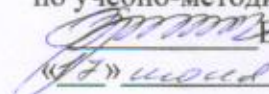


**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ НОВОСИБИРСКОЙ ОБЛАСТИ
«Новосибирский электромеханический колледж»
(ГБПОУ НСО «НЭК»)**

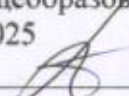
УТВЕРЖДАЮ

заместитель директора
по учебно-методической работе

 Е.О. Кравченко
«18» июня 2025 г.

**Методические указания
по выполнению практических работ
ОП.04. Основы алгоритмизации и программирования
09.02.07 Информационные системы и программирование**

Разработчик: Комарова А.В.,
преподаватель

Рассмотрено
на заседании кафедры общеобразовательных дисциплин
Протокол №10 от 16.06.2025
Заведующий кафедрой  Белимова В.Г.

Методические указания по выполнению практических работ по ОПЦ.04 Основы алгоритмизации и программирования разработаны в соответствии с рабочей программой по дисциплине специальности 09.02.07 Информационные системы и программирование.

Организация-разработчик: ГБПОУ НСО «НЭК»

Разработчик:

Комарова А.В., преподаватель высшей квалификационной категории

Согласовано:

Методист  Рахметова Т.Ю.

17.06.2025

СОДЕРЖАНИЕ

Оглавление

СОДЕРЖАНИЕ.....	3
Пояснительная записка.....	4
Порядок выполнения практической работы.....	4
Рекомендации по оформлению практической работы.....	4
Критерии оценки практической работы.....	5
Перечень практических работ.....	5
ПЗ №1. Знакомство со средой программирования.....	7
ПЗ №2. Составление программ линейной структуры.....	15
ПЗ №3. Составление программ разветвляющейся структуры.....	20
ПЗ №4. Составление программ разветвляющейся структуры.....	27
ПЗ №5. Составление программ циклической структуры.....	34
ПЗ №6. Составление программ циклической структуры.....	37
ПЗ №7. Обработка одномерных массивов.....	41
ПЗ №8. Обработка двумерных массивов.....	46
ПЗ №9. Обработка символьных строк.....	51
ПЗ №10. Обработка символьных строк.....	56
ПЗ №11. Работа с текстовыми файлами.....	59
ПЗ №12. Работа с двоичными файлами.....	63
ПЗ №13. Составление программ на типизированные файлы.....	67
ПЗ №14. Составление программ на нетипизированные файлы.....	73
ПЗ №18. Программирование модуля.....	88
ПЗ №19. Создание библиотеки подпрограмм.....	90
ПЗ №20. Использование указателей для организации связанных списков. Создание и удаление динамических переменных.....	90
ПЗ №25. События компонентов (элементов управления), их сущность и назначение.....	99
ПЗ №26. Создание процедур на основе событий.....	103
ПЗ №27. Создание проекта с использованием кнопочных компонентов.....	110
ПЗ №28. Создание проекта с использованием кнопочных компонентов.....	116
ПЗ №29. Создание проекта с использованием компонентов стандартных диалогов и системы меню.....	120
ПЗ №30. Создание проекта с использованием компонентов стандартных диалогов и системы меню.....	124
ПЗ №31. Разработка функциональной схемы работы приложения.....	128
ПЗ №32. Разработка оконного приложения с несколькими формами.....	131
ПЗ №33. Разработка игрового приложения.....	136
ПЗ №34. Создание процедур обработки событий. Компиляция и запуск приложения.....	141
ПЗ №35. Перегрузка методов. Тестирование и отладка приложения. Решение задач.....	145
ПЗ №36. Перегрузка методов. Тестирование и отладка приложения. Решение задач.....	149
ПЗ №37. Создание наследованного класса.....	153
ПЗ №38. Классы и объекты.....	157
ПЗ №39. Составление начальной иерархии и структуры классов.....	162
ПЗ №40. Итоговый проект.....	167
Список литературы.....	172
Приложение 1.....	173

Пояснительная записка

Методические указания по выполнению практических работ по учебной дисциплине ОП.04 Основы алгоритмизации и программирования разработаны в соответствии с рабочей программой учебной дисциплины и предназначены для приобретения необходимых практических навыков и закрепления теоретических знаний, полученных обучающимися при изучении учебной дисциплины, обобщения и систематизации знаний перед экзаменом.

Освоение содержания учебной дисциплины «Основы алгоритмизации и программирования» во время выполнения практических работ обеспечивает достижение обучающимися следующих результатов:

Код ПК, ОК	Умения	Знания
ОК 1 ОК 2 ОК 4 ОК 5 ОК 9 ОК 10 ПК 1.1- ПК 1.5 ПК 2.4, 2.5	Разрабатывать алгоритмы для конкретных задач. Использовать программы для графического отображения алгоритмов. Определять сложность работы алгоритмов. Работать в среде программирования. Реализовывать построенные алгоритмы в виде программ на конкретном языке программирования. Оформлять код программы в соответствии со стандартом кодирования. Выполнять проверку, отладку кода программы.	Понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции. Эволюцию языков программирования, их классификацию, понятие системы программирования. Основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти. Подпрограммы, составление библиотек подпрограмм Объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения

В соответствии с рабочей программой учебной дисциплины «Основы алгоритмизации и программирования» практические работы сгруппированы в конце четвертого семестра. Целесообразность данной группировки обусловлена необходимостью обобщения и систематизации знаний перед экзаменом.

Рабочая программа учебной дисциплины предусматривает проведение практических работ в объеме 80 часов.

Порядок выполнения практической работы

- записать название работы, ее цель в тетрадь;
- выполнить основные задания в соответствии с ходом работы;
- выполнить индивидуальные задания.

Рекомендации по оформлению практической работы

Задания выполняются обучающимися по шагам. Необходимо строго придерживаться порядка действий, описанного в практической работе.

Результаты выполнения практических заданий необходимо сохранять в своей папке на компьютере или USB – накопителе.

В случае пропуска занятий обучающийся осваивает материал самостоятельно в свободное от занятий время и сдает практическую работу с пояснениями о выполнении. Отчет о проделанной работе см. Приложение 1

Критерии оценки практической работы

- наличие цели выполняемой работы, выполнение более половины основных заданий (удовлетворительно);
- наличие цели выполняемой работы, выполнение всех основных и более половины дополнительных заданий (хорошо);
- наличие цели выполняемой работы, выполнение всех основных и индивидуальных заданий (отлично).

Перечень практических работ

№ п/п	Наименование практических работ	Кол- во часов
1.	ПЗ №1. Знакомство со средой программирования.	2
2.	ПЗ №2. Составление программ линейной структуры.	2
3.	ПЗ №3. Составление программ разветвляющейся структуры.	2
4.	ПЗ №4. Составление программ разветвляющейся структуры.	2
5.	ПЗ №5. Составление программ циклической структуры.	2
6.	ПЗ №6. Составление программ циклической структуры.	2
7.	ПЗ №7. Обработка одномерных массивов	2
8.	ПЗ №8. Обработка двумерных массивов	2
9.	ПЗ №9. Обработка символьных строк	2
10.	ПЗ №10. Обработка символьных строк	2
11.	ПЗ №11. Работа с текстовыми файлами.	2
12.	ПЗ №12. Работа с двоичными файлами.	2
13.	ПЗ №13. Составление программ на типизированные файлы.	2
14.	ПЗ №14. Составление программ на нетипизированные файлы.	2
15.	ПЗ №18. Программирование модуля.	2
16.	ПЗ №19. Создание библиотеки подпрограмм	2
17.	ПЗ №20. Использование указателей для организации связанных списков. Создание и удаление динамических переменных	2
18.	ПЗ №25. События компонентов (элементов управления), их сущность и назначение.	2
19.	ПЗ №26. Создание процедур на основе событий.	2
20.	ПЗ №27. Создание проекта с использованием кнопочных компонентов.	2
21.	ПЗ №28. Создание проекта с использованием кнопочных компонентов.	2
22.	ПЗ №29. Создание проекта с использованием компонентов стандартных диалогов и системы меню.	2
23.	ПЗ №30. Создание проекта с использованием компонентов стандартных диалогов и системы меню.	2
24.	ПЗ №31. Разработка функциональной схемы работы приложения.	2
25.	ПЗ №32. Разработка оконного приложения с несколькими формами.	2
26.	ПЗ №33. Разработка игрового приложения.	2
27.	ПЗ №34. Создание процедур обработки событий. Компиляция и запуск приложения.	2
28.	ПЗ №35. Перегрузка методов. Тестирование и отладка приложения. Решение задач	2

29.	ПЗ №36. Перегрузка методов. Тестирование и отладка приложения. Решение задач	2
30.	ПЗ №37. Создание наследованного класса.	2
31.	ПЗ №38. Классы и объекты.	2
32.	ПЗ №39. Составление начальной иерархии и структуры классов.	2
33.	ПЗ №40. Итоговый проект	2
Итого		80

ПЗ №1. Знакомство со средой программирования.

Тема: Знакомство со средой программирования Code::Blocks

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Освоить базовые навыки работы в интегрированной среде разработки (IDE) Code::Blocks и создать первые простые программы на C++.

Задачи:

1. Познакомиться с интерфейсом среды Code::Blocks.
2. Научиться создавать и настраивать проекты C++.
3. Освоить компиляцию, запуск и отладку программ.
4. Создать первые простые программы на языке C++.
5. Научиться находить и исправлять синтаксические ошибки.

2. Теоретические сведения

Code::Blocks — это бесплатная кроссплатформенная интегрированная среда разработки (IDE) для языков C, C++ и Fortran. Она поддерживает различные компиляторы, включая GCC (GNU Compiler Collection).

Основные элементы интерфейса:

1. **Меню** — доступ ко всем функциям среды.
2. **Панель инструментов** — быстрый доступ к часто используемым командам.
3. **Окно проекта (Management)** — отображает структуру файлов проекта.
4. **Редактор кода** — основное окно для написания программ.
5. **Окно журнала сборки (Build log)** — отображает сообщения компилятора.
6. **Окно отладчика** — для пошагового выполнения программы.

