**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Факультет «Кибербезопасности»

Кафедра «Защищенных сетей связи»

|  |  |
| --- | --- |
| Направление подготовки: | 10.05.07 – Противодействие техническим разведкам |
| Кафедра: | ЗСС |

**ЛАБОРАТОРНАЯ РАБОТА № 6**

по дисциплине:

|  |
| --- |
| **Технология программирование** |

на тему:

|  |
| --- |
| **Конструирование классов на основе принципа наследования** |

Вариант № 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Выполнил студент группы | | ИБТС-31 |
|  |  | Костандов Максим Кириллович | | |
|  |  | *Фамилия И. О.* | | |
|  |  | Руководитель |  | |
| *оценка* |  |  | *уч. степень, уч. звание* | |
|  |  |  | | |
| *дата, подпись* |  | *Фамилия И. О.* | | |

**Цель работы:** изучить механизм открытого (public) наследования в C++, познакомиться с понятием «виртуальная функция», освоить технологию конструирования и способы документирования программы, включающей в себя класс-наследники, изучить возможности инструментальных сред разработки по автоматической генерации кода.

**Постановка задачи**

1. Разработка базового класса

Создать базовый класс, который будет содержать:

Два свойства (например, атрибуты объекта).

Два метода, один из которых должен быть виртуальной функцией. Виртуальная функция должна быть переопределена в производных классах.

2. Разработка классов-наследников

Создать хотя бы два производных класса, которые будут наследовать свойства и методы от базового класса:

Наследование хотя бы одного свойства от базового класса.

Добавление хотя бы одного собственного свойства.

В классы-наследники должны быть включены следующие методы:

Метод, унаследованный от базового класса.

Виртуальная функция базового класса, переопределённая в производном классе.

Собственные методы производного класса, включая хотя бы один метод, изменяющий состояние (свойство) объекта.

3. Разработка программы для управления объектами

Программа должна выполнять следующие действия:

Создание объектов на основе классов-наследников.

Объединение объектов в массив, представляющий собой массив указателей на базовый класс.

Отображение значений свойств объектов на экране в цикле (например, через метод вывода информации).

Изменение свойств объектов по номеру элемента массива.

Вычисление заданного параметра с использованием свойств объектов.

Выход из программы.

4. Структура проекта

Каждый класс должен быть реализован в отдельном модуле (файле).

5. Меню программы

Все действия из группы 2 и 3 должны быть доступны через меню, которое:

Позволяет выбрать произвольный порядок выполнения действий.

Работает в цикле, пока пользователь не выберет выход из программы.  
  
  
**Задание**  
  
1. В соответствии с вариантом задания разработать базовый класс. В базовый класс следует включить свойства и методы, общие для заданных классов-наследников. Базовый класс должен включать в себя не менее двух свойств и двух методов, один из которых – виртуальная функция.

2. Разработать классы, производные от базового класса (наследники). Классы-наследники должны наследовать от базового класса хотя бы одно свойство, а также должны иметь хотя бы одно собственное свойство. В классы-наследники должны быть включены следующие методы:

a. Метод, наследуемый от базового класса.

b. Виртуальная функция базового класса, переопределённая в производном классе.

c. Собственные методы производного класса. В состав производного класса должен быть включён хотя бы один метод, изменяющий какое-либо свойство класса.

3. Разработать программу, выполняющую следующие действия:

a. Создание нескольких объектов на основе классов–наследников.

b. Объединение объектов в массив (массив указателей на базовый класс).

c. Отображение значений свойств объектов на экране в цикле.

d. Изменение свойств объектов по номеру элемента массива.

e. Вычисление заданного параметра.

f. Выход из программы.

4. Объявление и реализацию каждого класса поместить в отдельном модуле.

5. Действия из группы 2. и 3. должны быть доступны через меню; последовательность выполнения действий – произвольная, в цикле.

**Ход работы**

# Разработка алгоритма

1. **Определение структуры**:
   * Разработать классы с нужными свойствами и методами.
2. **Ввод данных**:
   * Написать функцию для ввода данных в объекты.
3. **Обработка массива**:
   * Объединить объекты в массив и изменить их свойства.
4. **Вывод данных**:
   * Реализовать функцию для вывода значений свойств объектов.
5. **Создание проекта**:
   * Создать новый проект и организовать код в модули.
6. **Реализация функций**:
   * Написать функции для ввода, обработки и вывода.
7. **Проверка и тестирование**:
   * Проверить корректность ввода и протестировать программу.

