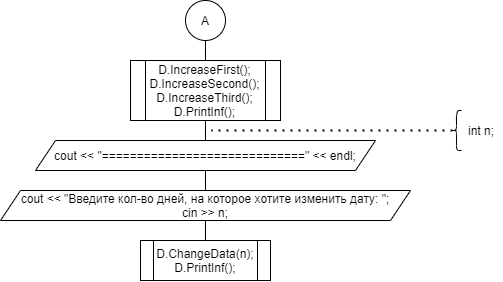


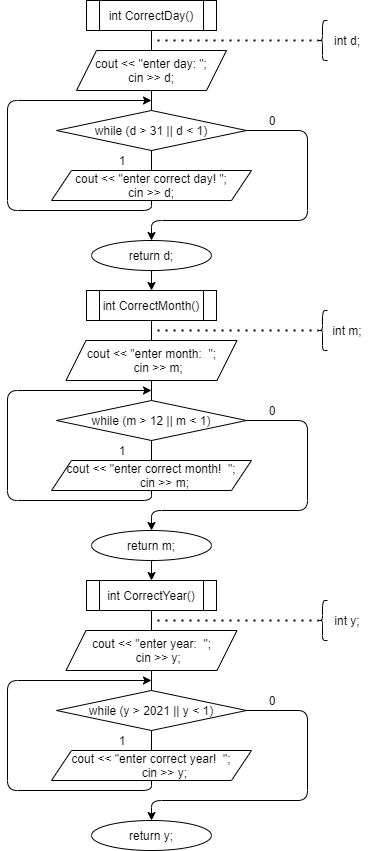
**5.2.** Также для выполнения задачи необходимо описать метод ChangeData, который позволяет изменять дату на n-дней.

