Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Отчет по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Вариант 22

Выполнил:

Студент ФДО гр. №з-433П2-4

О.С. Яковлева

04.05.2024 г.

Проверил:

Старший

преподаватель кафедры АСУ

А. Е. Косова

2024 г.

Оглавление

1. ВВЕДЕНИЕ.

1.1.Цель работы

Целью выполнения настоящей лабораторной работы является получение начальных навыков работы с C++:

1) реализация и использование классов;

2) определение методов;

3) работа с объектами;

4) работа со стандартными потоками ввода-вывода.

1.2.Задание

Общее задание:

1. В среде программирования на C++ создайте консольный проект с именем LAB1 в каталоге LAB1.

2. В проекте создайте файлы main.h (заголовочный файл) и main.cpp (файл исходного кода).

3. В файле main.h определите с помощью ключевого слова class объект Person.

Данные объекта:

Номер человека (целый тип)

ФИО (символьный массив)

Пол (логический тип: 0-муж., 1-жен.)

Возраст (вещественный тип)

Пусть данные имеют закрытый уровень доступа (private).

4. Опишите конструктор объекта, аргументы которого будут инициализировать все данные объекта.

5. Опишите конструктор объекта по умолчанию (без аргументов), проинициализировав все данные.

6. Опишите в объекте функцию void Print() с открытым уровнем доступа (public), которая будет выводить данные на экран.

7. Откройте файл main.cpp. С помощью директивы #include включите в файл main.cpp заголовочные файлы <stdlib.h>, <string.h>, <iostream.h>, а также ваш заголовочный файл "Main.h".

8. Ниже определите конструктор объекта, инициализирующий все данные объекта значениями аргументов. В теле конструктора используйте функцию strcpy(стр1, стр2) для копирования строки имени человека (ФИО).

9. Затем определите функцию void Person::Print(). В теле функции для вывода данных используйте стандартный поток вывода cout << значение1 << значение2 << … << endl;

10. Ниже напишите главную функцию программы int main(). Внутри ее создайте объект Person, указав все значения данных объекта. Выведите данные объекта на экран, вызвав функцию Print.

11. Затем создайте динамический объект Person с помощью обычного конструктора и оператора new. Выведите данные объекта на экран. Удалите динамический объект из памяти с помощью оператора delete.

12. Напишите функцию ввода данных в объект с клавиатуры void Person::Input(). В теле функции для ввода данных используйте стандартный поток ввода cin >> значение1 >> значение2 >> … ;

13. Затем в теле функции main создайте объект Person с помощью конструктора по умолчанию и введите данные в объект с клавиатуры, вызвав функцию Input. Выведите данные объекта на экран.

Задание для варианта 22: class Pin:

int diameter;

char type[MAX_LENGTH];

float needle;

bool used.

2. ОСНОВНАЯ ЧАСТЬ

2.1. Теоретическая часть

Класс в C++ -- пользовательский тип данных, в котором хранятся данные (поля класса) и объявляются функции, оперирующие этими данными. Переменные класса называются объектами.

Класс объявляется ключевым словом class. Содержимое класса делится на три уровня доступа модификаторами private, public и protected. В приватной области описываются данные для использования внутри класса, что позволяет скрыть реализацию членов класса, таким образом, осуществляется один из принципов ООП – инкапсуляция. В публичной области объявляется конструктор для начальной инициализации переменной класса, в защищенной области содержится список средств, доступных для дружественных классов и наследников.

Для упрощения процесса инициализации элементов класса в C++ используются конструкторы представляют специальную функцию, которая имеет то же имя, что и класс, которая не возвращает никакого значения и которая позволяют инициалилизировать объект класса во время его создания и таким образом гарантировать, что поля класса будут иметь определенные значения.

При необходимости освобождения участка памяти используется деструктор – метод класса, используемый для удаления объектов класса.

C++ позволяет определять функции внутри определения класса, так и разделять объявление и определение функций, последнее используется во многих программах. Разделение объявления и определения функций помогает упростить понимание интерфейса класса.

