|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**ЛАБОРАТОРНАЯ РАБОТА №1**

**«Классы и объекты в С++»**

**ДИСЦИПЛИНА: «Высокоуровневое программирование»**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-21Б | |  |  | ( | Суриков Н.С | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Пчелинцева Н. И. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

**Цель:** приобретение практических навыков и основ объектно-ориентированного

программирования, средствами языка C++.

**Задачи:**

1. Изучение основных концепций ООП;
2. Познакомиться с типом данных – «class»;
3. Познакомиться с операторами, предназначенными для работы с классами;
4. Научиться создавать объекты классов;
5. Изучить работу с методами класса;
6. Познакомиться с инициализацией пользовательских объектов.

**Условие задачи:**

1. Реализовать самостоятельно все примеры из методического пособия.
2. Разобраться с кодом проекта «MyMenu», высланный преподавателем.
3. В пункте 1 добавить по желанию свою реализацию (например, посчитать корень 25, как в примере на стр. 21, вывести псевдографику и т.д.).
4. Ответить на контрольные вопросы.

**Листинг программы:**

Main.cpp:

1 *#include* *"menu/CMenu.h"*

2 *#include* *"menu/CMenuItem.h"*

3

4 using namespace std;

5

6 *#pragma region функции-заглушки*

7

8 int exit()

9 {

10 std::cout << "Exit is running...\n\n";

11 exit(0);

12 return -1;

13 }

14

15 int hello()

16 {

17 std::cout << "Hello world!" << std::endl;

18 return 3;

19 }

20

21 int smileFace()

22 {

23 std::cout << std::endl;

24 std::cout << " \*\*\*\*\* " << std::endl;

25 std::cout << " \* \* " << std::endl;

26 std::cout << "\* o o \*" << std::endl;

27 std::cout << "\* v \*" << std::endl;

28 std::cout << " \* \* " << std::endl;

29 std::cout << " \*\*\*\*\* " << std::endl;

30 std::cout << std::endl;

31

32 return 1;

33 }

34

35 int squareOf25()

36 {

37 std::cout << "Square: " << 5 \* 5 << std::endl;

38 return 2;

39 }

40 *#pragma endregion*

41

42 const int ITEMS\_NUMBER = 4;

43

44 int main()

45 {

46 using namespace SNS;

47

48 CMenuItem items[ITEMS\_NUMBER]{CMenuItem{"Exit", exit},

49 CMenuItem{"Smile Face", smileFace},

50 CMenuItem("Square", squareOf25),

51 CMenuItem("Hello", hello)};

52

53 CMenu menu("My console menu", items, ITEMS\_NUMBER);

54 while (menu.runCommand())

55 {

56 };

57

58 return 0;

59 }

60

CMenuItem.h:

1 *#pragma once*

2

3 *#include* *<iostream>*

4

5

6 namespace SNS

7 {

8 class CMenuItem

9 {

10 public:

11 typedef int (\*Func)();

12 CMenuItem(std::string, Func);

13 Func func{};

14 std::string item\_name{};

15 std::string getName();

16 void print();

17 int run();

18 };

19 } *// namespace SNS*

CMenuItem.cpp:

1 *#include* *"./CMenuItem.h"*

2

3 namespace SNS

4 {

5 CMenuItem::CMenuItem(std::string name, Func func) : item\_name(name), func(func)

6 {

7 }

8

9 std::string CMenuItem::getName()

10 {

11 return item\_name;

12 }

13

14 void CMenuItem::print()

15 {

16 std::cout << item\_name;

17 }

18

19 int CMenuItem::run()

20 {

21 return func();

22 }

23 } *// namespace SNS*

24

CMenu.h:

1 *#pragma once*

2

3 *#include* *"./CMenuItem.h"*

4 *#include* *<cstddef>*

5

6 namespace SNS

7 {

8 class CMenu

9 {

10 public:

11 CMenu(std::string, CMenuItem \*, std::size\_t);

12 int getSelect() const;

13 bool isRun() const;

14 std::string getTitle();

15 size\_t getCount() const;

16 CMenuItem \*getItems();

17 void print();

18 int runCommand();

19

20 private:

21 int select{-1};

22 size\_t count{};

23 bool running{};

24 std::string title{};

25 CMenuItem \*items{};

26 };

27 } *// namespace SNS*

28

CMenu.cpp:

1 *#include* *"./CMenu.h"*

2

3 namespace SNS

4 {

5 CMenu::CMenu(std::string title, CMenuItem \*items, size\_t count) : title(title), items(items), count(count)

6 {

7 }

8

9 int CMenu::getSelect() const

10 {

11 return select;

12 }

13

14 bool CMenu::isRun() const

15 {

16 return running;

17 }

18

19 size\_t CMenu::getCount() const

20 {

21 return count;

22 }

23

24 std::string CMenu::getTitle()

25 {

26 return title;

27 }

28

29 CMenuItem \*CMenu::getItems()

30 {

31 return items;

32 }

33

34 void CMenu::print()

35 {

36 for (size\_t i{}; i < count; ++i)

37 {

38 std::cout << i << ". ";

39 items[i].print();

40 std::cout << std::endl;

41 }

42 }

43

44 int CMenu::runCommand()

45 {

46 print();

47 std::cout << "\n Select >> ";

48 std::cin >> select;

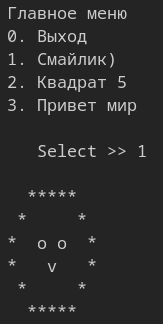
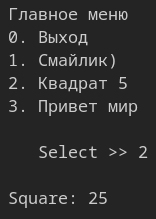
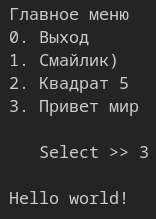
49 return items[select].run();

50 }

51 } *// namespace SNS*

52

**Результат работы:**

**  **

**Вывод:** в ходе выполнения лабораторной работы были получены

практические навыки работы с классами и библиотеками, а также

представление о работе объектно-ориентированного программирования

средствами языка С++.