Процесс создания программы:

1. Создание проекта
2. Написание кода
3. Компиляция (Build)
4. Запуск (Run)
5. Отладка (при необходимости)

Структура простой программы на C++:

```
#include <iostream> // Подключение библиотеки ввода-вывода

using namespace std; // Использование стандартного пространства имен

int main()          // Главная функция программы
{
    // Тело функции
    cout << "Hello, World!" << endl; // Вывод текста на экран

    return 0;       // Завершение программы
}
```

3. Пример решения задачи

Задача: Создать программу, которая запрашивает имя пользователя и приветствует его.

```
#include <iostream>
#include <string> // Библиотека для работы со строками

using namespace std;
```

```
int main()
{
    string name; // Объявление переменной для имени

    cout << "Введите ваше имя: ";
    cin >> name; // Ввод имени с клавиатуры

    cout << "Здравствуйте, " << name << "!" << endl;
    cout << "Добро пожаловать в мир программирования на C++!" << endl;

    return 0;
}
```

4. Контрольные вопросы

1. Что такое интегрированная среда разработки (IDE)?
2. Какие основные окна содержит интерфейс Code::Blocks?
3. Как создать новый проект в Code::Blocks?
4. Что делает команда Build (F9)?
5. Какой компилятор используется по умолчанию в Code::Blocks?
6. Для чего нужна директива `#include <iostream>`?
7. Что такое функция `main()`?
8. Как вывести текст на экран в C++?
9. Как ввести данные с клавиатуры?
10. Что означает `endl`?

5. Задания к практической работе

Общее условие: Для каждого задания создайте отдельный проект в Code::Blocks. Сохраняйте проекты в отдельной папке с именем "Практическая_работа_1".

Задание 1: Первая программа

1. Создайте новый проект с именем `FirstProgram`.
2. Напишите программу, которая выводит на экран ваши ФИО, группу и название специальности.
3. Каждая строка должна выводиться отдельной командой `cout`.
4. Запустите программу и убедитесь, что она работает правильно.

Пример вывода:

```
text
Иванов Иван Иванович
Группа: ИСП-20-1
Специальность: Информационные системы
```

Задание 2: Простой калькулятор

1. Создайте проект `SimpleCalculator`.
2. Напишите программу, которая запрашивает два числа с клавиатуры.
3. Выведите сумму, разность, произведение и частное этих чисел.
4. Форматируйте вывод так, чтобы каждое действие было на новой строке.

Пример работы программы:

```
text
Введите первое число: 10
Введите второе число: 5
```

Результаты:

```
10 + 5 = 15
10 - 5 = 5
```

$$10 * 5 = 50$$

$$10 / 5 = 2$$

Задание 3: Расчет периметра и площади

1. Создайте проект `Geometry`.
2. Напишите программу для расчета периметра и площади прямоугольника.
3. Программа должна запрашивать длину и ширину прямоугольника.
4. Выведите результаты расчетов с пояснениями.

Формулы:

- Периметр: $P = 2 * (a + b)$
- Площадь: $S = a * b$

Задание 4: Конвертер величин

1. Создайте проект `Converter`.
2. Напишите программу для конвертации километров в метры.
3. Запросите у пользователя количество километров.
4. Выведите результат в метрах.
5. *Дополнительно:* добавьте конвертацию сантиметров в метры.

Пример:

text

Введите расстояние в километрах: 3.5

3.5 км = 3500 метров

Задание 5: Работа с символами

1. Создайте проект `Characters`.
2. Напишите программу, которая запрашивает у пользователя символ.
3. Выведите этот символ, а также его ASCII-код.
4. Выведите символы, которые идут перед и после введенного в таблице ASCII.

Пример:

text

Введите символ: C

Вы ввели: C

Код ASCII: 67

Предыдущий символ: B (66)

Следующий символ: D (68)

Задание 6: Личная карточка студента

1. Создайте проект `StudentCard`.
2. Напишите программу, которая запрашивает у пользователя:
 - Фамилию
 - Имя
 - Возраст
 - Средний балл
3. Выведите все данные в виде аккуратной карточки с рамкой из символов `*`.
4. Добавьте расчет количества лет до окончания колледжа (предположим, что учиться нужно 3 года).

Пример:

text

```
*****
*      ЛИЧНАЯ КАРТОЧКА      *
*****
* Фамилия:  Иванов          *
* Имя:      Иван            *
* Возраст:  18 лет          *
* Средний балл: 4.5         *
* Осталось учиться: 2 года  *
```

Критерии оценки:

- **5 баллов** — все 6 программ работают без ошибок
- **4 балла** — работают 4-5 программ
- **3 балла** — работают 3 программы
- **2 балла** — менее 3 программ работают правильно

ПЗ №2. Составление программ линейной структуры.

Тема: Основы языка C++. Линейные алгоритмы

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Освоить базовые элементы языка C++ и научиться создавать программы с линейными алгоритмами.

Задачи:

1. Изучить базовые типы данных и операции языка C++.
2. Научиться работать с переменными и константами.
3. Освоить ввод и вывод данных в консольных приложениях.
4. Научиться использовать математические функции.
5. Создать программы с линейными алгоритмами.

2. Теоретические сведения (кратко)

Основные типы данных в C++:

- `int` - целые числа
- `float` и `double` - вещественные числа
- `char` - символы
- `bool` - логические значения (`true/false`)

Базовые операции:

- Арифметические: `+`, `-`, `*`, `/`, `%`
- Операции сравнения: `==`, `!=`, `<`, `>`, `<=`, `>=`
- Логические: `&&` (И), `||` (ИЛИ), `!` (НЕ)

Ввод/вывод в C++:

```
#include <iostream>
using namespace std;
```

```
int main() {
    int a;
    double b;

    // Ввод
    cout << "Введите число: ";
    cin >> a;

    // Вывод
    cout << "Вы ввели: " << a << endl;
    cout << "Результат: " << a * 2 << endl;

    return 0;
}
```

Математические функции (подключать `#include <cmath>`):

- `sqrt(x)` - квадратный корень
- `pow(x, y)` - `x` в степени `y`
- `abs(x)` - модуль числа
- `sin(x)`, `cos(x)`, `tan(x)` - тригонометрические функции

3. Контрольные вопросы

1. Какие типы данных в C++ используются для хранения целых чисел?
2. Чем отличается `float` от `double`?
3. Для чего используется директива `#include <cmath>`?

4. Как вывести число с определенным количеством знаков после запятой?
5. Что такое линейный алгоритм?
6. Какая функция вычисляет квадратный корень?
7. Как получить модуль числа в C++?
8. Что делает оператор endl?
9. Как организовать ввод нескольких переменных в одной строке?
10. Какие библиотеки нужно подключить для работы с математическими функциями?

4. Задания к практической работе

Общее условие: Для каждого варианта создайте отдельный проект. Название проекта: `Practical_2_Variant_`[номер варианта].

ВАРИАНТ 1

1. Площадь и периметр прямоугольника

Даны стороны прямоугольника. Найти его площадь и периметр.

2. Среднее арифметическое

Даны три числа. Найти их среднее арифметическое.

3. Конвертер температур

Перевести температуру из градусов Цельсия в градусы Фаренгейта по формуле:

$$F = C * 9/5 + 32$$

4. Сумма цифр трехзначного числа

Дано трехзначное число. Найти сумму его цифр.

5. Объем цилиндра

Даны радиус основания и высота цилиндра. Найти его объем.

6. Следующее и предыдущее число

Дано целое число. Вывести следующее и предыдущее число.

ВАРИАНТ 2

1. Площадь круга

Дан радиус круга. Найти его площадь.

2. Среднее геометрическое

Даны два положительных числа. Найти их среднее геометрическое.

3. Расстояние между точками

Даны координаты двух точек на плоскости. Найти расстояние между ними.

4. Перестановка цифр

Дано трехзначное число. Вывести число, полученное перестановкой первой и последней цифр.

5. Площадь треугольника по формуле Герона

Даны стороны треугольника. Найти его площадь по формуле Герона.

6. Процент от числа

Даны число и процент. Найти значение этого процента от числа.

ВАРИАНТ 3

1. Площадь поверхности куба

Дана длина ребра куба. Найти площадь его поверхности.

2. Модуль разности

Даны два числа. Найти модуль их разности.

3. Время в секундах

Дано время в часах, минутах и секундах. Найти общее количество секунд.

4. Четность числа

Дано целое число. Определить, является ли оно четным.

5. Корни квадратного уравнения

Даны коэффициенты квадратного уравнения. Найти его корни.

6. Стоимость покупки

Даны цена товара и количество. Найти общую стоимость покупки.

ВАРИАНТ 4

1. **Объем параллелепипеда**

Даны длина, ширина и высота параллелепипеда. Найти его объем.

2. **Сумма и произведение**

Даны три числа. Найти их сумму и произведение.

3. **Конвертер валют**

Дана сумма в рублях и курс доллара. Перевести сумму в доллары.

4. **Обратное число**

Дано целое число. Вывести число, обратное данному.

5. **Скорость движения**

Даны расстояние и время. Найти скорость движения.

6. **Квадрат и куб числа**

Дано число. Найти его квадрат и куб.

ВАРИАНТ 5

1. **Длина окружности**

Дан радиус окружности. Найти ее длину.

2. **Разность квадратов**

Даны два числа. Найти разность их квадратов.

3. **Площадь прямоугольного треугольника**

Даны катеты прямоугольного треугольника. Найти его площадь.

4. **Первая цифра числа**

Дано трехзначное число. Вывести его первую цифру.

5. **Средняя скорость**

Даны скорость на первом участке пути, скорость на втором участке и время на каждом участке. Найти среднюю скорость на всем пути.

6. **Сумма арифметической прогрессии**

Даны первый член, разность и количество членов арифметической прогрессии. Найти сумму прогрессии.

ВАРИАНТ 6

1. **Объем шара**

Дан радиус шара. Найти его объем.

2. **Максимум из трех чисел**

Даны три числа. Найти максимальное из них.

3. **Периметр равностороннего треугольника**

Дана сторона равностороннего треугольника. Найти его периметр.

4. **Последняя цифра числа**

Дано целое число. Вывести его последнюю цифру.

5. **Площадь трапеции**

Даны основания и высота трапеции. Найти ее площадь.

6. **Проценты по вкладу**

Даны сумма вклада, процентная ставка и срок в годах. Найти сумму на счету через указанное время.

