САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М.А. Бонч-Бруевича

ФАКУЛЬТЕТ ВЕЧЕРНЕГО И ЗАОЧНОГО ОБУЧЕНИЯ

Контрольная работа за _3_ семестр
По дисциплине *Технологии программирования*
Конструирование простейшего класса
Вариант _9_

Фамилия: ***Курчин***
Имя: ***Виталий***
Отчество: ***Владимирович***
Курс : ***2***
Студ. билет № : ***1510321***
Группа №: ***ОБ-56с***

Санкт-Петербург
2016

Задание.

В работе требуется сконструировать класс с заданным набором свойств. Набор свойств следует взять в соответствии со своим вариантом задания. В класс также должны быть добавлены методы для просмотра и изменения значений любого из свойств объекта.

Требования к конструированию класса: доступ к свойствам — закрытый, к методам — открытый. В классе следует предусмотреть конструктор по умолчанию, конструктор с параметрами.

Действия, выполняемые программой:

1. создание объекта с помощью конструктора по умолчанию,

2. создание объекта с помощью конструктора с параметрами,

3. создание массива объектов (размерность массива 3 или 4 элемента),

4. инициализация свойств каждого объекта массива(исходные данные вводятся с клавиатуры),

5. просмотр свойств каждого объекта,

6. вычисление заданного параметра для массива объектов в соответствии с вариантом задания (выполнить с помощью глобальной функции).

Требования к структуре программного кода: программа должна иметь модульную структуру, т.е. состоять из нескольких файлов: модуля класса, состоящего из заголовочного файла и файла реализации, и главного модуля, содержащего функцию main().

Марка принтера, Формат бумаги, Скорость печати, Цена. Определить самый дешевый принтер.

Реализация программы

Листинг файла printer.h

```
#ifndef Printer_H

#include "stdafx.h"
#include <memory.h>
#include <iostream>
#include <string.h>

#define MAX_STRING_LEN 255
using namespace std;

class Printer
{
protected:
	char* marka;
	char* format;
```

```
	char* speed;
	int money;

public:
	void init();
	void show();

	void setMarka(const char* _marka);
	void setFormat(const char*  _format);
	void setSpeed(const char*  _speed);
	void setMoney(int _money);

	const char* getMarka();
	const char* getFormat();
	const char* getSpeed();
	int getMoney();

	Printer();
	Printer(char* _marka, char* _format, char* _speed, int _money);

};

#endif
```

Листинг файла printer.cpp

```
#include "stdafx.h"
#include "printer.h"
#include <string.h>

void Printer::init()
{
	marka = new char[MAX_STRING_LEN];
	memset(marka, 0, MAX_STRING_LEN);
	format = new char[MAX_STRING_LEN];
	memset(format, 0, MAX_STRING_LEN);
	speed = new char[MAX_STRING_LEN];
	memset(speed, 0, MAX_STRING_LEN);
	money = 0;
}

Printer::Printer()
{
	init();
	cout << "Default Printer" << endl;
}


Printer::Printer(char* _marka, char* _format, char* _speed, int _money)
{
	init();
	setMarka(_marka);
	setFormat(_format);
	setSpeed(_speed);
	setMoney(_money);
	printf("Constructor (Printer) with params\n");
}

//setters
void Printer::setMarka(const char* _marka)
{
	if (_marka == NULL)
	{
		printf("_marka == NULL");
```

```
                return;
        }
        strcpy(marka, _marka);
}

void Printer::setFormat(const char* _format)
{
        if (_format == NULL)
        {
                printf("_format==NULL");
                return;
        }

        strcpy(format, _format);
}

void Printer::setSpeed(const char* _speed)
{
        if (_speed == NULL)
        {
                printf("_speed==NULL");
                return;
        }

        strcpy(speed, _speed);
}


void Printer::setMoney(int _money)
{
        if (_money == NULL)
        {
                printf("_money==NULL");
                return;
        }

        money = _money;
}


//getters
const char* Printer::getMarka()
{
        return marka;
}

const char* Printer::getFormat()
{
        return format;
}

const char* Printer::getSpeed()
{
        return speed;
}


int Printer::getMoney()
{
        return money;
}


//implement show method
void Printer::show()
```

```
{
	printf("marka = %s |  format = %s | speed = %s |money = %d\n", getMarka(),
getFormat(), getSpeed(), getMoney());
}
```