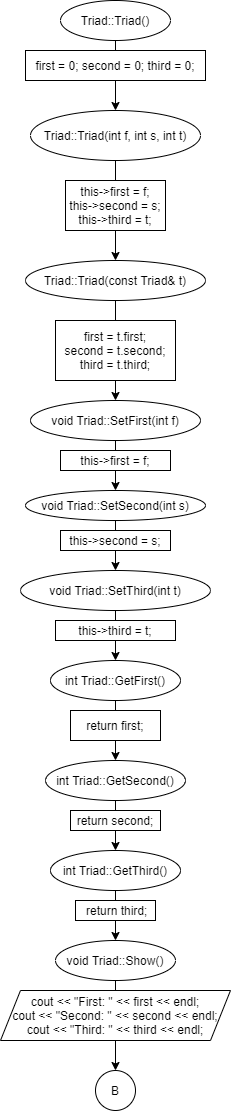


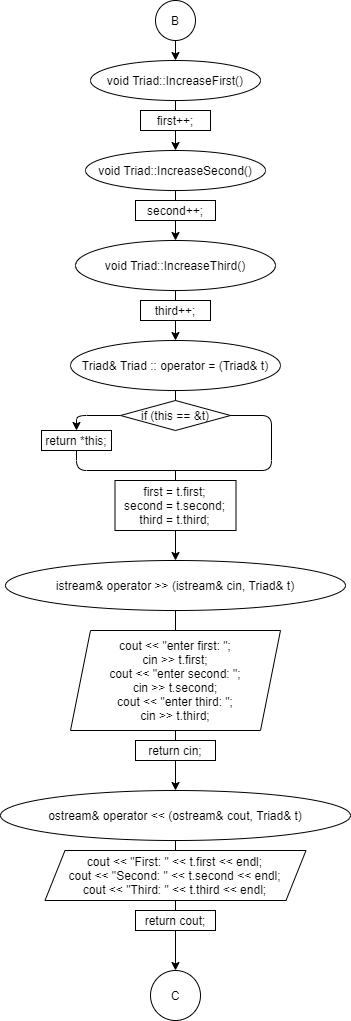
**Блок-схема**

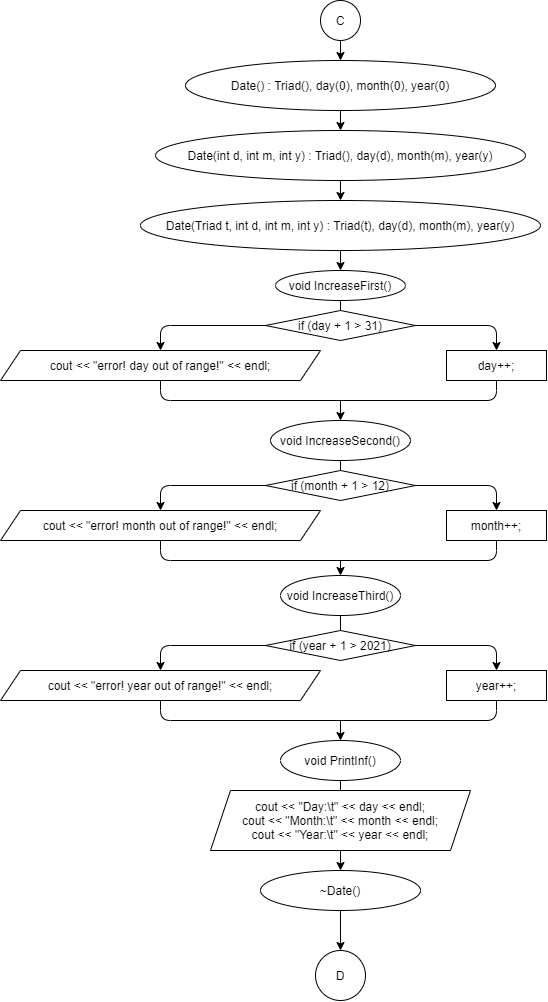


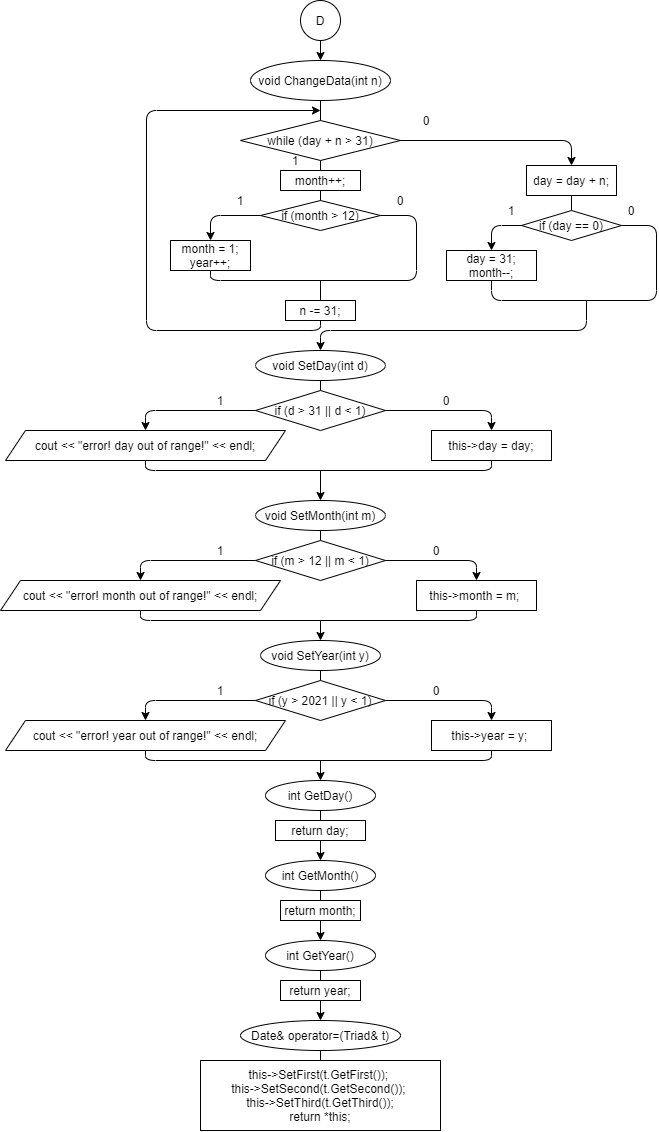












**Код**

#pragma once

#include "Data.h"

#include <iostream>

using namespace std;

int CorrectDay()

{

int d;

cout << "enter day: ";

cin >> d;

while (d > 31 || d < 1)

{

cout << "enter correct day! ";

cin >> d;