Критерий	Максимальный балл
Все 6 программ работают без ошибок	3
Правильное использование типов данных	2

Критерий	Максимальный балл
Качественный ввод/вывод с пояснениями	2
Использование математических функций (если нужно)	1
Аккуратное форматирование кода	1
Итого	9

Оценки:

- 8-9 баллов: отлично
- 6-7 баллов: хорошо
- 4-5 баллов: удовлетворительно
- менее 4: неудовлетворительно

Советы по выполнению:

1. Начинайте с самых простых задач
2. Тестируйте программы с разными входными данными
3. Проверяйте правильность вычислений на калькуляторе
4. Комментируйте сложные моменты в коде
5. Сохраняйте промежуточные версии программ

ПЗ №3. Составление программ разветвляющейся структуры.

Тема: Составление программ разветвляющейся структуры

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Научиться создавать программы с использованием условных операторов для реализации разветвляющихся алгоритмов.

Задачи:

1. Освоить синтаксис операторов `if`, `else if`, `else`.
2. Научиться использовать оператор выбора `switch`.
3. Применять логические операции в условиях.
4. Создавать вложенные условные конструкции.
5. Реализовывать сложные условия с использованием логических операторов.

2. Теоретические сведения

Условный оператор `if`:

// Полная форма

```
if (условие) {  
    // выполняется если условие истинно (true)  
    операторы;  
} else {  
    // выполняется если условие ложно (false)  
    операторы;  
}
```

// Неполная форма

```
if (условие) {  
    операторы;  
}
```

// Форма с несколькими условиями

```
if (условие1) {  
    операторы;  
} else if (условие2) {  
    операторы;  
} else {  
    операторы;  
}
```

Оператор выбора `switch`:

```
switch (выражение) {  
    case значение1:  
        операторы;  
        break;  
    case значение2:  
        операторы;  
        break;  
    default:  
        операторы;  
}
```

Логические операции:

- `&&` - И (and)
- `||` - ИЛИ (or)
- `!` - НЕ (not)

Операции сравнения:

- `==` - равно
- `!=` - не равно
- `<` - меньше
- `>` - больше
- `<=` - меньше или равно
- `>=` - больше или равно

3. Контрольные вопросы

1. Какой оператор используется для проверки нескольких вариантов значений одной переменной?
2. В чем отличие операторов `=` и `==`?
3. Как проверить, что число находится в диапазоне от 10 до 20?
4. Что произойдет, если забыть `break` в операторе `switch`?
5. Можно ли использовать оператор `switch` для проверки вещественных чисел?
6. Как проверить, что число четное?
7. Что выведет программа: `if (5) cout << "true"; else cout << "false";`?
8. Какой оператор имеет более высокий приоритет: `&&` или `||`?
9. Можно ли использовать несколько `else if` в одной конструкции?
10. Какой тип данных должен быть у выражения в `switch`?

4. Задания к практической работе

Общее условие: Каждый студент получает один вариант. Создайте отдельный проект в Qt Creator для каждого варианта. Название проекта: `PR_3_Variant_[номер]`.

ВАРИАНТ 1

Задача 1: Определить, является ли число четным или нечетным.

Задача 2: Найти максимальное из трех чисел.

Задача 3: Проверить, является ли год високосным. Год високосный, если он делится на 4, но не делится на 100, либо делится на 400.

Задача 4: Определить четверть координатной плоскости, в которой находится точка с координатами (x, y). Учесть, что точка может лежать на осях.

Задача 5: По номеру месяца вывести время года (зима, весна, лето, осень). Использовать оператор `switch`.

Задача 6: Решить квадратное уравнение $ax^2 + bx + c = 0$. Учесть все случаи ($D > 0$, $D = 0$, $D < 0$).

Задача 7: Проверить, является ли треугольник с заданными сторонами прямоугольным.

ВАРИАНТ 2

Задача 1: Определить знак введенного числа.

Задача 2: Найти минимальное из трех чисел.

Задача 3: Проверить, образуют ли три числа возрастающую последовательность.

Задача 4: Определить, принадлежит ли точка с координатами (x, y) кругу с центром в начале координат и радиусом R.

Задача 5: По оценке студента вывести словесное описание: 5 - отлично, 4 - хорошо, 3 - удовлетворительно, 2 - неудовлетворительно, 1 - плохо. Использовать `switch`.

Задача 6: Вычислить значение функции:

text

$f(x) = x^2$, если $x \leq 0$

$f(x) = 1/x$, если $x > 0$

Задача 7: Проверить, можно ли из трех отрезков заданной длины составить треугольник.

ВАРИАНТ 3

Задача 1: Проверить, делится ли число на 3 и на 5 одновременно.

Задача 2: Упорядочить три числа по возрастанию.

Задача 3: Определить, является ли число двузначным.

Задача 4: Проверить, лежит ли точка внутри прямоугольника с вершинами (0,0), (a,0), (a,b), (0,b).

Задача 5: По номеру дня недели определить, выходной это или рабочий день. Использовать switch.

Задача 6: Вычислить значение функции:

text

$f(x) = 2x + 5$, если $x < -1$

$f(x) = x^2$, если $-1 \leq x \leq 1$

$f(x) = x - 3$, если $x > 1$

Задача 7: Определить тип треугольника по сторонам (равносторонний, равнобедренный, разносторонний).

Критерии оценки:

Критерий	Баллы
Все 7 программ работают без ошибок	3
Правильное использование if/else	2
Правильное использование switch	1
Использование логических операторов	1
Качественный ввод/вывод с пояснениями	1
Обработка всех возможных случаев	1
Аккуратное форматирование кода	1
Итого	10

Шкала оценок:

- 9-10 баллов: отлично
- 7-8 баллов: хорошо
- 5-6 баллов: удовлетворительно
- менее 5: неудовлетворительно

Пример тестирования:

Для каждой задачи покажите работу программы с разными входными данными:

1. Нормальный случай
2. Граничный случай
3. Ошибочный ввод (если применимо)

Пример для задачи 2 варианта 1:

text

Тест 1:

Введите три числа: 5 10 3

Максимальное число: 10

Тест 2:

Введите три числа: -1 -5 -3

Максимальное число: -1

Тест 3:

Введите три числа: 7 7 7

Максимальное число: 7

Советы по выполнению:

1. Начните с простых задач (1-3)
2. Внимательно проверяйте условия
3. Используйте отступы для улучшения читаемости кода
4. Тестируйте программы с разными входными данными
5. Проверяйте граничные случаи
6. Для сложных условий используйте скобки для явного указания приоритета
7. Не забывайте про `break` в операторе `switch`
8. Всегда обрабатывайте случай `default` в `switch`
9. Комментируйте сложные логические выражения
10. Сохраняйте промежуточные версии программ

ПЗ №4. Составление программ разветвляющейся структуры.

Тема: Составление программ разветвляющейся структуры (продолжение)

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Закрепить навыки создания сложных разветвляющихся алгоритмов с использованием вложенных условий и комбинации различных условных конструкций.

Задачи:

1. Освоить создание сложных логических выражений.
2. Научиться работать с вложенными условными операторами.
3. Применять операторы `switch` для сложных задач.
4. Создавать программы с многоуровневыми ветвлениями.
5. Реализовывать проверку сложных условий.

2. Теоретические сведения

Сложные логические выражения:

// Проверка диапазона

```
if (x >= 10 && x <= 20) {  
    // x находится в диапазоне [10, 20]  
}
```

// Проверка нескольких условий

```
if (условие1 && условие2 || условие3) {  
    // Порядок выполнения: &&, затем ||  
}
```

// Использование скобок для изменения приоритета

```
if ((условие1 || условие2) && условие3) {  
    // Сначала проверяются условие1 ИЛИ условие2,  
    // затем результат И условие3  
}
```

Вложенные условные операторы:

```
if (условие1) {  
    if (условие2) {  
        // Выполняется если условие1 И условие2 истинны  
    } else {  
        // Выполняется если условие1 истинно, а условие2 ложно  
    }  
} else {  
    if (условие3) {  
        // Выполняется если условие1 ложно, а условие3 истинно  
    }  
}
```

Комбинация switch и if:

```
switch (выражение) {  
    case значение1:  
        if (дополнительное_условие) {  
            операторы;  
        }  
}
```

```
break;  
// другие case  
}
```

3. Контрольные вопросы

1. Как проверить, что число находится вне диапазона [10, 20]?
2. В чем разница между `if (a && b)` и `if (a || b)`?
3. Как использовать оператор `switch` для проверки диапазона значений?
4. Что такое вложенные условия и когда их следует использовать?
5. Как проверить, что число положительное и нечетное?
6. Можно ли использовать оператор `switch` внутри `if`?
7. Как организовать проверку ввода пользователя на корректность?
8. Что выведет: `if (0 && 1 || 1) cout << "true";`?
9. Как оптимизировать сложные логические выражения?
10. Когда лучше использовать `switch`, а когда `if-else`?

4. Задания к практической работе

Общее условие: Каждый студент получает один вариант. Создайте отдельный проект для каждого варианта. Название проекта: `PR_4_Variant_[номер]`. Каждая задача должна быть в отдельном файле ..

ВАРИАНТ 1

Задача 1: Определить, принадлежит ли точка (x, y) области:

- Внутри круга радиусом 5 с центром в начале координат
- И вне квадрата со стороной 8, центр в начале координат

Задача 2: Для студента определить статус по среднему баллу:

- 4.5-5.0: "Отличник"
- 4.0-4.49: "Хорошист"
- 3.0-3.99: "Троечник"
- менее 3.0: "Неуспевающий"

Дополнительно проверить, есть ли задолженности.

Задача 3: Определить, является ли число "счастливым" билетом (6-значное число, сумма первых трех цифр равна сумме последних трех).

Задача 4: По дате (день, месяц, год) определить:

- Является ли год високосным
- Существует ли такая дата
- Время года

Задача 5: Рассчитать стоимость билета в зависимости от:

- Возраста (детский до 12, взрослый 12-60, пенсионер 60+)
- Типа билета (обычный, льготный, VIP)
- Дня недели (скидка в выходные)

Задача 6: Определить вид треугольника по координатам трех вершин:

- Равносторонний
- Равнобедренный
- Прямоугольный
- Разносторонний

Задача 7: Система начисления бонусов:

- Если покупка > 10000 и клиент постоянный (+5%)
- Если день рождения (+10%)
- Если акционный товар (+15%)

Максимальный бонус 25%.