В C++ можно использовать различные типы объектов, которые различаются по использованию памяти. Существуют глобальные объекты, которые создаются при запуске и уничтожаются при завершении программы, а также локальные объекты. Статические локальные объекты создаются перед первым своим использованием и уничтожаются при завершении работы программы, для глобальных и локальных статических объектов выделяется статическая память. Локальные автоматические объекты создаются в блоке

кода и удаляются при завершении работы блока. Эти объекты размещаются в стеке и удаляются в порядке, противоположном порядку создания.

Для большей гибкости C++ позволяет создавать динамические объекты, которые размещаются в динамической памяти. Такое размещение позволяет эффективно использовать память, так как выделяется необходимый кусок памяти, их жизненный цикл не зависит от блока, в котором используются объекты, а созданием (new) и удалением (delete) объекта управляет программист. Это, с одной стороны, позволяет использовать большие объемы памяти, но, с другой, накладывает определенное ограничение – необходимо не забывать об удалении объекта.

Инструменты для работы с системой ввода-вывода определены в стандартной библиотеке, подключаемой заголовочным файлом <iostream>. Существует 4 стандартных потока ввода-вывода: cin (ввод), cout (вывод), cerr (вывод ошибок), clog (полностью буферизованный вывод ошибок).

2.2.Описание и результаты выполненной работы

Проект был создан в Visual Studio 2022 (версия Microsoft Visual Studio Community 2022 (ARM 64-bit) – Current Version 17.10.1). Проект создан с помощью инструмента CMake с предопределенными параметрами для release и debug сборок. Сборка Release выполнена со следующими параметрами: Local Machine, x64 Release, output file LAB1.exe.

При запуске .exe файла Windows выдает предупреждение системы безопасности об отсутствии цифровой подписи. За неимением сертификата подписать исполняемый файл цифровой подписью не представляется возможным. При игнорировании сообщения запуск проекта на сторонних машинах выполняется без ошибок.

Проект был выполнен с соблюдением всех шагов, прописанных в задании. Результаты размещены в приложенном архиве, за исключением исполняемого файла .exe. Проект также размещен на Github по ссылке: https://github.com/yakovlevaos/LR1.

В выбранной среде был создан проект с именем LAB1.

1) Созданы 2 файла, в соответствии с заданием, файл main.cpp и файл main.h.

2) В соответствии с вариантом задания, определили с помощью ключевого слова class класс Pin:

Данные имеют уровень доступа private:

int diameter – диаметр кнопки

char type[MAX_LENGTH] – тип кнопки, символьный массив. MAX_LENGTH описана в константе, константа определена с публичным уровнем доступа (static const int MAX_LENGTH = 20;)

float needle – длина жала

bool used – новая кнопка или использованная.

Также описали сеттеры и геттеры:

int getDiameter() const { return diameter; } const char* getType() const { return type; } float getNeedle() const { return needle; } bool isUsed() const { return used; }

void setDiameter(int newDiameter) { diameter = newDiameter; }

void setType(const char* newType) { strncpy(type, newType, MAX_LENGTH - 1); type[MAX_LENGTH - 1] = '\0'; }

void setNeedle(float newNeedle) { needle = newNeedle; } void setUsed(bool newUsed) { used = newUsed; }

4) Описали конструктор объекта: explicit Pin(int diameter, const char *type, float needle, bool used);

6) Описали в объекте функции Print() и Input() для вывода/ввода данных.

void Print();

void Input();

7) Включили заголовочные файлы в файл main.cpp + директиву using namespace std:

#include<iostream>

#include<cstring>

#include "main.h"

using namespace std;

8) Определили конструктор по умолчанию и конструктор объекта, инициализирующего все данные объекта значениями аргументов. Согласно заданию, необходимо использовать функцию strcpy(стр1, стр2) для копирования строки. Поскольку strcpy не является безопасной функцией и может вызвать утечку памяти, было принято решение использовать более безопасную функцию strncpy для 1) предотвращения переполнения буфера (указали максимальное количество символов копирования), 2) гарантированного завершения строки нулевым символом, 3) безопасного копирования фиксированной длины. Также определили деструктор и описали константы.