**Результаты работы программы**

|  |  |
| --- | --- |
|  | **Меню** |
|  | **Добавление принтера** |
|  | **Добавление монитора** |
|  | **Список девайсов** |
|  | **Изменение параметра устройства** |

# Программа на языке С++

Main.cpp

#include <iostream>

#include "PeripheralDevice.h"

#include "Printer.h"

#include "Monitor.h"

using namespace std;

int main() {

PeripheralDevice\* devices[10];

int deviceCount = 0;

int choice;

while (true) {

cout << "\nMenu: \n";

cout << "1. Add Printer\n";

cout << "2. Add Monitor\n";

cout << "3. Display Devices\n";

cout << "4. Modify Device\n";

cout << "5. Calculate Minimum Price\n";

cout << "6. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

if (choice == 1) {

string name;

double price;

int speed;

cout << "Enter printer name: ";

cin >> name;

cout << "Enter printer price: ";

cin >> price;

cout << "Enter print speed: ";

cin >> speed;

devices[deviceCount++] = new Printer(name, price, speed);

} else if (choice == 2) {

string name;

double price;

double size;

cout << "Enter monitor name: ";

cin >> name;

cout << "Enter monitor price: ";

cin >> price;

cout << "Enter screen size: ";

cin >> size;

devices[deviceCount++] = new Monitor(name, price, size);

} else if (choice == 3) {

for (int i = 0; i < deviceCount; i++) {

devices[i]->display();

}

} else if (choice == 4) {

int index;

cout << "Enter device number to modify (0 to " << deviceCount - 1 << "): ";

cin >> index;

if (index < deviceCount) {

int subChoice;

cout << "1. Modify Printer Speed\n";

cout << "2. Modify Monitor Size\n";

cout << "Enter your choice: ";

cin >> subChoice;

if (subChoice == 1) {

int speed;

cout << "Enter new print speed: ";

cin >> speed;

static\_cast<Printer\*>(devices[index])->setPrintSpeed(speed);

} else if (subChoice == 2) {

double size;

cout << "Enter new screen size: ";

cin >> size;

static\_cast<Monitor\*>(devices[index])->setScreenSize(size);

}

} else {

cout << "Invalid device number!\n";

}

} else if (choice == 5) {

double minPrice = 999999;

for (int i = 0; i < deviceCount; i++) {

double price = devices[i]->calculateMinPrice();

if (price < minPrice) {

minPrice = price;

}

}

cout << "Minimum price: " << minPrice << endl;

} else if (choice == 6) {

cout << "Exiting program...\n";

break;

} else {

cout << "Invalid choice, try again.\n";

}

}

for (int i = 0; i < deviceCount; i++) {

delete devices[i];

}

return 0;

}

Monitor.cpp

#include "Monitor.h"

Monitor.h

#ifndef MONITOR\_H

#define MONITOR\_H

#include "PeripheralDevice.h"

class Monitor : public PeripheralDevice {

private:

double screenSize;

public:

Monitor(string n = "", double p = 0, double size = 0) : PeripheralDevice(n, p), screenSize(size) {}

void display() override {

PeripheralDevice::display();

cout << "Screen Size: " << screenSize << " inches" << endl;

}

double calculateMinPrice() override {

return price - (screenSize \* 2);

}

void setScreenSize(double size) {

screenSize = size;

}

};

#endif

PeripheralDevice.cpp

#include "PeripheralDevice.h"

PeripheralDevice.h

#ifndef PERIPHERALDEVICE\_H

#define PERIPHERALDEVICE\_H

#include <iostream>

#include <string>

using namespace std;

class PeripheralDevice {

protected:

string name;

double price;

public:

PeripheralDevice(string n = "", double p = 0) : name(n), price(p) {}

virtual void display() {

cout << "Name: " << name << ", Price: " << price << endl;

}

virtual double calculateMinPrice() = 0;

virtual ~PeripheralDevice() {}

};

#endif

Printer.cpp

#include "Printer.h"

**Printer.h**

#ifndef PRINTER\_H

#define PRINTER\_H

#include "PeripheralDevice.h"

class Printer : public PeripheralDevice {

private:

int printSpeed

public:

Printer(string n = "", double p = 0, int speed = 0) : PeripheralDevice(n, p), printSpeed(speed) {}

void display() override {

PeripheralDevice::display();

cout << "Print Speed: " << printSpeed << " pages per minute" << endl;

}

double calculateMinPrice() override {

return price - (printSpeed \* 0.1);

}

void setPrintSpeed(int speed) {

printSpeed = speed;

}

};

#endif

**Вывод:**

Алгоритм разработки программы включает в себя создание структуры данных, функций для ввода, обработки и вывода информации. Важно организовать код в модули, что обеспечит гибкость и удобство тестирования. После реализации программы необходимо провести проверку её работы с разными наборами данных, чтобы убедиться в корректности выполнения всех операций.