ВАРИАНТ 2

Задача 1: Определить, попадает ли точка (x, y) в область:

- Внутри прямоугольника от (-2,-2) до (2,2)
- И вне круга радиусом 1 с центром в начале координат

Задача 2: Калькулятор налогов:

- Если доход < 20000 - налог 13%
- Если 20000-50000 - налог 20%
- Если > 50000 - налог 30%

Для пенсионеров скидка 5%.

Задача 3: Проверить, является ли число палиндромом (читается одинаково слева направо и справа налево).

Задача 4: По времени (часы, минуты) определить:

- Утро (6-11)
- День (12-17)
- Вечер (18-23)
- Ночь (0-5)

Вывести приветствие.

Задача 5: Расчет стоимости парковки:

- Первый час: 50 руб.
- Каждый следующий час: 30 руб.
- После 5 часов: 200 руб. фиксировано
- Для инвалидов скидка 50%

Задача 6: Определить взаимное расположение двух окружностей (заданы координаты центров и радиусы):

- Не пересекаются
- Касаются
- Пересекаются
- Одна внутри другой

Задача 7: Система оценки студентов:

- Экзамен: 0-59 - "Неуд", 60-74 - "Удовл", 75-89 - "Хор", 90-100 - "Отл"
- Учитывать посещаемость ($< 70\%$ - снижение на балл)
- Учитывать активность ($+5\%$ за активность)

ВАРИАНТ 3

Задача 1: Определить, принадлежит ли точка (x, y) области:

- $y > x^2$
- $y < 2x + 1$
- $x > 0$

Задача 2: Определить категорию товара по цене и сроку годности:

- Дорогой (> 1000) и свежий (> 30 дней)
- Средний (500-1000) или свежий
- Дешевый (< 500) и несвежий
- Уценка (несвежий и любой цены)

Задача 3: Проверить корректность номера телефона в формате +7(XXX)XXX-XX-XX.

Задача 4: По номеру карты (16 цифр) определить:

- Тип карты (Visa: начинается на 4, MasterCard: 51-55)
- Проверить контрольную сумму (алгоритм Луна)
- Вывести маскированный номер

Задача 5: Расчет стоимости доставки:

- Вес < 1 кг: 100 руб.
- 1-5 кг: 200 руб.
- 5-10 кг: 300 руб.
- 10 кг: 500 руб.

Плюс срочная доставка $\times 2$.

Задача 6: Определить, можно ли из заданных длин сторон построить треугольник, и если да, то вычислить его площадь и периметр.

Задача 7: Система бронирования отеля:

- Тип номера (эконом, стандарт, люкс)
- Количество ночей
- Сезон (низкий, средний, высокий)
- Скидка постоянным клиентам 10%

Требования к выполнению:

Общие требования:

1. Каждая программа должна проверять корректность входных данных
2. Использовать осмысленные имена переменных
3. Добавлять комментарии к сложным логическим выражениям
4. Форматировать вывод для удобства чтения
5. Тестировать все возможные случаи

Критерии оценки:

Критерий	Макс. балл
Корректность работы всех программ	3
Использование сложных логических выражений	2
Применение вложенных условий	2
Проверка корректности входных данных	1
Качественный информативный вывод	1
Обработка всех возможных случаев	1
Итого	10

Шкала оценок:

- 9-10 баллов: отлично
- 7-8 баллов: хорошо
- 5-6 баллов: удовлетворительно
- менее 5: неудовлетворительно

ПЗ №5. Составление программ циклической структуры.

Тема: Основы циклических структур. Простые циклы

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Освоить создание программ с использованием простых циклических структур.

Задачи:

1. Изучить синтаксис циклов `for`, `while`, `do-while`.
 2. Научиться применять циклы для суммирования рядов и вычисления факториалов.
 3. Освоить работу с цифрами чисел в циклах.
 4. Создать простые программы с использованием одного цикла.
- ### 2. Теоретические сведения

Три вида циклов в C++:

1. **Цикл `for`** (со счетчиком):

```
for (инициализация; условие; изменение) {  
    // тело цикла  
}
```

2. **Цикл `while`** (с предусловием):

```
while (условие) {  
    // тело цикла  
}
```

3. **Цикл `do-while`** (с постусловием):

```
do {  
    // тело цикла  
} while (условие);
```

Основные отличия:

- `for` используется, когда известно количество итераций
- `while` проверяет условие ДО выполнения тела цикла
- `do-while` проверяет условие ПОСЛЕ выполнения тела цикла (гарантирует хотя бы одно выполнение)

3. Контрольные вопросы

1. Какой цикл гарантирует хотя бы одно выполнение тела?
2. В чем разница между `i++` и `++i`?
3. Что произойдет, если условие цикла всегда истинно?
4. Как выйти из цикла досрочно?
5. Как пропустить текущую итерацию цикла?
6. Можно ли использовать вещественные числа в счетчике цикла `for`?
7. Что выведет программа:

```
for (int i = 0; i < 5; i++) {  
    if (i == 2) continue;  
    if (i == 4) break;  
    cout << i << " ";  
}
```

8. Какой цикл лучше использовать для обработки пользовательского ввода до тех пор, пока не будет введено определенное значение?

4. Задания к практической работе

Вариант 1

1. **Факториал числа**

Вычислить $n!$ с использованием цикла `for`.

Пример: $5! = 120$

2. **Сумма ряда**

Вычислить сумму: $S = 1 + 1/2 + 1/3 + \dots + 1/n$

Вход: n (натуральное число)

3. **Количество цифр**

Подсчитать количество цифр в заданном целом числе.

Пример: 12345 $\rightarrow$ 5 цифр

4. **Сумма цифр**

Найти сумму цифр заданного числа.

Пример: 123 $\rightarrow 1+2+3 = 6$

5. **Таблица умножения на 7**

Вывести таблицу умножения числа 7 от 1 до 10.

6. **Обратное число**

Вывести цифры числа в обратном порядке.

Пример: 123 $\rightarrow$ 321

7. **Четные числа**

Вывести все четные числа от 2 до 100.

Вариант 2

1. **Факториал (while)**

Вычислить n! с использованием цикла while.

2. **Сумма квадратов**

Вычислить сумму: $S = 1^2 + 2^2 + 3^2 + \dots + n^2$

3. **Проверка палиндрома**

Проверить, является ли число палиндромом.

Пример: 121 $\rightarrow$ является палиндромом

4. **Максимальная цифра**

Найти максимальную цифру в числе.

Пример: 52891 $\rightarrow$ 9

5. **Таблица умножения на 9**

Вывести таблицу умножения числа 9 от 1 до 10.

6. **Простые числа до N**

Вывести все простые числа от 2 до N.

7. **Гармонический ряд**

Вычислить: $S = 1 - 1/2 + 1/3 - 1/4 + \dots \pm 1/n$

Критерии оценки (Практическая работа 5):

Критерий	Баллы
Все 7 программ работают	4
Использованы разные типы циклов	2
Проверка корректности ввода	2
Качественные комментарии	1
Форматирование вывода	1
Итого	10

ПЗ №6. Составление программ циклической структуры.

Тема: Сложные циклические структуры. Вложенные циклы

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Освоить создание программ со сложными циклическими структурами и вложенными циклами.

Задачи:

1. Научиться работать с вложенными циклами.
2. Освоить решение комбинаторных задач с помощью циклов.
3. Изучить использование циклов для работы с последовательностями.
4. Создать программы для решения математических задач.

2. Теоретические сведения

Вложенные циклы:

```
for (int i = 0; i < n; i++) {    // Внешний цикл
    for (int j = 0; j < m; j++) { // Внутренний цикл
        // Тело вложенного цикла
    }
}
```

Операторы управления циклом:

- `break` - выход из текущего цикла
- `continue` - переход к следующей итерации текущего цикла
- `return` - выход из функции

Особенности вложенных циклов:

- Внутренний цикл выполняется полностью для каждой итерации внешнего
- Общее количество итераций = $n \times m$
- Можно использовать разные типы циклов

3. Контрольные вопросы

1. Сколько раз выполнится тело внутреннего цикла:

```
for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 3; j++) {
        cout << "*";
    }
}
```

2. Как выйти из всех вложенных циклов сразу?
3. Что делает оператор `continue` во вложенном цикле?
4. Как оптимизировать вложенные циклы для повышения производительности?
5. В чем разница между `break` и `return` внутри цикла?
6. Можно ли использовать `while` внутри `for`?
7. Как вывести треугольник из звездочек с помощью вложенных циклов?

4. Задания к практической работе

Вариант 1

1. Полная таблица умножения

Вывести полную таблицу умножения от 1 до 10.

2. Треугольник из звездочек

Вывести треугольник:

```
*
**
***
****
*****
```

3. **Трехзначные числа с уникальными цифрами**

Вывести все трехзначные числа, все цифры которых различны.

4. **Задача Л. Эйлера**

Найти все решения уравнения: $31x + 21y = 1770$ в натуральных числах.

5. **Сумма ряда с косинусами**

Вычислить: $y = \cos x + \cos 2x + \dots + \cos 30x$

6. **Простые пары**

Найти все пары простых чисел (p, q) такие, что $p + q = 100$.

7. **Шахматная доска**

Вывести шахматную доску 8×8 (белые клетки - '□', черные - '■').

Вариант 2

1. **Таблица Пифагора**

Вывести таблицу Пифагора в аккуратном формате.

2. **Ромб из звездочек**

Вывести ромб:

text

*

*

3. **Числа, сумма цифр которых равна N**

Найти все трехзначные числа, сумма цифр которых равна заданному N.

4. **Синусы с шагом**

Вычислить: $y = \sin 1 + \sin 1.1 + \sin 1.2 + \dots + \sin 2$

5. **Совершенные числа**

Найти все совершенные числа до 10000.

6. **Дружественные числа**

Найти все пары дружественных чисел до 1000.

7. **Треугольник Паскаля**

Вывести первые 10 строк треугольника Паскаля.