const int MAX_TYPE_LENGTH = 19; // Максимальная длина строки для типа, равна 19, чтобы позволить безопасное копирование строки и корректное завершение строки нулевым символом.

const int MAX_LENGTH = 20; // Максимальная длина строки, может быть переопределена в классе Pin, задана на 20, чтобы дать пространство для ввода и нулевого символа.

const int MIN_DIAMETER = 0; // Минимально допустимое значение для диаметра

const float MIN_NEEDLE_SIZE = 0.0f; // Минимально допустимый размер иглы

Pin::Pin(){

diameter = MIN_DIAMETER; // Устанавливаем диаметр в 0

strncpy(type, "Default", MAX_TYPE_LENGTH); // Копируем строку "Default" в type, ограничивая длину

type[MAX_TYPE_LENGTH - 1] = '\0'; // Завершаем строку нулевым символом

needle = MIN_NEEDLE_SIZE; // Устанавливаем размер иглы в 0.0

used = false; // Устанавливаем флаг использования в false

}

// Определение конструктора класса Pin с параметрами

Pin::Pin(int diameter, const char *type, float needle, bool used){

this->diameter = diameter; // Устанавливаем диаметр

strncpy(this->type, type, MAX_TYPE_LENGTH - 1); // Копируем тип, ограничивая длину

this->type[MAX_TYPE_LENGTH - 1] = '\0'; // Завершаем строку нулевым символом

this->needle = needle; // Устанавливаем размер иглы

this->used = used; // Устанавливаем флаг использования

}

// Определение деструктора класса Pin

Pin::~Pin(){

cout << "Destructor called for Pin object" << endl; // Сообщение при вызове деструктора

}

9) Определили функцию Print() для вывода данных:

void Pin::Print() const{ // Вывод информации об объекте Pin с использованием геттеров для доступа к данным

cout << "Pin diameter: " << getDiameter()

<< " Type: " << getType()

<< " Needle size: " << getNeedle()

<< " Used: " << (isUsed() ? "true" : "false") << endl;

}

Дополнительно определили методы для установки диаметра, типа, размера иглы и использования:

// Метод для установки диаметра

bool Pin::setDiameter(int dia)

```
{
// Устанавливаем диаметр, если он неотрицательный
if (dia >= MIN_DIAMETER)
{
diameter = dia;
return true;
}
return false;
}

// Метод для установки типа
bool Pin::setType(const char *t)
{
// Проверяем, что длина строки меньше максимального значения, и копируем тип
if (strlen(t) < MAX_TYPE_LENGTH)
{
strncpy(type, t, MAX_TYPE_LENGTH - 1);
type[MAX_TYPE_LENGTH - 1] = '\0';
return true;
}
return false;
}
// Метод для установки размера иглы
bool Pin::setNeedle(float n)
{
// Устанавливаем размер иглы, если он неотрицательный
if (n >= MIN_NEEDLE_SIZE)
{
needle = n;
```

```
return true;
}
return false;
}

// Метод для установки использования
bool Pin::setUsed(bool u)
{
// Устанавливаем флаг использования
used = u;
return true;
}
```