}

return d;

}

int CorrectMonth()

{

int m;

cout << "enter month: ";

cin >> m;

while (m > 12 || m < 1)

{

cout << "enter correct month! ";

cin >> m;

}

return m;

}

int CorrectYear()

{

int y;

cout << "enter year: ";

cin >> y;

while (y > 2021 || y < 1)

{

cout << "enter correct year! ";

cin >> y;

}

return y;

}

int main()

{

setlocale(LC\_ALL, "rus");

Triad A(777,666,333);

A.Show();

//entering date

int day, month, year;

day = CorrectDay();

month = CorrectMonth();

year = CorrectYear();

//print class Date

Date D(day, month, year);

D.IncreaseFirst();

D.IncreaseSecond();

D.IncreaseThird();

D.PrintInf();

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

//change data

int n;

cout << "Введите кол-во дней, на которое хотите изменить дату: ";

cin >> n;

D.ChangeData(n);

D.PrintInf();

}

#pragma once

#include <iostream>

using namespace std;

class Triad

{

int first;

int second;

int third;

public:

//constructors

Triad();

Triad(int d, int m, int y);

Triad(const Triad& t);

//destructor

~Triad() {};

//Getter and Setters

void SetFirst(int f);

void SetSecond(int s);

void SetThird(int t);

int GetFirst();

int GetSecond();

int GetThird();

void Show();

void IncreaseFirst();

void IncreaseSecond();

void IncreaseThird();

Triad& operator=(Triad& t);

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

friend ostream& operator << (ostream& cout, Triad& t);

};

#include "Data.h"

#include "Triad.h"

#include <iostream>

using namespace std;

Triad::Triad()

{ first = 0; second = 0; third = 0; }

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

this->first = f;

this->second = s;

this->third = t;

}

Triad::Triad(const Triad& t)

{

first = t.first;

second = t.second;

third = t.third;

}

void Triad::SetFirst(int f)

{ this->first = f; }

void Triad::SetSecond(int s)

{ this->second = s; }

void Triad::SetThird(int t)

{ this->third = t; }

int Triad::GetFirst()

{ return first; }

int Triad::GetSecond()

{ return second; }

int Triad::GetThird()

{ return third; }

//methods

void Triad::Show()

{

cout << "First: " << first << endl;

cout << "Second: " << second << endl;

cout << "Third: " << third << endl;

}

void Triad::IncreaseFirst()

{

first++;

}

void Triad::IncreaseSecond()

{

second++;

}

void Triad::IncreaseThird()

{

third++;

}

Triad& Triad :: operator = (Triad& t)

{

if (this == &t)

return \*this;

first = t.first;

second = t.second;

third = t.third;

}

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

{

cout << "enter first: ";

cin >> t.first;

cout << "enter second: ";

cin >> t.second;

cout << "enter third: ";

cin >> t.third;

return cin;

}

ostream& operator << (ostream& cout, Triad& t)

{

cout << "First: " << t.first << endl;

cout << "Second: " << t.second << endl;

cout << "Third: " << t.third << endl;

return cout;

}

#pragma once

#include "Triad.h"

#include <iostream>

using namespace std;

class Date : public Triad

{

int day, month, year;

public:

//constructors

Date() : Triad(), day(0), month(0), year(0) {};

Date(int d, int m, int y) : Triad(), day(d), month(m), year(y) {};

Date(Triad t, int d, int m, int y) : Triad(t), day(d), month(m), year(y) {}

void IncreaseFirst()

{

if (day + 1 > 31)

cout << "error! day out of range!" << endl;

else day++;

}

void IncreaseSecond()

{

if (month + 1 > 12)

cout << "error! month out of range!" << endl;

else month++;

}

void IncreaseThird()

{

if (year + 1 > 2021)

cout << "error! year out of range!" << endl;

else year++;

}

void PrintInf()

{

cout << "Day:\t" << day << endl;

cout << "Month:\t" << month << endl;

cout << "Year:\t" << year << endl;

}

void ChangeData(int n)

{

while (day + n > 31)

{

month++;

if (month > 12)

{

month = 1;

year++;

}

n -= 31;

}

day = day + n;

if (day == 0)

{

day = 31;

month--;

}

}

~Date() {};

void SetDay(int d)

