**Отчет по лабораторной работе №4**

**“ Конструирование классов на основе принципа наследования”**

1. **Постановка задачи**

В соответствии с вариантом задания разработать базовый класс. В базовый класс следует включить свойства и методы, общие для заданных классов­наследников. Базовый класс должен включать в себя не менее двух свойств и двух методов, один из которых – виртуальная функция.

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

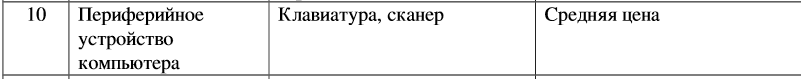
* 1. Метод, наследуемый от базового класса без переопределения.
  2. Виртуальная функция базового класса, переопределённая в производном классе.
  3. Собственные методы производного класса. В состав производного класса должен быть включён хотя бы один метод, изменяющий какое­либо свойство класса.

Разработать программу, выполняющую следующие действия:

* 1. Создание нескольких объектов на основе классов–наследников.
  2. Объединение объектов в массив (массив указателей на базовый класс).
  3. Отображение значений свойств объектов на экране в цикле.
  4. Изменение свойств объектов по номеру элемента массива.
  5. Вычисление заданного параметра.
  6. Выход из программы.

Объявление и реализацию каждого класса поместить в отдельный модуль.

Действия над объектами (просмотр, изменение, вычисление параметра) должны быть доступны через меню; последовательность выполнения действий – произвольная, в цикле.



1. **Структуры, использованные в программе**

## Наследование

Наследование – это механизм, посредством которого на базе существующих классов создаются новые классы, получающие по наследству от базовых классов часть их свойств и методов. Наследование позволяет существенно экономить программный код.

Созданные с использованием механизма наследования новые классы называются классынаследники или классы­потомки. Различают открытое, защищенное и закрытое наследование. Выбор варианта наследования Выбор варианта наследования влияет на доступность унаследованных членов класса в классе­наследнике. В данной работе рассматривается только открытое наследование.

При открытом наследовании новые классы наследуют свойства и методы, объявленные в разделе public и protected базовых классов (рисунок 1). При этом свойства и методы из раздела protected базового класса доступны только базовому классу и его наследникам, тогда как свойства и методы из раздела public доступны объектам любого класса. Исключением из этого правила являются конструкторы и деструкторы, которые не наследуются. В классе­наследнике наследуемые свойства получают уровень доступа в соответствии с доступностью в базовом классе: свойства и методы базового класса из раздела public наследуются с уровнем доступа public, а свойства и методы из раздела protected наследуются с уровнем доступа protected.

## Виртуальная функция

Виртуальные функции — это функция, объявленная в базовом классе с помощью ключевого слова virtual, такая функция в классах­потомках замещается на функцию, принадлежащую производному классу и имеющую то же имя.

1. **Описание программы**

Программа разработана на языке C++ в среде Microsoft Visual Studio 19 для OC Windows, выполняет обработку массива данных, в котором случайным образом задается информация о количестве и наличии товаров в магазине. Реализованы следующие методы:

- Вывод характеристик товара(show)

- Изменение характеристик товара (можно изменить абсолютно все характеристики) (SetManufacturer, SetPrice,SetColor,ChangeInStock,SetNumbOfKey,SetResolution)

- Вывод средней цены товара

- Наличие товара в магазине (GetInStock)

Имеется родительский класс perepherial и 2 дочерних класса Keyboard и scanner. Большая часть методов и переменных реализована в родительском классе, но у дочерних так же есть свои методы и переменные. Для вывода информации у каждого дочернего класса есть перегруженный метод show наследованный от родительского класса.

В родительском классе метод show является виртуальным методом

**Методы родительского класса perepherial**:

* SetColor – изменить цвет товара
* GetPrice – вывести среднюю цену всех товаров
* SetPrice – установить цену товара
* GetInStock – наличие/отсутствие товара в магазине
* ChangeInStock – изменить наличие/отсутствие товара в магазине
* Show – вывести все характеристики
* SetManufacturer – ввод нового производителя

**Методы дочернего класса Keyboard**:

* GetNumberOfKey – вывести количество клавиш на клавиатуре
* SetNumbOfKey – установить количество клавиш на клавиатуре

**Методы дочернего класса scanner**:

* GetResolution – вывести разрешение экрана
* SetResolution – установить разрешение у сканера

**Поля родительского класса perepherial:**

* char Color – цвет
* char Manufacturer - производитель
* float Price – цена товара
* bool inStock – наличие товара

**Поля родительского класса perepherial:**

* NumbOfKey=54 – количество клавиш

**Поля родительского класса scanner:**

* float Resolution = 600 - разрешение

Все методы родительского класса также присутствуют в дочерних классах

Для создания массива объектов из scanner и Keyboard используется vector со ссылкой на родительский класс. Когда появляется необходимость обратится к методам дочерних классов используется dynamic\_cast.