Также определили функцию Input() для ввода данных. Внутри функции выполнили проверку правильности введенных пользователем данных циклом while для каждого типа.

```
void Pin::Input()
{
int tempDiameter;
float tempNeedle;
char tempType[MAX_LENGTH];
int tempUsed;
// Ввод диаметра
cout << "Enter Pin diameter: ";
while (!(cin >> tempDiameter) || !setDiameter(tempDiameter))
{
cout << "Invalid input. Please enter a valid non-negative integer for diameter: ";
cin.clear(); // Очистка потока ввода
cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Игнорирование неверного ввода
```

```
}
cin.ignore(); // Игнорирование символа новой строки

// Ввод типа

std::cout << "Enter type (max " << MAX_TYPE_LENGTH - 1 << " characters): ";

while (true) {

std::cin.getline(tempType, MAX_TYPE_LENGTH); // Чтение строки

// Проверка и очистка потока

if (std::cin.fail()) {

std::cin.clear(); // Очистить состояние потока

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Игнорировать остаточные символы новой строки

std::cout << "Invalid input. Please enter a type with less than " << MAX_TYPE_LENGTH - 1 << " characters: ";

continue;}

if (setType(tempType)) {

break; // Установка типа, если длина корректная}

else {

std::cout << "Invalid input. Please enter a type with less than " << MAX_TYPE_LENGTH << " characters: ";

}
```

```
}// Ввод размера иглы
cout << "Enter needle size: ";
while (!(cin >> tempNeedle) || !setNeedle(tempNeedle))
{
cout << "Invalid input. Please enter a valid floating-point number for needle size: ";
cin.clear(); // Очистка потока ввода
cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Игнорирование неверного ввода
}
// Ввод использования
cout << "Enter if the pin is used (1 for true, 0 for false): ";
while (!(cin >> tempUsed) || (tempUsed != 0 && tempUsed != 1))
{
cout << "Invalid input. Please enter either 1 or 0 for used: ";
cin.clear(); // Очистка потока ввода
cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Игнорирование неверного ввода
}
setUsed(tempUsed); // Установка флага использования
}
```

10) Написали основную функцию main, где создали объект Pin указав все значения данных объекта, затем вывели их на экран. После этого создали динамический объект Pin, вывели данные на экран, удалили динамический объект. Затем создали объект Pin с помощью конструктора по умолчанию и возможностью ввода данных с клавиатуры. Перед завершением программы спрашиваем пользователя, хочет ли он выйти. Если пользователь отказывается выходить, вызываем функцию main для повтора работы программы, если пользователь соглашается – выходим из программы. Перед выходом делаем паузу: пользователь должен нажать Enter.

```
int main()
{
// Статическое выделение памяти для объектов pin и pin2 с использованием конструктора по умолчанию и с параметрами
Pin pin; // Создание объекта pin с использованием конструктора по умолчанию
pin.Print(); // Вывод информации о pin
Pin pin2(2, "Drawing", 1.3, true); // Создание объекта pin2 с передачей параметров в конструктор
pin2.Print();  // Вывод информации о pin2
// Динамическое выделение памяти для объектов pin3 и pin4 с использованием конструктора по умолчанию и с параметрами
Pin *pin3 = new Pin(); // Создание объекта pin3 с использованием конструктора по умолчанию
pin3->Input(); // Ввод данных о pin3
pin3->Print(); // Вывод информации о pin3
delete pin3; // Освобождение памяти, занятой объектом pin3
Pin *pin4 = new Pin(3, "Map", 6.5, false); // Создание объекта pin4 с передачей параметров в конструктор
pin4->Print(); // Вывод информации о pin4
delete pin4; // Освобождение памяти, занятой объектом pin4
// Используем цикл для повторного выполнения программы вместо рекурсивного вызова main
char choice;
do
{
cout << "Do you want to exit? (y/n): ";
cin >> choice;
if (choice == 'n' || choice == 'N')
{
```

Pin *newPin = new Pin(); // Создание нового объекта Pin

newPin->Input(); // Ввод данных для нового объекта

newPin->Print(); // Вывод информации о новом объекте

delete newPin; // Освобождение памяти, занятой новым объектом}

} while (choice != 'y' && choice != 'Y'); // Продолжение цикла, пока не будет введен 'y' или 'Y'

cout << "Exiting program...";

cout << "\nPress Enter to exit...";

cin.ignore(); // Ожидание нажатия клавиши Enter

cin.get();// Удержание консоли до нажатия Enter

return 0; // Завершение программы}

Результаты выполнения программы приведены на Рисунке 1.