Критерии оценки:

Критерий	Баллы
Все 7 программ работают	5
Использование вложенных циклов	3
Оптимизация алгоритмов	2
Качественный вывод результатов	2
Обработка граничных случаев	2
Комментарии к сложным алгоритмам	1
Итого	15

Шкала оценок:

- 13-15 баллов: отлично
- 10-12 баллов: хорошо
- 7-9 баллов: удовлетворительно
- менее 7: неудовлетворительно

ПЗ №7. Обработка одномерных массивов

Тема: Обработка одномерных массивов

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Освоить методы работы с одномерными массивами в языке C++.

Задачи:

1. Изучить способы объявления и инициализации массивов.
2. Научиться выполнять ввод и вывод элементов массива.
3. Освоить основные алгоритмы обработки массивов.
4. Научиться находить минимальные и максимальные элементы.
5. Освоить сортировку и поиск в массивах.

2. Теоретические сведения

Массив — это структура данных, содержащая фиксированное количество элементов одного типа. Элементы массива хранятся в непрерывной области памяти и доступны по индексу.

Объявление массива:

тип имя_массива[размер];

Примеры:

```
int A[10];           // Массив из 10 целых чисел
double B[5] = {1.5, 2.7, 3.1, 4.8, 5.2}; // Инициализация
float C[] = {0.1, 0.2, 0.3}; // Автоматическое определение размера
```

Обращение к элементам массива:

- Индексация начинается с 0
- Последний элемент имеет индекс (размер-1)
- Пример: A[0] — первый элемент, A[9] — последний элемент массива размером 10

Основные операции с массивами:

1. Ввод/вывод массива
2. Поиск минимального/максимального элемента
3. Суммирование элементов
4. Сортировка элементов
5. Поиск по условию
3. Контрольные вопросы
 1. Может ли массив содержать один элемент?
 2. Можно ли во время выполнения программы изменить размер массива?
 3. Могут ли элементами некоторого массива быть числа 1, 1.41, 4.98, 30?
 4. Верно ли, что тип элементов массива может быть любым?
 5. Какие операции над массивами как единым целым допустимы в языке C++?
 6. Можно ли переписать элементы массива x в массив y с помощью оператора y=x?
4. Задания к практической работе

Общее условие: Размер массивов во всех задачах — 10 элементов (если не указано иное).

ВАРИАНТ 1

Задача 1: Дан массив из 10 целых чисел. Найти сумму чисел, расположенных между минимальным и максимальным элементами (включая их).

Задача 2: Дан массив из 10 вещественных чисел. Если минимальный элемент равен 0, поменять местами минимальный и максимальный элементы.

Задача 3: Дан массив из 10 целых чисел. Найти наибольшее из нечетных чисел и количество четных чисел.

Задача 4: Дан массив из 10 целых чисел. Если минимальный элемент меньше максимального в 2 раза, вывести TRUE, иначе FALSE.

Задача 5: Дан массив из 15 целых чисел. Если минимальный элемент предшествует максимальному, подсчитать сумму этих элементов.

Задача 6: Дан массив из 10 целых чисел. Если минимальный на 5-м месте, а максимальный на 10-м, найти среднее арифметическое этих элементов.

ВАРИАНТ 2

Задача 1: Дан массив из 10 целых чисел. Найти минимальный элемент и заменить его целой частью среднего арифметического всех элементов.

Задача 2: Дан массив из 10 вещественных чисел. Если максимальный элемент равен 0, поменять местами минимальный и максимальный элементы.

Задача 3: Дан массив из 10 целых чисел. Найти наименьшее из четных чисел (если такие есть).

Задача 4: Дан массив из 10 целых чисел. Если минимальный элемент находится на первом месте, а максимальный — на последнем, поменять их местами.

Задача 5: Дан массив из 15 вещественных чисел. Если максимальный элемент предшествует минимальному, подсчитать их произведение.

Задача 6: Дан массив из 10 целых чисел. Если минимальный элемент находится на 1-м месте, а максимальный на 2-м, обнулить все элементы от минимального до конца массива.

ВАРИАНТ 3

Задача 1: Дан массив из 10 целых чисел. Поменять местами наибольшее и наименьшее число в массиве.

Задача 2: Дан массив из 10 вещественных чисел. Поменять местами минимальный элемент с первым элементом, а максимальный — с последним.

Задача 3: Дан массив из 10 целых чисел. Подсчитать количество элементов, удовлетворяющих условию: $A[k] < (A[k-1] + A[k+1])/2$ (для k от 1 до 8).

Задача 4: Дан массив из 10 целых чисел. Если минимальный элемент меньше максимального в 3 раза, увеличить все элементы в 3 раза.

Задача 5: Дан массив из 15 целых чисел. Если минимальный или максимальный элемент равен 0, обнулить все элементы, предшествующие минимальному.

Задача 6: Дан массив из 10 целых чисел. Если минимальный элемент меньше максимального в 2 раза, все элементы перед максимальным сделать равными минимальному.

ВАРИАНТ 4

Задача 1: Дан массив из 10 целых чисел (среди которых есть одинаковые). Создать новый массив из различных элементов исходного.

Задача 2: Дан массив из 10 вещественных чисел. Установить максимальный и минимальный элементы равными нулю.

Задача 3: Дан массив из 10 целых чисел. Подсчитать количество элементов, удовлетворяющих условию: $A[k] > 2*k$.

Задача 4: Дан массив из 10 целых чисел. Если минимальный находится на 2-м месте, а максимальный на 1-м, уменьшить все элементы в 2 раза.

Задача 5: Дан массив из 15 целых чисел. Найти расстояние (в элементах) между максимальным и минимальным элементами.

Задача 6: Дан массив из 10 целых чисел. Если максимальный на 2-м месте, а минимальный на 5-м, все элементы от минимального до конца сделать равными максимальному.

Требования к выполнению:

Для каждой задачи:

1. Создать отдельный файл .
2. Объявить массив соответствующего типа
3. Реализовать ввод массива с клавиатуры (или инициализацию)

4. Выполнить требуемые операции
5. Вывести результаты в понятном формате

Критерии оценки:

Критерий	Макс. балл
Все 6 программ работают	4
Правильное объявление и работа с массивами	2
Корректный поиск min/max элементов	2
Правильная реализация условий задач	2
Качественный ввод/вывод данных	2
Комментарии в коде	1
Форматирование вывода	1
Тестирование с разными данными	1
Итого	15

Шкала оценок:

- 13-15 баллов: отлично
- 10-12 баллов: хорошо
- 7-9 баллов: удовлетворительно
- менее 7: неудовлетворительно

ПЗ №8. Обработка двумерных массивов

Тема: Обработка двумерных массивов (матриц)

Дисциплина: Основы алгоритмизации и программирования

1. Цели и задачи практической работы

Цель: Освоить методы работы с двумерными массивами (матрицами) в языке C++.

Задачи:

1. Изучить способы объявления и инициализации двумерных массивов.
 2. Научиться выполнять ввод, вывод и обработку матриц.
 3. Освоить работу с строками и столбцами матриц.
 4. Научиться находить максимальные и минимальные элементы в матрицах.
 5. Освоить базовые операции над матрицами.
2. Теоретические сведения

Двумерный массив (матрица) — это массив массивов. Каждый элемент матрицы имеет два индекса: номер строки и номер столбца.

Объявление матрицы:

тип имя_матрицы[количество_строк][количество_столбцов];

Примеры:

```
int A[3][4];           // Матрица 3x4 целых чисел
double B[2][3] = {     // Инициализация
    {1.5, 2.7, 3.1},
    {4.2, 5.8, 6.3}
};
float C[][3] = {        // Автоматическое определение строк
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Обращение к элементам матрицы:

- $A[i][j]$ — элемент в i -й строке, j -м столбце
- Индексация начинается с 0
- Для матрицы $N \times M$ строки: $0..N-1$, столбцы: $0..M-1$

3. Контрольные вопросы

1. Как объявить матрицу 5×10 целых чисел?
2. Как обратиться к элементу в 3-й строке, 2-м столбце матрицы A?
3. Что такое главная диагональ квадратной матрицы?
4. Как найти сумму элементов строки матрицы?
5. Как поменять местами две строки матрицы?
6. Что такое симметричная матрица?
7. Как создать единичную матрицу?

4. Задания к практической работе

Общее условие: Размеры матриц во всех задачах — 4×4 (квадратная) или 4×5 (прямоугольная), если не указано иное. Для каждого варианта создайте отдельный проект в Qt Creator. Время выполнения: 2 академических часа (рекомендуется выполнить 3-4 задачи).

ВАРИАНТ 1 (квадратные матрицы 4×4)

Задача 1: Найти максимальный элемент матрицы и переставить строки и столбцы так, чтобы он оказался в левом верхнем углу.

Задача 2: Проверить, является ли матрица симметричной относительно главной диагонали.

Задача 3: В каждой строке найти максимальный элемент и поменять его местами с элементом на главной диагонали.

Задача 4: Найти сумму элементов строки, содержащей минимальный элемент матрицы.

ВАРИАНТ 2 (прямоугольные матрицы 4×5)

Задача 1: Упорядочить строки матрицы по возрастанию их первых элементов.

Задача 2: Найти столбец с минимальным произведением элементов (только среди столбцов, где все элементы ≤ 10 по модулю).

Задача 3: Для каждой строки найти максимальный и минимальный элементы, вычислить их произведение.

Задача 4: Найти среднее арифметическое элементов каждой строки.

ВАРИАНТ 3 (квадратные матрицы 4×4)

Задача 1: Упорядочить строки матрицы по возрастанию суммы их элементов.

Задача 2: Поменять местами строки, симметричные относительно середины (1-ю с 4-й, 2-ю с 3-й).

Задача 3: Найти строку, содержащую только нечетные элементы, с максимальной суммой модулей.

Задача 4: Все элементы с максимальным значением по модулю заменить нулями.

ВАРИАНТ 4 (прямоугольные матрицы 4×5)

Задача 1: Упорядочить строки матрицы по возрастанию их максимальных элементов.

Задача 2: Поменять местами столбцы, симметричные относительно середины.

Задача 3: Найти среднее арифметическое элементов каждого столбца.

Задача 4: Для каждого столбца найти максимальный и минимальный элементы, вычислить их среднее арифметическое.

Требования к выполнению:

Для каждой задачи:

1. Создать отдельный файл .
2. Объявить матрицу соответствующего размера
3. Реализовать ввод матрицы с клавиатуры (или инициализацию)
4. Вывести исходную матрицу
5. Выполнить требуемые операции
6. Вывести результаты в понятном формате

Критерии оценки за 2 часа работы:

Критерий	Макс. балл
Выполнено 3 задачи	6
Выполнено 4 задачи	8
Правильное объявление и работа с матрицами	2
Корректная реализация алгоритмов	2
Качественный ввод/вывод данных	2
Комментарии в коде	1
Форматирование вывода	1
Итого (максимум)	14

Шкала оценок:

- 12-14 баллов: отлично
- 9-11 баллов: хорошо
- 6-8 баллов: удовлетворительно
- менее 6: неудовлетворительно

ПЗ №9. Обработка символьных строк.

Тема: Обработка символьных строк

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Освоить методы работы с символьными строками в языке C++.

Задачи:

1. Изучить способы объявления и инициализации строк.
 2. Научиться использовать функции для работы со строками.
 3. Освоить операции сравнения и поиска в строках.
 4. Научиться выполнять базовые манипуляции со строками.
 5. Изучить стандартный класс string.
2. Теоретические сведения

Два способа работы со строками в C++:

1. **Строки в стиле C (символьные массивы):**