Взаимодействие с пользователем реализовано с помощью switch case и while

1. **Результаты работы программы**

На рисунках ниже представлена информация по вызову различных функций программы:

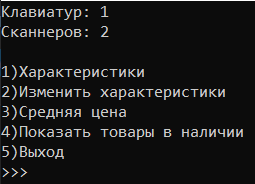


Рис.1 Стартовое меню

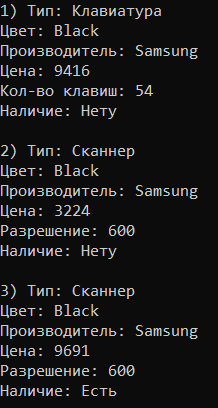


Рис.2 Пример работы функции “Характеристики”

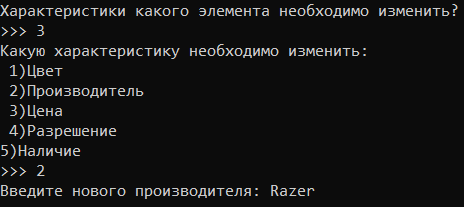


Рис. 3 Пример работы функции “Изменение характеристики”

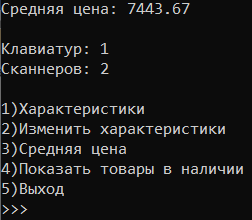


Рис. 4 Пример работы функции “Средняя цена”

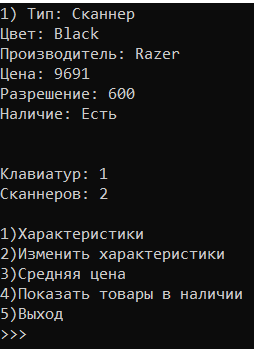


Рис. 5 Пример работы функции “Показать наличие”

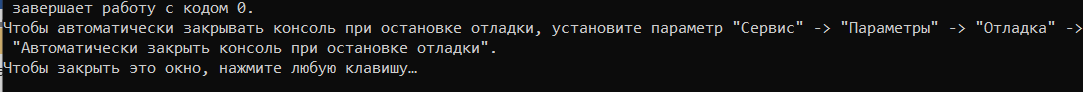


Рис. 6 Пример работы функции “Выход\”

1. **Выводы**

В ходе проделанной работы были получены следующие выводы:

1. Изучены методы конструирования классов
2. Освоена среда Visual Studio
3. Разработана программа, демонстрирующая работу базы данных магазина и склада, а также дающая возможность редактировать характеристики приборов
4. В коде реализованы два дочерних, и один родительский класс имеющие следующие методы:

**Методы родительского класса perepherial**:

* SetColor – изменить цвет товара
* GetPrice – вывести средюю цену всех товаров
* SetPrice – установить цену товара
* GetInStock – наличие/отсутствие товара в магазине
* ChangeInStock – изменить наличие/отсутствие товара в магазине
* Show – вывести все характеристики
* SetManufacturer – ввод нового производителя

**Методы дочернего класса Keyboard**:

* GetNumberOfKey – вывести количество клавиш на клавиатуре
* SetNumbOfKey – установить количество клавиш на клавиатуре

**Методы дочернего класса scanner**:

* GetResolution – вывести разрешение экрана
* SetResolution – установить разрешение у сканера

1. **Код программы**

#include <iostream>

#include <vector>

#include "Keyboard.h"

#include "scanner.h"

#include <memory>

#include <ctime>

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "rus");

bool Work = 1;

int NumbKey = rand() % 3 + 1;

int NumbScan = rand() % 3 + 1;

int NumbPer = NumbKey + NumbScan;