Листинг файла ConsoleApplication1.cpp

```
// ConsoleApplication1.cpp: определяет точку входа для консольного приложения.
//

#include "stdafx.h"
#include <memory.h>
# include <iostream>
#include <string.h>
#include "printer.h"


void low_cost(Printer** printers, int size){

	int cost = 9999999;
	Printer low_cost;
	for (int i = 0; i < size; i++)
	{
		if (cost>printers[i]->getMoney())
		{
			cost = printers[i]->getMoney();
			low_cost.setMarka(printers[i]->getMarka());
			low_cost.setFormat(printers[i]->getFormat());
			low_cost.setSpeed(printers[i]->getSpeed());
			low_cost.setMoney(printers[i]->getMoney());
		}

	}
	low_cost.show();
}


int _tmain(int argc, _TCHAR* argv[])
{
	//default constructor
	Printer printer1;
	printer1.setMarka("printer1");
	printer1.setFormat("A3");
	printer1.setSpeed("2 lists / min");
	printer1.setMoney(7800);
	printer1.show();

	//constructor with params
	Printer printer2("printer2", "A4", "11 lists / min", 12000);
	printer2.show();

	int size;
	cout << "Please enter the dimension of the array : ";
	cin >> size;


	Printer** printers = new Printer*[size];

	char* marka = new char[MAX_STRING_LEN];
	memset(marka, 0, MAX_STRING_LEN);
	char* format = new char[MAX_STRING_LEN];
	memset(format, 0, MAX_STRING_LEN);
	char* speed = new char[MAX_STRING_LEN];
	memset(speed, 0, MAX_STRING_LEN);
```

```
	int money = 0;

	for (int i = 0; i < size; i++)
	{
		printers[i] = new Printer();

		cout << "Please enter marka: ";
		cin >> marka;
		printers[i]->setMarka(marka);
		fflush(stdin);
		cout << "Please enter format: ";
		cin >> format;
		printers[i]->setFormat(format);
		fflush(stdin);
		cout << "Please enter speed: ";
		cin >> speed;
		printers[i]->setSpeed(speed);
		fflush(stdin);
		cout << "Please enter money: ";
		cin >> money;
		printers[i]->setMoney(money);
		fflush(stdin);

		cout << "*****************************\n";
	}

	for (int i = 0; i<size; i++)
		printers[i]->show();


	cout << "\nFind with low cost";
	low_cost(printers, size);

	system("pause");
}
```

Результат выполнения программы:

```
Default Printer
marka = printer1 !  format = A3 ! speed = 2 lists / min !money = 7800
Constructor (Printer) with params
marka = printer2 !  format = A4 ! speed = 11 lists / min !money = 12000
Please enter the dimension of the array : 2
Default Printer
Please enter marka: printer1
Please enter format: A4
Please enter speed: 12 lists / min
Please enter money: 11400
*****************************
Default Printer
Please enter marka: printer5
Please enter format: A1
Please enter speed: 1 list / min
Please enter money: 11720
*****************************
marka = printer1 !  format = A4 ! speed = 12 !money = 11400
marka = printer5 !  format = A1 ! speed = 1 !money = 11720

Find with low costDefault Printer
marka = printer1 !  format = A4 ! speed = 12 !money = 11400
Для продолжения нажмите любую клавишу . . .
```

```
protected:
	char* marka;
	char* format;
	char* speed;
	int money;
```

Методы описаны модификатором public:

```
public:
	void init();
	void show();

	void setMarka(const char* _marka);
	void setFormat(const char*  _format);
	void setSpeed(const char*  _speed);
	void setMoney(int _money);

	const char* getMarka();
	const char* getFormat();
	const char* getSpeed();
	int getMoney();
```

Класс Printer имеет два типа конструктора, как и описано в задании:

```
	Printer();
	Printer(char* _marka, char* _format, char* _speed, int _money);
```

А также метод вывода (show) и методы позволяющие обращаться к полям класса:

```
	void setMarka(const char* _marka);
	void setFormat(const char*  _format);
	void setSpeed(const char*  _speed);
	void setMoney(int _money);

	const char* getMarka();
	const char* getFormat();
	const char* getSpeed();
```