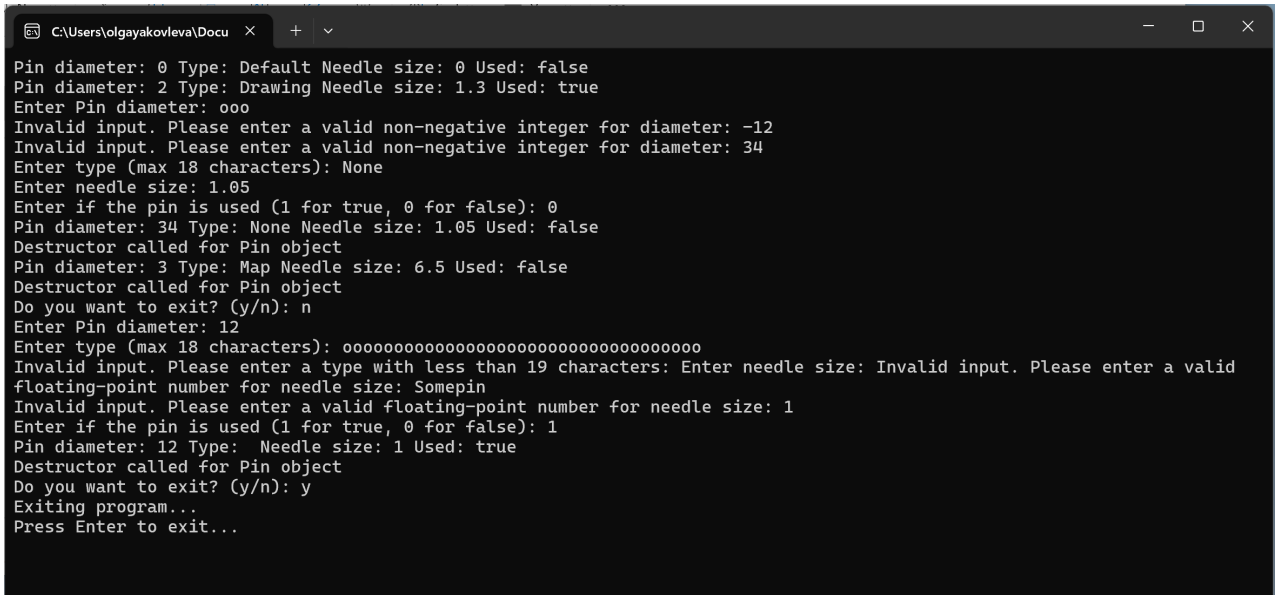

Рисунок 1—Результаты выполнения работы

3. ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были получены навыки работы классами, объектами, функциями в C++, а также стандартными потоками ввода-вывода.

ПРИЛОЖЕНИЕ А

Листинг программы

main.h

```
#ifndef LAB1_MAIN_H
#define LAB1_MAIN_H

#include <cstring> // для strncpy
class Pin {
// Данные открытого уровня доступа
public:
static const int MAX_LENGTH = 20; // Максимальная длина строки для поля type

// Конструкторы и деструктор
Pin(); // Конструктор по умолчанию
explicit Pin(int diameter, const char *type, float needle, bool used); // Конструктор с параметрами
~Pin(); // Деструктор

// Методы для работы с объектом
void Print() const; // Вывод информации об объекте
void Input(); // Ввод информации об объекте

// Геттеры
int getDiameter() const { return diameter; } // Возвращает значение диаметра
const char* getType() const { return type; } // Возвращает строку типа
float getNeedle() const { return needle; } // Возвращает значение иглы
bool isUsed() const { return used; } // Возвращает значение использования
```

```
// Сеттеры
bool setDiameter(int newDiameter); // Устанавливает значение диаметра
bool setType(const char* newType); // Устанавливает значение типа
bool setNeedle(float newNeedle); // Устанавливает значение иглы
bool setUsed(bool newUsed); // Устанавливает значение использования

// Данные закрытого уровня доступа
private:
int diameter; // Диаметр
char type[MAX_LENGTH]; // Тип
float needle; // Игла
bool used; // Используется ли объект
};

#endif
```

main.cpp

```
#include <iostream>
#include <cstring>
#include <limits>
#include "main.h"
using namespace std;
// Константы
```

const int MAX_TYPE_LENGTH = 19; // Максимальная длина строки для типа, равна 19, чтобы позволить безопасное копирование строки и корректное завершение строки нулевым символом.