//peripheral per[NumbPer];

std::vector <std::unique\_ptr<peripheral>> per{};

for (int i = 0; i < NumbKey; i++)

{

per.emplace\_back(std::make\_unique<Keyboard>());

}

for (int i = NumbKey; i < NumbPer; i++)

{

per.emplace\_back(std::make\_unique<scanner>());

}

while (Work)

{

int choice;

std::cout << "Клавиатур: "<< NumbKey <<"\nСканнеров: "<< NumbScan <<"\n\n1)Характеристики\n2)Изменить характеристики\n3)Средняя цена\n4)Показать товары в наличии\n5)Выход\n>>> ";

std::cin >> choice;

switch (choice)

{

case 1:

{

system("cls");

for (int i = 0; i < NumbPer; i++)

{

const auto& item = per[i];

std::cout << i+1 << ") ";

item->show();

}

std::cout << std::endl;

break;

}

case 2:

{

system("cls");

int a;

std::cout << "Характеристики какого элемента необходимо изменить?\n>>> ";

std::cin >> a;

a--;

const auto& item = per[a];

auto key = dynamic\_cast<Keyboard\*>(item.get());

if (key)

{

int b;

std::cout << "Какую характеристику необходимо изменить:\n 1)Цвет\n 2)Производитель \n 3)Цена \n 4)Кол-во клавиш \n 5)Наличие\n>>> ";

std::cin >> b;

switch (b)

{

case 1:

{

char newColor[30];

std::cout << "Введите новый цвет: ";

std::cin >> newColor;

key->SetColor(newColor);

system("cls");

break;

}

case 2:

{

char newManufacturer[30];

std::cout << "Введите нового производителя: ";

std::cin >> newManufacturer;

key->SetManufacturer(newManufacturer);

system("cls");

break;

}

case 3:

{

float newPrice;

std::cout << "Введите новую цену: ";

std::cin >> newPrice;

key->SetPrice(newPrice);

system("cls");

break;

}

case 4:

{

int newNumbKey;

std::cout << "Введите новое кол-во клавиш: ";

std::cin >> newNumbKey;

key->SetNumbOfKey(newNumbKey);

system("cls");

break;

}

case 5:

{

key->ChangeInStock();

break;

}

default:

{

system("cls");

std::cout << "Ошибка ввода\n\n";

break;

}

}

}

else

{

int b;

std::cout << "Какую характеристику необходимо изменить:\n 1)Цвет\n 2)Производитель \n 3)Цена \n 4)Разрешение \n5)Наличие\n>>> ";

std::cin >> b;

auto scan = dynamic\_cast<scanner\*>(item.get());

switch (b)

{

case 1:

{

char newColor[30];

std::cout << "Введите новый цвет: ";

std::cin >> newColor;

scan->SetColor(newColor);

system("cls");

break;

}

case 2:

{

char newManufacturer[30];

std::cout << "Введите нового производителя: ";

std::cin >> newManufacturer;

scan->SetManufacturer(newManufacturer);

system("cls");

break;

}

case 3:

{

float newPrice;

std::cout << "Введите новую цену: ";

std::cin >> newPrice;

scan->SetPrice(newPrice);

system("cls");

break;

}

case 4:

{

int newResolution;

std::cout << "Введите новое разрешение: ";

std::cin >> newResolution;

scan->SetResolution(newResolution);

system("cls");

break;

}

case 5:

{

key->ChangeInStock();

break;

}

default:

{

system("cls");

std::cout << "Ошибка ввода\n\n";

break;

}

}

}

break;

}

case 3:

{

system("cls");

float TotalPrice = 0;

for (int i = 0; i < NumbPer; i++)

{

const auto& item = per[i];

TotalPrice += item->GetPrice();

}

TotalPrice /= float(NumbPer);

std::cout << "Средняя цена: " << TotalPrice << "\n\n";

break;

}

case 4:

{

system("cls");

int a = 0;

for (int i = 0; i < NumbPer; i++)

{

const auto& item = per[i];

if (item->GetInStock())

{

a++;

std::cout << a << ") ";

item->show();

}

}

std::cout << std::endl;

break;

}

case 5:

{

Work = 0;

system("cls");

break;

}

default:

{

system("cls");

std::cout << "Ошибка ввода\n";

}

}

}

}