```
char str1[20] = "Привет"; // с явным размером
char str2[] = "Мир";      // с автоматическим определением размера
```

2. **Строки в стиле C++ (класс string):**

```
#include <string>
string str1 = "Привет";
string str2 = "Мир";
```

Основные особенности:

- Строки в стиле C заканчиваются нуль-символом '\0'
- Класс string более удобен и безопасен
- Для строк в стиле C используются функции из <cstring>
- Для класса string используются методы объекта

3. Контрольные вопросы

1. **Как описываются строковые переменные?**
 - В стиле C: `char str[размер];`
 - В стиле C++: `string str;`
2. **Какие операции допустимы над строковыми данными?**
 - Сравнение, объединение, копирование, поиск, замена, определение длины
3. **В чем отличие строковой переменной от массива символов?**
 - Строка всегда заканчивается '\0', массив символов может не заканчиваться
4. **Какие стандартные функции для работы со строками вы знаете?**
 - Для C: `strlen()`, `strcpy()`, `strcat()`, `strcmp()`
 - Для C++: методы класса string: `length()`, `find()`, `substr()`, `replace()`
5. **Чему равно значение `x[0]` после присваивания `x='вопрос'`?**
 - 'в' (первый символ строки)

4. Задания к практической работе

Общее условие: Для каждого варианта создайте отдельный проект в Qt Creator. Время выполнения: 2 академических часа (рекомендуется выполнить 4 задачи). Используйте класс string для всех задач.

ВАРИАНТ 1

Задача 1: Сравнение строк (выполнить сравнения из теоретической части):

- "Процессор" ____ "Процесс"
- "кошка" ____ "кошка"
- "муха" ____ "СЛОН"
- "Иванов" ____ "Петров"

Задача 2: Определить, сколько в строке гласных букв (русских).

Задача 3: Удалить из строки все символы заданной буквы.

Задача 4: Даны две строки S1 и S2. Создать новую строку: первые 3 символа из S1 + последние 2 символа из S2.

ВАРИАНТ 2

Задача 1: Сравнение строк:

- "Balkon" ____ "balkon"
- "окно" ____ "окно"
- "кино" ____ "кино"
- "Смирнов" ____ "Смирнова"

Задача 2: Подсчитать количество содержащихся в строке прописных латинских букв.

Задача 3: Заменить в строке все вхождения одной подстроки на другую.

Задача 4: Дана строка с пробелами. Вывести подстроку между первым и вторым пробелом.

ВАРИАНТ 3

Задача 1: Сравнение строк:

- "мышь" ____ "мышь"
- "окно" ____ "ОКНО"
- "Процесс" ____ "процесс"
- "Строка" ____ "Строки"

Задача 2: Удалить из строки все вхождения заданной подстроки.

Задача 3: Подсчитать количество вхождений подстроки в строку.

Задача 4: Даны символ C и строки S, S0. Перед каждым вхождением символа C в строку S вставить строку S0.

ВАРИАНТ 4

Задача 1: Сравнение строк:

- "Apple" ____ "apple"
- "Windows" ____ "Windows"
- "Linux" ____ "UNIX"
- "Java" ____ "JavaScript"

Задача 2: В строке имеются только две одинаковых буквы. Найти их.

Задача 3: Вставить символ после каждого вхождения другого символа.

Задача 4: Дана строка с пробелами. Вывести подстроку между первым и последним пробелом.

Требования к выполнению:

Для каждой задачи:

1. Создать отдельный файл .
2. Использовать класс string
3. Реализовать ввод данных с клавиатуры
4. Вывести исходные данные и результаты
5. Добавить комментарии

Критерии оценки работы:

Критерий	Макс. балл
Выполнено 3 задачи	6
Выполнено 4 задачи	8
Правильное использование string	2

Критерий	Макс. балл
Корректная реализация алгоритмов	2
Качественный ввод/вывод	2
Обработка ошибок ввода	1
Форматирование вывода	1
Итого (максимум)	14

Шкала оценок за 2 часа:

- 12-14 баллов: отлично
- 9-11 баллов: хорошо
- 6-8 баллов: удовлетворительно
- менее 6: неудовлетворительно

Рекомендация для 2-часовой работы: Выполните 3 основные задачи вашего варианта.

ПЗ №10. Обработка символьных строк.

Тема: Сложная обработка символьных строк

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Научиться выполнять сложные операции обработки текста с использованием строк в C++.

Задачи:

1. Освоить разбиение текста на слова и обработку отдельных слов.
2. Научиться работать с текстом как с последовательностью слов.
3. Изучить методы поиска и фильтрации слов по различным критериям.
4. Освоить преобразование строк в числа и обратно.
5. Реализовать алгоритмы обработки текстовых сообщений.

2. Теоретические сведения

Обработка текста в C++:

1. **Разбиение строки на слова:**

```
string text = "Привет, мир! Как дела?";  
vector<string> words;  
string word;
```

```
for (char c : text) {  
    if (isalpha(c)) { // если символ - буква  
        word += c;  
    } else if (!word.empty()) { // если слово не пустое  
        words.push_back(word);  
        word.clear();  
    }  
}
```

2. **Преобразование символа в число:**

```
char digit = '5';  
int number = digit - '0'; // получаем число 5
```

3. **Проверка символов:**

```
isdigit(c) // проверка на цифру  
isalpha(c) // проверка на букву  
isspace(c) // проверка на пробельный символ  
ispunct(c) // проверка на знак препинания
```

4. Задания к практической работе

Общее условие: Для каждого варианта создайте отдельный проект в Qt Creator. Время выполнения: 2 академических часа (рекомендуется выполнить 4 задачи). Используйте класс string и vector для всех задач.

ВАРИАНТ 1 (Работа с текстом)

Задача 1: Подсчитать, сколько раз заданное слово встречается в тексте.

Задача 2: Найти самое длинное слово в тексте.

Задача 3: Вывести все слова, которые начинаются на заданную букву.

Задача 4: Дан текст, содержащий цифры. Вывести на экран все имеющиеся в нем цифры.

ВАРИАНТ 2 (Работа с текстом)

Задача 1: Найти самое короткое слово в тексте (исключая однобуквенные).

Задача 2: Подсчитать количество слов в тексте.

Задача 3: Вывести все слова, которые начинаются и оканчиваются на одну и ту же букву.

Задача 4: Дан текст, содержащий цифры. Вывести на экран их сумму.

ВАРИАНТ 3 (Работа с текстом)

Задача 1: Вывести все слова-палиндромы в тексте.

Задача 2: Вывести все слова, состоящие из N букв.

Задача 3: Преобразовать текст так, чтобы каждое слово начиналось с заглавной буквы.

Задача 4: Дан текст, содержащий цифры. Вывести на экран количество цифр в нем.

ВАРИАНТ 4 (Работа с текстом)

Задача 1: Удалить из текста все однобуквенные слова.

Задача 2: Удалить из текста все повторяющиеся слова.

Задача 3: Поменять слова в тексте по принципу: первое со вторым, третье с четвертым и т.д.

Задача 4: Дан текст, содержащий цифры. Вывести на экран наибольшую цифру.

Требования к выполнению:

Для каждой задачи:

1. Создать отдельный файл .
2. Использовать класс string и vector
3. Реализовать ввод текста с клавиатуры
4. Вывести результаты в понятном формате
5. Добавить комментарии к алгоритмам

Критерии оценки работы:

Критерий	Макс. балл
Выполнено 3 задачи	6
Выполнено 4 задачи	8
Правильная обработка текста	2
Корректное разбиение на слова	2
Обработка знаков препинания	1
Качественный вывод результатов	1
Комментарии к алгоритмам	1
Обработка ошибок ввода	1
Итого (максимум)	14

Шкала оценок за 2 часа:

- 12-14 баллов: отлично
- 9-11 баллов: хорошо
- 6-8 баллов: удовлетворительно
- менее 6: неудовлетворительно

ПЗ №11. Работа с текстовыми файлами.

Тема: Работа с текстовыми файлами

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Освоить базовые операции работы с текстовыми файлами в языке C++.

Задачи:

1. Изучить способы открытия, чтения и записи текстовых файлов.
2. Научиться выполнять обработку строк в файлах.
3. Освоить операции с несколькими файлами одновременно.
4. Научиться фильтровать и преобразовывать данные в файлах.
5. Реализовать основные алгоритмы обработки файлов.

2. Теоретические сведения

Основные операции с файлами в C++:

1. **Подключение библиотеки:**

```
#include <fstream>
```

2. **Открытие файлов:**

```
ifstream inFile;    // для чтения  
ofstream outFile;   // для записи  
fstream ioFile;     // для чтения и записи
```

3. **Открытие файла:**

```
inFile.open("input.txt");  
outFile.open("output.txt");
```

4. **Проверка открытия:**