const int MAX_LENGTH = 20; // Максимальная длина строки, может быть переопределена в классе Pin, задана на 20, чтобы дать пространство для ввода и нулевого символа.

```
const int MIN_DIAMETER = 0; // Минимально допустимое значение для диаметра
const float MIN_NEEDLE_SIZE = 0.0f; // Минимально допустимый размер иглы
// Определение конструктора по умолчанию класса Pin
Pin::Pin()
{
diameter = MIN_DIAMETER;   // Устанавливаем диаметр в 0
strncpy(type, "Default", MAX_TYPE_LENGTH); // Копируем строку "Default" в type, ограничивая длину
type[MAX_TYPE_LENGTH - 1] = '\0';   // Завершаем строку нулевым символом
needle = MIN_NEEDLE_SIZE;  // Устанавливаем размер иглы в 0.0
used = false;  // Устанавливаем флаг использования в false
}
// Определение конструктора класса Pin с параметрами
Pin::Pin(int diameter, const char* type, float needle, bool used)
{
this->diameter = diameter;  // Устанавливаем диаметр
strncpy(this->type, type, MAX_TYPE_LENGTH - 1); // Копируем тип, ограничивая длину
this->type[MAX_TYPE_LENGTH - 1] = '\0'; // Завершаем строку нулевым символом
this->needle = needle;  // Устанавливаем размер иглы
this->used = used;  // Устанавливаем флаг использования
}

// Определение деструктора класса Pin
Pin::~Pin()
{
```

```
cout << "Destructor called for Pin object" << endl; // Сообщение при вызове деструктора
}
// Определение метода Print класса Pin
void Pin::Print() const
{
// Вывод информации об объекте Pin с использованием геттеров для доступа к данным
cout << "Pin diameter: " << getDiameter()
<< " Type: " << getType()
<< " Needle size: " << getNeedle()
<< " Used: " << (isUsed() ? "true" : "false") << endl;
}
// Метод для установки диаметра
bool Pin::setDiameter(int dia)
{
// Устанавливаем диаметр, если он неотрицательный
if (dia >= MIN_DIAMETER)
{
diameter = dia;
return true;
}
return false;
}
// Метод для установки типа
bool Pin::setType(const char* t)
{
// Проверяем, что длина строки меньше максимального значения, и копируем тип
if (strlen(t) < MAX_TYPE_LENGTH)
```

```
{
strncpy(type, t, MAX_TYPE_LENGTH - 1);
type[MAX_TYPE_LENGTH - 1] = '\0';
return true;
}
return false;
}
// Метод для установки размера иглы
bool Pin::setNeedle(float n)
{
// Устанавливаем размер иглы, если он неотрицательный
if (n >= MIN_NEEDLE_SIZE)
{
needle = n;
return true;
}
return false;
}
// Метод для установки использования
bool Pin::setUsed(bool u)
{
// Устанавливаем флаг использования
used = u;
return true;
}
// Определение метода Input класса Pin
void Pin::Input()
{
int tempDiameter;
float tempNeedle;
```

```
char tempType[MAX_LENGTH];
int tempUsed;
// Ввод диаметра
cout << "Enter Pin diameter: ";
while (!(cin >> tempDiameter) || !setDiameter(tempDiameter))
{
cout << "Invalid input. Please enter a valid non-negative integer for diameter: ";
cin.clear(); // Очистка потока ввода
cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Игнорирование неверного ввода
}
cin.ignore(); // Игнорирование символа новой строки

// Ввод типа
std::cout << "Enter type (max " << MAX_TYPE_LENGTH - 1 << " characters): ";
while (true) {
std::cin.getline(tempType, MAX_TYPE_LENGTH); // Чтение строки

// Проверка и очистка потока
if (std::cin.fail()) {
std::cin.clear(); // Очистить состояние потока
std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Игнорировать остаточные символы новой строки
std::cout << "Invalid input. Please enter a type with less than " << MAX_TYPE_LENGTH - 1 << " characters: ";
continue;
}
if (setType(tempType)) {
```

```
break; // Установка типа, если длина корректная
}
else {
std::cout << "Invalid input. Please enter a type with less than " << MAX_TYPE_LENGTH << " characters: ";
}
}
// Ввод размера иглы
