# ЛАБОРАТОРНАЯ РАБОТА №3

# ИССЛЕДОВАНИЕ МЕХАНИЗМА МНОЖЕСТВЕННОГО

# НАСЛЕДОВАНИЯ

Цель работы

Приобретение практических навыков при написании объектно-ориентированных программ с использованием механизма множественного наследования.

Задания

1. Изучить основы работы с механизмами множественного наследования;
2. Разработать программу с описанием интерфейса ввода-вывода и иерархии классов, заданной по варианту (Рисунок 1). Для каждого класса описать конструкторы и деструктор (по необходимости), функции ввода и вывода значений полей. В каждом классе должны присутствовать минимум одно уникальное поле и один уникальный метод. Проиллюстрировать корректную работу механизма множественного наследования — для этого создать объекты базовых классов и заполнить их поля данными, вывести на печать. Создать объект класса-наследника, его поля заполнить значениями соответствующих полей базовых классов. Вывести на печать данные полученного объекта;

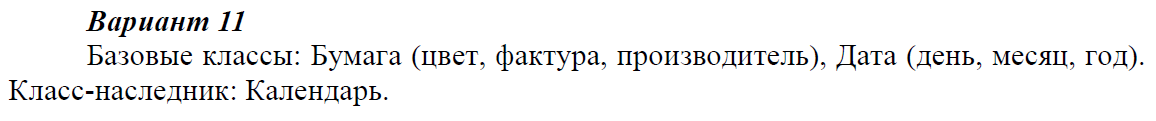


Рисунок 1 – Вариант задания

1. Разработать тестовые примеры и выполнить отладку программы;
2. Получить результаты работы программы и исследовать её свойства для различных режимов работы, сформулировать выводы;

Текст программы

#include <iostream>

#include <string>

using namespace std;

// Клас Бумага

class Paper

{

protected:

string color;

string facture;

string devlpr;

public:

Paper(string col = "", string fac = "", string dev = "") { // конструктор

color = col;

facture = fac;

devlpr = dev;

}

void setColor(string col) { color = col; } // методы ввода полей

void setFacture(string fac) { facture = fac; }

void setDeveloper(string dev) { devlpr = dev; }

string getColor() { return color; } // методы вывода полей

string getFacture() { return facture; }

string getDeveloper() { return devlpr; }

void print() {

cout << color << endl;

cout << facture << endl;

cout << devlpr << endl;

}

};

// Класс Дата

class Date

{

protected:

int day;

int month;

int year;

public:

Date(int d = 0, int m = 0, int y = 0) { // конструктор

day = d;

month = m;

year = y;

}

void setDay(int d) { day = d; } // методы ввода полей

void setMonth(int m) { month = m; }

void setYear(int y) { year = y; }

int getDay() { return day; } // методы вывода полей

int getMonth() { return month; }

int getYear() { return year; }

void print() {

cout << day << endl;

cout << month << endl;

cout << year << endl;

}

};

// Класс Календарь

class Calendar : public Paper, public Date

{

public:

// конструктор

Calendar(string p1 = "", string p2 = "", string p3 = "", int d1 = 0, int d2 = 0, int d3 = 0) : Paper(p1, p2, p3), Date(d1, d2, d3) {};

void Show() { // вывод всех полей

cout << color << endl;

cout << facture << endl;

cout << devlpr << endl;

cout << day << endl;

cout << month << endl;

cout << year << endl;

}

void print() {

Paper::print();

Date::print();

}

};

// Главная функция

int main()

{

// Заполнение объекта класса Paper

Paper p;

string s;

cout << "Color - ";

cin >> s;

p.setColor(s);

cout << "Facture - ";

cin >> s;

p.setFacture(s);

cout << "Developer - ";

cin >> s;

p.setDeveloper(s);

// Заполнение объекта класса Date

Date d;

int i;

cout << "Day - ";

cin >> i;

d.setDay(i);

cout << "Month - ";

cin >> i;

d.setMonth(i);

cout << "Year - ";

cin >> i;

d.setYear(i);

// Создание объекта класса - наследника Calendar, в качестве параметров переданы значения предыдущих объектов

Calendar c(p.getColor(), p.getFacture(), p.getDeveloper(), d.getDay(), d.getMonth(), d.getYear());

cout << endl;

c.Show();

cout << endl;

c.print();

}

Тестовые примеры

При запуске написанной программы был введён ряд требуемых значений, после чего все они были повторно выведены на экран (Рисунок 2).

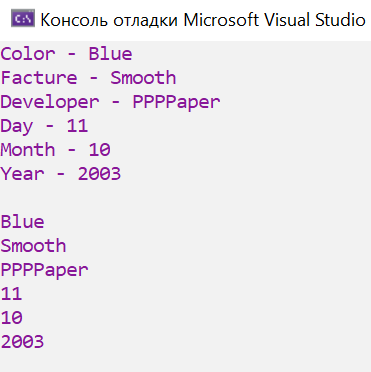


Рисунок 2 – Тестирование программы

Сначала программа предложила пользователю вручную заполнить поля объектов классов Paper и Date. Затем был создан объект класса Calendar, производного от первых двух классов. В нём были вызваны методы заполнения полей унаследованных классов, которым были переданы значения полей ранее созданных объектов.

Далее программа вызвала уникальный для класса Calendar метод Show, который вывел все его заполненные поля.

Вывод

В ходе работы был изучен принцип множественного наследования в объектно-ориентированном программировании. Была написана программа, содержащая класс, унаследованный от двух базовых, и обработку его объекта.