Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное учреждение высшего образования  
«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т  
по лабораторной работе**

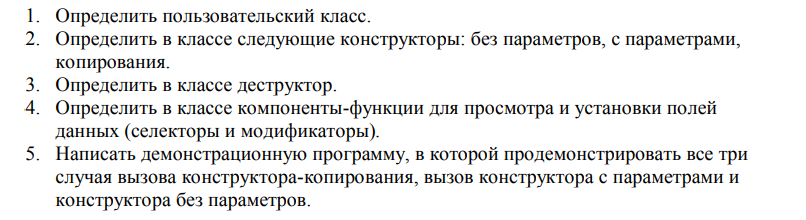
Дисциплина: «Основы алгоритмизации и программирования»

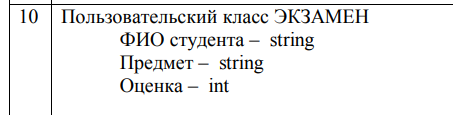
Выполнил  
студент группы ИВТ-21-1б  
Мифтахов И.Г

Проверил  
ст. п. кафедры ИТАС  
Яруллин Д.В.

Пермь 2022

Постановка задачи:



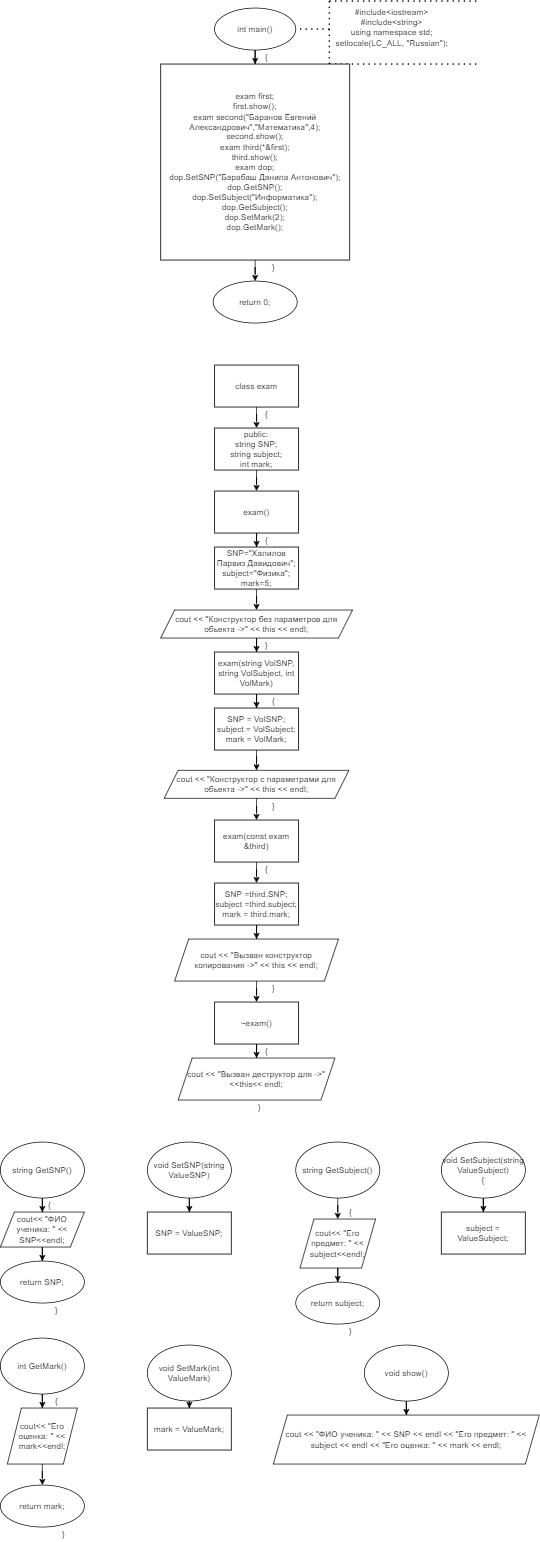


Анализ задачи:

1. Пользовательский класс exam содержит все поля и методы в области видимости public. Поля – SNP(string ФИО), subject(string предмет), mark(int Оценка). Описаны 3 конструктора – без параметров, с параметрами, конструктор копирования. Также описан деструктор.
2. Для каждого поля есть геттер GetNONE(), выводящий поле структуры на экран и возвращающий значение этого поля. Аналогично есть сеттеры SetNONE(PARAMETR), принимающий в себя значение для поля, внутри сеттера параметр присваивается полю. Аналогичную операцию проводят конструкторы. Без параметров – заполняет заранее написанными данными, с параметрами – заполняет переданными данными(аналогично сеттерам), копирования(копирует все поля для своего обьекта у конструктора без параметров(в моем случае). Также реализован метод show(), вызывающийся для каждого обьекта после конструкторов.
3. Ответы на вопросы:

* Конструктор необходим для инициализации полей класса.
* 3. Без параметров, с параметрами, конструктор копирования.
* Для завершения работы с конструктором класса после выхода из области видимости ф-ии. Если конструктор описан явно, то необходимо и деструктор описать явно.
* Без параметров – для инициализации полей по умолчанию(заранее заданными значениями). С параметрами – для инициализации полей переданными в конструктор значениями. Копирования – для инициализации полей с помощью копирования их из другого обьекта.
* Конструктор копирования вызывается в случае, когда нужно заполнить n обьектов одинаковыми значениями полей.
* Инициализация полей переданными параметрами, инициализация полей по умолчанию, инициализация полей с помощью копирования, перегрузка конструкторов, конструктор списка по умолчанию, конструктор списка инициализаторов.
* Не могут иметь аргументов; Не возвращают значение (или void ); Не может быть объявлен как const , volatile или static . Однако они могут вызываться для уничтожения объектов, объявленных как const , volatile или static ; Может быть объявлен как virtual . Используя виртуальные деструкторы, можно уничтожать объекты, не зная их тип — правильный деструктор для объекта вызывается с помощью механизма виртуальных функций. Обратите внимание, что для абстрактных классов деструкторы также могут объявляться как чисто виртуальные функции; не описаны явно – автоматически вызовутся.
* Private, Public, Protected;
* Адрес обьекта класса;
* Внутри класса можно не передать параметры, вне – обязательно. Использование шаблонов ф-ий порой обязательно.
* Конструкторы не возвращают значений.
* Конструктор и деструктор.
* Аналогично, деструктор не возвращает значений.
* Сеттер.
* Никакой.
* Конструктор с параметрами.
* Конструктор с параметрами для первого обьекта и конструктор копирования для второго.
* Аналогично предыдущему, только сначала второй обьект вызовет конструктор без параметров.
* С параметрами.
* p.set\_name(“name”);

Блок-схема:



Код программы:

#include<iostream>

#include<string>

using namespace std;

class exam

{

public:

string SNP;

string subject;

int mark;

exam()

{

SNP="Халилов Парвиз Давидович";

subject="Физика";

mark=5;

cout << "Конструктор без параметров для обьекта ->" << this << endl;

}

exam(string VolSNP, string VolSubject, int VolMark)

{

SNP = VolSNP;

subject = VolSubject;

mark = VolMark;

cout << "Конструктор с параметрами для обьекта ->" << this << endl;

}

exam(const exam &third)

{

SNP =third.SNP;

subject =third.subject;

mark = third.mark;

cout << "Вызван конструктор копирования ->" << this << endl;

}

~exam()

{

cout << "Вызван деструктор для ->" <<this<< endl;

}

string GetSNP()

{

cout<< "ФИО ученика: " << SNP<<endl;

return SNP;

}

void SetSNP(string ValueSNP)

{

SNP = ValueSNP;

}

string GetSubject()

{

cout<< "Его предмет: " << subject<<endl;

return subject;

}

void SetSubject(string ValueSubject)

{

subject = ValueSubject;

}

int GetMark()

{

cout<< "Его оценка: " << mark<<endl;

return mark;

}

void SetMark(int ValueMark)

{

mark = ValueMark;

}

void show()

{

cout << "ФИО ученика: " << SNP << endl << "Его предмет: " << subject << endl << "Его оценка: " << mark << endl;

}

};

int main()

{

setlocale(LC\_ALL, "Rus");

exam first;

first.show();

exam second("Баранов Евгений Александрович","Математика",4);

second.show();

exam third(\*&first);

third.show();

exam dop;

dop.SetSNP("Барабаш Данила Антонович");

dop.GetSNP();

dop.SetSubject("Информатика");

dop.GetSubject();

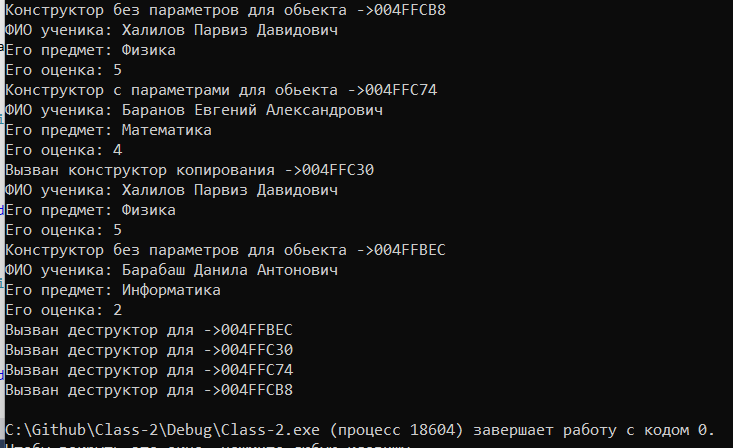
dop.SetMark(2);

dop.GetMark();

return 0;

}

Скриншоты результатов:



Анализ результатов:

Программа работает корректно. Выводятся по-разному инициализированные поля с указанием того, какой конструктор вызван. После – деструктор наглядно показывает завершение с конструкторами.