cout << "Enter needle size: ";
while (!(cin >> tempNeedle) || !setNeedle(tempNeedle))
{
cout << "Invalid input. Please enter a valid floating-point number for needle size: ";
cin.clear(); // Очистка потока ввода
cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Игнорирование неверного ввода
}
// Ввод использования
cout << "Enter if the pin is used (1 for true, 0 for false): ";
while (!(cin >> tempUsed) || (tempUsed != 0 && tempUsed != 1))
{
cout << "Invalid input. Please enter either 1 or 0 for used: ";
cin.clear(); // Очистка потока ввода
cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Игнорирование неверного ввода
}
setUsed(tempUsed); // Установка флага использования
}
// Главная функция программы
int main()
```

```
{
// Статическое выделение памяти для объектов pin и pin2 с использованием конструктора по умолчанию и с параметрами
Pin pin; // Создание объекта pin с использованием конструктора по умолчанию
pin.Print(); // Вывод информации о pin
Pin pin2(2, "Drawing", 1.3, true); // Создание объекта pin2 с передачей параметров в конструктор
pin2.Print();  // Вывод информации о pin2
// Динамическое выделение памяти для объектов pin3 и pin4 с использованием конструктора по умолчанию и с параметрами
Pin* pin3 = new Pin(); // Создание объекта pin3 с использованием конструктора по умолчанию
pin3->Input(); // Ввод данных о pin3
pin3->Print(); // Вывод информации о pin3
delete pin3;   // Освобождение памяти, занятой объектом pin3
Pin* pin4 = new Pin(3, "Map", 6.5, false); // Создание объекта pin4 с передачей параметров в конструктор
pin4->Print(); // Вывод информации о pin4
delete pin4;   // Освобождение памяти, занятой объектом pin4
// Используем цикл для повторного выполнения программы вместо рекурсивного вызова main
char choice;
do
{
cout << "Do you want to exit? (y/n): ";
cin >> choice;
if (choice == 'n' || choice == 'N')
{
Pin* newPin = new Pin(); // Создание нового объекта Pin
```

```
newPin->Input(); // Ввод данных для нового объекта
newPin->Print(); // Вывод информации о новом объекте
delete newPin;   // Освобождение памяти, занятой новым объектом
}
} while (choice != 'y' && choice != 'Y'); // Продолжение цикла, пока не будет введен 'y' или 'Y'
cout << "Exiting program...";
cout << "\nPress Enter to exit...";
cin.ignore(); // Ожидание нажатия клавиши Enter
cin.get();// Удержание консоли до нажатия Enter
return 0; // Завершение программы
}// Динамическое выделение памяти для объектов pin3 и pin4 с использованием конструктора по умолчанию и с параметрами
Pin *pin3 = new Pin(); // Создание объекта pin3 с использованием конструктора по умолчанию
pin3->Input(); // Ввод данных о pin3
pin3->Print(); // Вывод информации о pin3
delete pin3; // Освобождение памяти, занятой объектом pin3
Pin *pin4 = new Pin(3, "Map", 6.5, false); // Создание объекта pin4 с передачей параметров в конструктор
pin4->Print(); // Вывод информации о pin4
delete pin4; // Освобождение памяти, занятой объектом pin4
// Используем цикл для повторного выполнения программы вместо рекурсивного вызова main
char choice;
do
{
cout << "Do you want to exit? (y/n): ";
cin >> choice;
if (choice == 'n' || choice == 'N')
```

```
{
Pin *newPin = new Pin(); // Создание нового объекта Pin
newPin->Input(); // Ввод данных для нового объекта
newPin->Print(); // Вывод информации о новом объекте
delete newPin; // Освобождение памяти, занятой новым объектом
}
} while (choice != 'y' && choice != 'Y'); // Продолжение цикла, пока не будет введен 'y' или 'Y'
cout << "Exiting program...";
cout << "\nPress Enter to exit...";
cin.ignore(); // Ожидание нажатия клавиши Enter
cin.get();// Удержание консоли до нажатия Enter
return 0; // Завершение программы
}
```