{

if (d > 31 || d < 1)

{

cout << "error! day out of range!" << endl;

}

else this->day = day;

};

void SetMonth(int m)

{

if (m > 12 || m < 1)

{

cout << "error! month out of range!" << endl;

}

this->month = m;

};

void SetYear(int y)

{

if (y > 2021 || y < 1)

{

cout << "error! year out of range!" << endl;

}

this->year = y;

};

int GetDay() { return day; }

int GetMonth() { return month; }

int GetYear() { return year; }

Date& operator=(Triad& t)

{

this->SetFirst(t.GetFirst());

this->SetSecond(t.GetSecond());

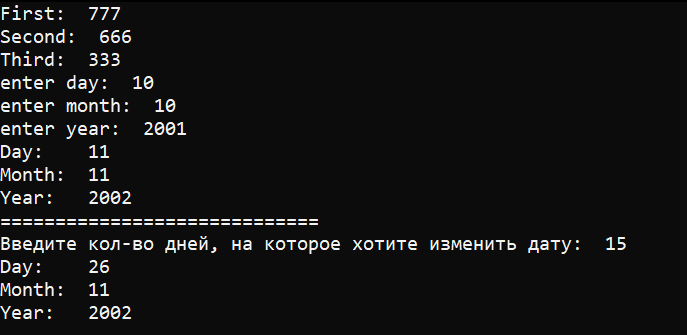
this->SetThird(t.GetThird());

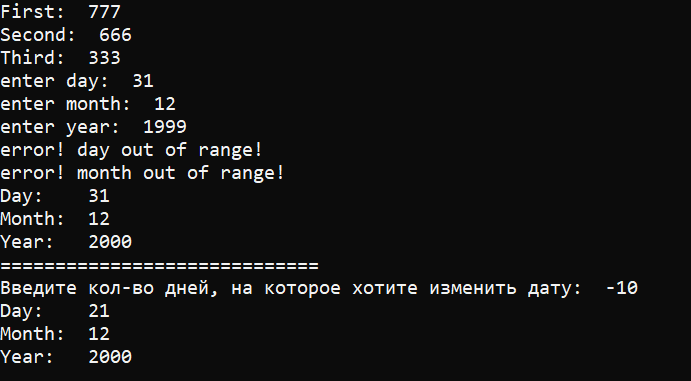
return \*this;

}

};

**Скриншоты**





**Контрольные вопросы**



1. Наследование позволяет реализовать типизированное повторное использование объектов, создание иерархии «тип-подтип» с сохранением подтипом всех свойств своего предка. Можно рассматривать наследование для всех трех составляющих объекта.



2. Когда вы открыто наследуете родительский класс, то унаследованные public-члены остаются public.



3. Когда вы открыто наследуете родительский класс, то унаследованные private-члены остаются недоступными для дочернего класса.



4. К protected-членам родительского класса доступ открыт для членов дочернего класса.



5. 



6. Конструкторы не наследуются.



7. Деструкторы не наследуются.



8. 



9. 

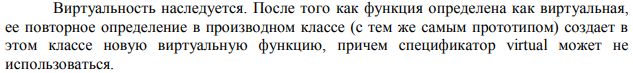


10. 

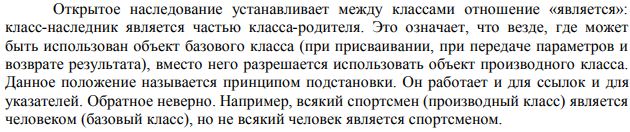


11. 



12. 



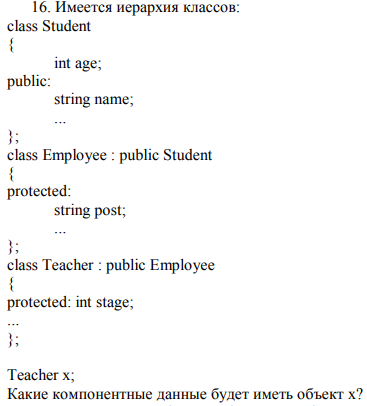
13. 



14. 



15. Пункт 13 и 14 – принцип подстановки.



16. age, name, post, stage.



17. Student(); Employee() : Student(); Teacher() : Employee();