Создание объекта при помощи конструктора по умолчанию:

```
	//default constructor
	Printer printer1;
	printer1.setMarka("printer1");
	printer1.setFormat("A3");
	printer1.setSpeed("2 lists / min");
	printer1.setMoney(7800);
	printer1.show();
```

Создание объекта при помощи конструктора с параметрами:

```
	//constructor with params
	Printer printer2("printer2", "A4", "11 lists / min", 12000);
	printer2.show();
```

Создание динамического массива объектов:

```
	int size;
	cout << "Please enter the dimension of the array : ";
	cin >> size;

	Printer** printers = new Printer*[size];
```

Заполнение динамического массива объектов происходит в цикле for:

```
for (int i = 0; i < size; i++)
	{
		printers[i] = new Printer();

		cout << "Please enter marka: ";
		cin >> marka;
		printers[i]->setMarka(marka);
		fflush(stdin);
		cout << "Please enter format: ";
		cin >> format;
		printers[i]->setFormat(format);
		fflush(stdin);
		cout << "Please enter speed: ";
		cin >> speed;
		printers[i]->setSpeed(speed);
		fflush(stdin);
		cout << "Please enter money: ";
		cin >> money;
		printers[i]->setMoney(money);
		fflush(stdin);

		cout << "**************************\n";
	}
```

После заполнение динамического массива, происходит его вывод:

```
	for (int i = 0; i<size; i++)
		printers[i]->show();
```

И в конце происходит поиск принтера с самой низкой ценой:

```
int cost = 9999999;
	Printer low_cost;
	for (int i = 0; i < size; i++)
	{
		if (cost>printers[i]->getMoney())
		{
			cost = printers[i]->getMoney();
			low_cost.setMarka(printers[i]->getMarka());
			low_cost.setFormat(printers[i]->getFormat());
			low_cost.setSpeed(printers[i]->getSpeed());
			low_cost.setMoney(printers[i]->getMoney());
		}

	}
	low_cost.show();
```

Вопросы к контрольной работе.

1. Что такое класс в объектно-ориентированном программировании?

   Класс — это пользовательский тип данных, объединяющий данные и алгоритмы для обработки этих данных. Класс моделирует группу каких-либо реальных объектов (студенты, машины), процессов (путешествия), явлений (погода).

2. Какую структуру имеет модуль в C++?

Файл проекта имеет такую же структуру, как и файл модуля. Подобно файлу модуля, это файл исходного кода на языке C++, который компилируется с другими файлами при создании исполняемого файла.

В файле проекта имеется определенный набор ключевых элементов:

- Директива препроцессора #include <vcl\vcl.h> предназначена для включения в текст проекта заголовочного файла, ссылающегося на описания классов библиотеки компонентов.
- Директива препроцессора #pragma hrdstop предназначена для ограничения списка заголовочных файлов, доступных для предварительной компиляции.
- Директива USEFORM сообщает, какие модули и формы используются в проекте.
- директива USERES компилятора присоединяет файлы ресурсов к выполняемому файлу. При создании проекта автоматически создается файл ресурсов с расширением *.res для хранения курсоров, пиктограммы приложения и др.
- Application->Initialize() Это утверждение критично только в случае, если приложение является OLE automation-сервером. В остальных случаях оно фактически ничего не делает.
- Application->CreateForm() Это утверждение создает форму приложения. По умолчанию, каждая форма в приложении имеет свое утверждение CreateForm.
- Application->Run() Это утверждение запускает приложение (точнее, переводит его в состояние ожидания наступления одного из событий, на которое оно должно реагировать).
- Конструкция try...catch используется для корректного завершения приложения в случае возникновения ошибки при инициализации, создании форм, запуске приложения.

3. Какими средствами осуществляется консольный ввод данных в языке Си, C++?

В стандартном C++ существует два основных пути ввода-вывода информации: с помощью потоков, реализованных в STL (Standard Template Library) и посредством традиционной системы ввода-вывода, унаследованной от C.

4. Какие свойства (принципы) объектно-ориентированного программирования вы знаете?

В центре ООП находится понятие объекта. Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы. Сокрытие данных называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм подтипов — возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.