**Контрольные вопросы:**

1. Что такое ООП?

Объектно-ориентированное программирование, концепция программирования в которой основной упор делается на данные, а не на алгоритмы. Это подход, при котором программа рассматривается как набор объектов, взаимодействующих друг с другом. У каждого есть свойства и поведение.

2. Какие ещё существуют парадигмы программирования?

* Императивное программирование
* Декларативное программирование
* *Структурное программирование*
* *Функциональное программирование*
* Логическое программирование
* Объектно-ориентированное программирование

3. Назовите основные концепции ООП.

* Инкапсуляция (связывает код и данные)
* Наследование (один объект приобретает свойства другого)
* Полиморфизм (использование одного и того же интерфейса для общего класса действий)

4. Что такое класс? Отличие класса от структуры.

Класс – механизм для создания объектов.

Основное отличие между классом и структурой заключается в том, что структура по умолчанию имеет открытый доступ к своим членам (переменным и функциям), в то время как класс по умолчанию имеет закрытый доступ.

5. Что такое экземпляр класса?

Экземпляр класса (объект) – если сам класс – это трафарет, то экземпляр класса – это реальный объект, в котором инициализированы все свойства класса.

6. Что такое поле класса? Как к нему обратиться?

Поля - Переменные, которые находятся в классе представляют его свойства. Для того, чтобы чётко понимать, что эти переменные являются полями класса, существует, общепринятое правило называть эти переменные начиная с префикса «m», сокращение от member – член, участник класса. Таким образом, объявление переменной в классе будет иметь синтаксис: int m\_age{};

7. Что такое метод класса? Как его вызвать?

Методы – помимо свойств класс имеет ряд функций, которые находятся в теле самого класса и описывают поведение объекта. (сам синтаксис описания функций ничем не отличается).

Чтобы вызвать такую функцию нужно обратиться к ней через точку, словно это поле объекта: Ivan.print(); (синтаксис ничем не отличается от вызова обычной функции, за тем исключением, что сперва мы указываем объект, у которого мы вызываем эту функцию)

8. Что такое конструктор и деструктор? Для чего они нужны?

Конструктор класса – функция, которая вызывается автоматически при создании объекта, имеет такое же имя, как и имя класса, в котором она объявлена, а также не имеет типа возвращаемого значения.

Конструкторы служат для начальной инициализации полей данных класса.

Деструктор вызывает либо компилятор, в конце использования, либо мы, если наш объект, который мы хотим уничтожить, находится в свободной памяти.

Деструктор точно также, как и конструктор, не может возвращать ничего из функции, поэтому тип возвращаемого значения для него не указывается.

Имеет название самого класса, но перед ним содержит символ тильда «~». И он не умеет работать с аргументами.

Деструкторы бывают полезны, когда в объекте идёт динамическое выделение памяти, либо же открытие потока. Компилятор не сможет автоматически закрыть поток, или освободить память, поэтому мы должны делать это явно в деструкторе.

9. Что такое модификаторы доступа? Для чего они используются?

Всего модификаторов доступа существует три:

* Public – после этого ключевого слова все поля и методы класса будут считаться доступными извне в любом файле или функции, которая импортирует наш класс.
* Private – после этого ключевого слова все поля и методы класса, будут считаться закрытыми для доступа извне. При этом к ним всё ещё можно обратиться в текущем классе, но вне класса вызвать их у экземпляра этого класса не выйдет.
* Protected – очень специфичный модификатор доступа. Все методы, поля у класса, которые идут после него, будут доступны только для текущего класса или его потомков.

10. Какими способами можно создать класс?

Способ 1: class [Имя класса]{ [Поля класса], public: [методы класса с описанием] };

Способ 2: class [Имя класса]{ [Поля класса], public: [методы класса без описания] };

[Имя класса]::[Имя метода](){

[Описание метода]

};

11. Какими способами можно создать объект класса?

Способ 1: [Имя класса] [Имя объекта] = {[Переменные для инициализации]}

Способ 2: [Имя класса] [Имя объекта] = [Конструктор]([Переменные для инициализации])

Способ 3: [Имя класса] [Имя объекта] = new [Конструктор]([Переменные для инициализации])

12. Что такое псевдоним типа, как его создать?

// создание псевдонима (обёртки) типа через typedef

typedef unsigned int money;

// создание псевдонима (обёртки) типа через alias (С++ 11)

using age = unsigned int;

В первом случае мы пишем ключевое слово – typedef, затем тип, для которого

хотим создать псевдоним, а затем имя этого псевдонима.

Во – втором случае мы пишем using, затем имя псевдонима, а после знака равно

тип, для которого создаём обёртку.

Обратите внимание, что cout автоматически преобразует наш псевдоним к

изначальному типу, это означает, что мы всего – лишь создали другое имя для

типа данных.

13. Как правильно реализовать класс на языке программирования C++?

Принято разделять прототип класса от реализации его функций. А именно

– прототипы классов, как и прототипы функций следует прописывать в

заголовочных файлах, а реализацию методов класса в файлах исходного кода.

14. Приведите пример программы, где нужно использовать ООП.

Игровая разработка: Использование классов для объектов в игре (игроков, врагов, предметов и т.д.) помогает организовать игровой мир, взаимодействия и логику.