```
if (!inFile.is_open()) {  
    cout << "Ошибка открытия файла!" << endl;  
    return 1;  
}
```

5. **Чтение из файла:**

```
string line;  
while (getline(inFile, line)) {  
    // обработка строки  
}
```

6. **Запись в файл:**

```
outFile << "Текст для записи" << endl;
```

7. **Закрытие файлов:**

```
inFile.close();  
outFile.close();
```

4. Задания к практической работе

ВАРИАНТ 1

Задача 1: Найти количество строк в файле, которые начинаются с заданной буквы.

Задача 2: Найти самую длинную строку в файле и вывести ее номер и длину.

Задача 3: Переписать в новый файл все строки, вставив в конец каждой строки ее номер.

Задача 4: Дан файл с целыми числами. Переписать все четные числа в один файл, нечетные — в другой.

ВАРИАНТ 2

Задача 1: Найти количество строк в файле, которые начинаются и заканчиваются одной буквой.

Задача 2: Найти самую короткую строку в файле и вывести ее номер и длину.

Задача 3: Переписать в новый файл все строки, длина которых больше заданного числа.

Задача 4: Дан файл с целыми числами. Переписать все положительные числа в один файл, отрицательные — в другой.

ВАРИАНТ 3

Задача 1: Выяснить, имеется ли в файле строка, которая начинается с заданной буквы. Если да, то вывести ее.

Задача 2: Вывести все нечетные строки файла (1, 3, 5...).

Задача 3: Переписать в новый файл все строки, вставив в конец каждой строки количество символов в ней.

Задача 4: Дан файл с символами. Переписать в один файл все цифры, в другой — все буквы.

ВАРИАНТ 4

Задача 1: Вывести первый символ каждой строки файла.

Задача 2: Вывести все строки файла, в которых имеется хотя бы один пробел.

Задача 3: Переписать в новый файл все строки четной длины.

Задача 4: Даны два файла с числами. Создать новый файл, каждый элемент которого равен сумме соответствующих чисел из исходных файлов.

Требования к выполнению:

Для каждой задачи:

1. Создать отдельный файл .
2. Создать тестовые файлы с данными
3. Проверять успешность открытия файлов
4. Выводить результаты на экран и/или в файлы
5. Добавлять комментарии

Критерии оценки работы:

Критерий	Макс. балл
Выполнено 3 задачи	6
Выполнено 4 задачи	8
Корректная работа с файлами	2
Проверка открытия файлов	1
Обработка ошибок	1
Чтение и запись данных	1
Качественный вывод результатов	1
Создание тестовых файлов	1

Критерий	Макс. балл
Итого (максимум)	14

Шкала оценок за 2 часа:

- 12-14 баллов: отлично
- 9-11 баллов: хорошо
- 6-8 баллов: удовлетворительно
- менее 6: неудовлетворительно

Рекомендация для 2-часовой работы: Выполните 3 основные задачи вашего варианта.

ПЗ №12. Работа с двоичными файлами.

Тема: Работа с двоичными файлами

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Освоить работу с двоичными файлами в языке C++.

Задачи:

1. Изучить особенности работы с двоичными файлами.
2. Научиться записывать и читать различные типы данных в двоичные файлы.
3. Освоить позиционирование в двоичных файлах.
4. Научиться выполнять операции с последовательными и произвольным доступом.
5. Реализовать основные алгоритмы обработки данных в двоичных файлах.

2. Теоретические сведения

Особенности двоичных файлов:

- Хранят данные в бинарном формате (как в памяти)
- Сохраняют точное представление данных
- Не используют символы конца строки
- Более эффективны для числовых данных

Основные операции с двоичными файлами:

1. **Открытие файла в двоичном режиме:**

```
fstream file("data.bin", ios::binary | ios::in | ios::out);
```

2. **Запись данных:**

```
int number = 42;  
file.write((char*)&number, sizeof(number));
```

3. **Чтение данных:**

```
int number;  
file.read((char*)&number, sizeof(number));
```

4. **Определение размера файла:**

```
file.seekg(0, ios::end); // перейти в конец файла  
size_t fileSize = file.tellg(); // получить позицию (размер)  
file.seekg(0, ios::beg); // вернуться в начало
```

5. **Позиционирование:**

```
file.seekg(offset, ios::beg); // от начала файла  
file.seekg(offset, ios::cur); // от текущей позиции  
file.seekg(offset, ios::end); // от конца файла
```

4. Задания к практической работы

ВАРИАНТ 1

Задача 1: Создать двоичный файл и записать в него квадраты натуральных чисел от 1 до N. Вывести на экран все компоненты файла с нечетными порядковыми номерами.

Задача 2: Создать двоичный файл, состоящий из N целых чисел. Вывести на экран все четные числа из файла.

Задача 3: Создать двоичный файл, состоящий из N вещественных чисел. Найти сумму всех положительных чисел в файле.

Задача 4: Создать двоичный файл с N целыми числами. Вывести все числа, не кратные заданному числу K.

ВАРИАНТ 2

Задача 1: Создать двоичный файл и записать в него степени числа 3 ($3^1, 3^2, \dots, 3^N$). Вывести на экран все компоненты файла с четными порядковыми номерами.

Задача 2: Создать двоичный файл, состоящий из N целых чисел. Вывести на экран все отрицательные числа из файла.

Задача 3: Создать двоичный файл, состоящий из N вещественных чисел. Найти максимальное значение среди чисел, стоящих на нечетных позициях.

Задача 4: Создать двоичный файл с N вещественными числами. Вывести все числа, не попадающие в заданный диапазон [A, B].

ВАРИАНТ 3

Задача 1: Создать двоичный файл и записать в него обратные натуральные числа ($1, 1/2, \dots, 1/N$). Вывести на экран все компоненты файла с порядковыми номерами, кратными 3.

Задача 2: Создать двоичный файл, состоящий из N целых чисел. Вывести на экран все числа из файла, попадающие в заданный интервал [MIN, MAX].

Задача 3: Создать двоичный файл, состоящий из N вещественных чисел. Подсчитать среднее арифметическое чисел, стоящих на четных позициях.

Задача 4: Создать двоичный файл с N вещественными числами. Вывести все числа с нечетными номерами, большие заданного числа X.

ВАРИАНТ 4

Задача 1: Создать двоичный файл и записать в него N первых членов последовательности Фибоначчи. Вывести на экран все компоненты файла с порядковыми номерами, не кратными 3.

Задача 2: Создать двоичный файл, состоящий из N целых чисел. Вывести на экран все числа из файла, не попадающие в заданный интервал [MIN, MAX].

Задача 3: Создать двоичный файл, состоящий из N вещественных чисел. Вывести на экран все числа с четными номерами, меньшие заданного числа X.

Задача 4: Создать двоичный файл с N целыми числами. Вывести все числа, кратные заданному числу K.

Требования к выполнению:

Для каждой задачи:

1. Создать отдельный файл .
2. Реализовать создание двоичного файла с тестовыми данными
3. Реализовать чтение и обработку данных из файла
4. Выводить результаты на экран в понятном формате
5. Добавлять комментарии к коду

Критерии оценки работы:

Критерий	Макс. балл
Выполнено 3 задачи	6
Выполнено 4 задачи	8
Корректная работа с двоичными файлами	2
Правильное чтение/запись данных	1
Обработка ошибок ввода/вывода	1

Критерий	Макс. балл
Позиционирование в файле	1
Качественный вывод результатов	1
Использование sizeof() для типов	1
Итого (максимум)	14

Шкала оценок за 2 часа:

- 12-14 баллов: отлично
- 9-11 баллов: хорошо
- 6-8 баллов: удовлетворительно
- менее 6: неудовлетворительно

ПЗ №13. Составление программ на типизированные файлы.

Тема: Работа с типизированными файлами (файлами записей)

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Освоить работу с типизированными файлами - двоичными файлами, организованными как последовательность записей фиксированного размера.

Задачи:

1. Изучить особенности структурированных файлов записей.
2. Научиться создавать и работать с файлами структур.
3. Освоить прямой доступ к записям по номеру.
4. Реализовать базовые операции CRUD (Create, Read, Update, Delete).
5. Изучить методы поиска и сортировки в типизированных файлах.

2. Теоретические сведения

Типизированный файл - это двоичный файл, организованный как последовательность записей одинакового размера, где каждая запись представляет собой структуру данных.

Ключевые особенности:

- Все записи имеют одинаковый размер
- Прямой доступ к записи по номеру: позиция = номер_записи * sizeof(структура)
- Эффективный произвольный доступ
- Сохранение структуры данных

Основные операции:

// Позиционирование к записи N

```
file.seekg(N * sizeof(Record), ios::beg);
```

// Чтение записи

```
Record rec;
```

```
file.read((char*)&rec, sizeof(Record));
```

// Запись записи

```
file.write((char*)&rec, sizeof(Record));
```

// Количество записей в файле

```
file.seekg(0, ios::end);
```

```
int count = file.tellg() / sizeof(Record);
```

3. Контрольные вопросы

1. Чем типизированный файл отличается от обычного двоичного файла?
2. Как вычислить позицию записи с номером N в файле?
3. Что такое логическое удаление записей?
4. Как организовать поиск по полю в типизированном файле?
5. В чем преимущество прямого доступа к записям?
6. Как определить количество записей в файле?
7. Что такое "уплотнение" файла и зачем оно нужно?

4. Задания к практической работе

Общее условие: Для каждого варианта создайте отдельный проект в Qt Creator. Время выполнения: 2 академических часа (рекомендуется выполнить 3 задачи).

ВАРИАНТ 1 - Студенты

Структура:

cpp

```
struct Student {  
    int id;        // номер зачетки
```

```

char name[50]; // ФИО
int group;     // номер группы
double mark;   // средний балл
};

```

Задание 1: Создать файл и записать в него 3 студента

Задание 2: Прочитать и вывести всех студентов из файла

Задание 3: Найти студента по номеру зачетки

ВАРИАНТ 2 - Книги

Структура:

```

cpp
struct Book {
    int id;           // шифр книги
    char title[100]; // название
    char author[50]; // автор
    int year;         // год издания
};

```

Задание 1: Создать файл и записать в него 3 книги

Задание 2: Прочитать и вывести все книги из файла

Задание 3: Найти книги указанного автора

ВАРИАНТ 3 - Сотрудники

Структура:

```

cpp
struct Employee {
    int id;           // табельный номер
    char name[50];    // ФИО
    char post[30];    // должность
    double salary;    // зарплата
};

```

Задание 1: Создать файл и записать в него 3 сотрудника

Задание 2: Прочитать и вывести всех сотрудников из файла

Задание 3: Найти сотрудников с зарплатой больше заданной

ВАРИАНТ 4 - Товары

Структура:

```

cpp
struct Product {
    int id;           // артикул
    char name[50];    // название
    char cat[30];     // категория
    double price;     // цена
    int quantity;     // количество
};

```

Задание 1: Создать файл и записать в него 3 товара

Задание 2: Прочитать и вывести все товары из файла

Задание 3: Найти товары заданной категории

Требования к выполнению:

Для каждой задачи:

1. Создать отдельный файл .cpp
2. Реализовать класс для работы с файлом
3. Создать тестовые данные
4. Выводить результаты на экран

5. Добавить обработку ошибок

Критерии оценки:

Критерий	Макс. балл
Выполнено 3 задачи	6
Выполнено 4 задачи	8
Корректная работа с типизированным файлом	2
Реализация CRUD операций	2
Обработка ошибок	1
Качественный интерфейс	1
Создание тестовых данных	1
Итого (максимум)	14

Шкала оценок за 2 часа:

- 12-14 баллов: отлично
- 9-11 баллов: хорошо
- 6-8 баллов: удовлетворительно
- менее 6: неудовлетворительно

ПЗ №14. Составление программ на нетипизированные файлы в C++.

Тема: Составление программ на нетипизированные

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи

Цель: Научиться работать с бинарными файлами

Задачи:

- Открывать/закрывать бинарные файлы
- Записывать и читать различные типы данных
- Обрабатывать ошибки при работе с файлами

2. Теоретические сведения

// Основные операции:

ofstream **out**("file.bin", ios::binary); *// запись*

ifstream **in**("file.bin", ios::binary); *// чтение*

fstream **file**("file.bin", ios::binary | ios::in | ios::out); *// чтение и запись*

file.**write**((**char***&переменная, **sizeof**(переменная));

file.**read**((**char***&переменная, **sizeof**(переменная));

3. Контрольные вопросы

1. Какой флаг используется для бинарного режима?
2. Как записать структуру в файл?
3. Что делает функция **sizeof**()?
4. Как проверить, открылся ли файл?
5. Почему нужно закрывать файл?

4. Задания к практической работе (12 заданий)

Задание 1: Создать бинарный файл и записать в него 20 случайных чисел от 1 до 100.

Задание 2: Прочитать файл из задания 1 и вывести числа, которые больше 50.

Задание 3: Записать в файл массив из 10 структур:

```
struct Product {  
    char name[30]; // Название товара  
    float price; // Цена  
    int quantity; // Количество  
};
```

Задание 4: Из файла с товарами найти самый дорогой товар.

Задание 5: Создать файл с данными студентов:

```
struct Student {  
    char surname[30];  
    int grades[5]; // 5 оценок  
};
```

Записать 8 студентов.

Задание 6: Для файла студентов посчитать средний балл каждого.

Задание 7: Создать копию бинарного файла (файл-дубликат).

Задание 8: Объединить два бинарных файла в один.

Задание 9: В файле с числами заменить все отрицательные числа на их модуль.

Задание 10: Создать файл-журнал температур за месяц (31 день), записать случайные значения от -20 до +30.

Задание 11: Найти в температурном файле самую высокую и низкую температуру.

Задание 12: Создать файл с паролями (10 строк по 20 символов), затем найти пароли, содержащие цифры.

ПЗ №15. Организация процедур.

Тема: Организация процедур

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться создавать и использовать процедуры (функции типа void) в C++.
2. Освоить передачу параметров в процедуры по значению, по ссылке и через указатели.
3. Научиться организовывать программу с помощью процедур.

Задачи:

- Создавать процедуры без параметров и с параметрами.
- Использовать процедуры для вывода информации на экран.
- Применять процедуры для модификации данных.
- Организовывать меню программы с помощью процедур.

2. Теоретические сведения

Процедура — это функция типа void, которая не возвращает значение, но выполняет определенные действия.

Структура процедуры:

```
void имя_процедуры(параметры) {  
    // тело процедуры  
    // выполняемые действия  
}
```

Особенности процедур:

1. Не имеют возвращаемого значения (используется void)
2. Выполняют действия: вывод на экран, изменение переменных, расчеты
3. Могут изменять переданные параметры (при передаче по ссылке)
4. Могут вызывать другие процедуры

Типы передачи параметров в процедуры:

1. По значению (создается копия):

```
void printNumber(int x) {  
    cout << x;  
    // x внутри процедуры - копия  
}
```

2. По ссылке (работа с оригиналом):

```
void changeNumber(int &x) {  
    x = x * 2; // изменяет оригинал  
}
```

3. Через указатель:

```
void swapNumbers(int *a, int *b) {  
    int temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

3. Контрольные вопросы

1. Что такое процедура в C++?
2. Чем процедура отличается от функции?
3. Какой тип возвращаемого значения у процедуры?
4. Что означает void в объявлении процедуры?
5. Какие способы передачи параметров в процедуры вы знаете?
6. В чем разница между передачей параметра по значению и по ссылке?
7. Можно ли изменить значение переменной в процедуре при передаче по значению?

8. Как передать массив в процедуру?
9. Можно ли вызвать процедуру внутри другой процедуры?
10. Зачем нужны процедуры в программировании?

4. Задания к практической работе (10 задач)

Общие требования:

- Все задачи решать ТОЛЬКО с использованием процедур (тип void)
- В процедурах не использовать return для возврата значений
- В main() только вызовы процедур и минимальный код
- Каждую задачу оформить как отдельную программу

Задание 1: Создать процедуру drawLine, которая выводит линию из 30 символов '-'. Вызвать ее 5 раз в программе.

Задание 2: Написать процедуру showStudent, которая выводит информацию о студенте:

```
struct Student {  
    char name[30];  
    int group;  
    float averageGrade;  
};
```

Процедура должна принимать структуру Student и выводить все поля.

Задание 3: Создать процедуру calculateAndShow, которая принимает два числа и выводит:

- Их сумму
- Их разность
- Их произведение
- Их частное (с проверкой деления на ноль)

Задание 4: Написать процедуру sortArray, которая принимает массив целых чисел и его размер, и сортирует массив по возрастанию.

Задание 5: Создать процедуру findMinMax, которая находит и выводит минимальный и максимальный элемент в массиве.

Задание 6: Написать процедуру convertTemperature, которая конвертирует температуру из Цельсия в Фаренгейты и выводит оба значения.

Задание 7: Создать процедуру calculateSalary, которая рассчитывает и выводит зарплату сотрудника:

- Основной оклад
- Премия (15% от оклада)
- Налог (13% от суммы)
- Итоговая зарплата

Задание 8: Написать процедуру printCalendar, которая выводит календарь на месяц (только числа от 1 до 31 в 7 колонок).

Задание 9: Создать процедуру processOrder, которая обрабатывает заказ в кафе:

- Принимает количество гамбургеров, картошки и напитков
- Выводит чек с ценами и суммой
- Цены: гамбургер - 150р, картошка - 80р, напиток - 60р

Задание 10: Написать процедуру showMenu для программы учета студентов, которая выводит меню:

1. Добавить студента
2. Удалить студента
3. Найти студента
4. Показать всех
5. Выход

Критерии оценки:

- **Задачи 1-3:** Базовый уровень (3 балла)
- **Задачи 4-7:** Средний уровень (4 балла)
- **Задачи 8-10:** Продвинутый уровень (5 баллов)

ПЗ №16. Организация функций.

1. Цели и задачи практической работы

Цели:

1. Научиться создавать и использовать функции в C++.
2. Освоить передачу параметров в функции.
3. Научиться разбивать программу на логические блоки.

Задачи:

- Создавать функции с параметрами и без.
- Использовать функции для решения задач.
- Передавать данные в функции разными способами.

2. Теоретические сведения

Функция — это блок кода, который выполняет конкретную задачу.

Как создать функцию:

// 1. Объявление (прототип)

тип имя_функции(параметры);

// 2. Определение

```
тип имя_функции(параметры) {  
    // код функции  
    return значение; // если не void  
}
```

// 3. Вызов

имя_функции(аргументы);

Пример:

```
#include <iostream>
```

```
using namespace std;
```

// Функция приветствия

```
void sayHello() {  
    cout << "Привет!" << endl;  
}
```

// Функция сложения

```
int sum(int a, int b) {  
    return a + b;  
}
```

```
int main() {  
    sayHello(); // Вызов  
    int result = sum(5, 3); // Вызов с параметрами  
    cout << "Сумма: " << result << endl;  
    return 0;  
}
```

3. Контрольные вопросы

1. Что такое функция в программировании?
2. Как создать простую функцию?
3. Чем отличается void функция от функции с возвращаемым значением?
4. Как передать массив в функцию?
5. Что такое параметры и аргументы функции?
6. Можно ли вызвать функцию внутри другой функции?

7. Зачем нужны функции?
8. Сколько значений может возвращать функция?
9. Что выведет программа, если забыть return?
10. Можно ли создать функцию с именем main?

4. Задания к практической работе (10 задач)

Общие требования:

- Для каждой задачи создать отдельную функцию
- В main только вызов функций и ввод/вывод
- Проверить работу с разными данными

Задание 1: Создать функцию printStars, которая выводит на экран 10 звездочек (*). Вызвать ее 3 раза.

Задание 2: Написать функцию maxNumber, которая принимает два целых числа и возвращает наибольшее.

Задание 3: Создать функцию calculateCircle, которая принимает радиус и выводит:

- Длину окружности: $2 * \pi * R$
- Площадь круга: $\pi * R^2$ ($\pi = 3.14159$)

Задание 4: Написать функцию isPositive, которая проверяет, положительное ли число. Возвращает true или false.

Задание 5: Создать функцию sumArray, которая принимает массив и его размер, возвращает сумму всех элементов.

Задание 6: Написать функцию convertCurrency, которая переводит рубли в доллары (курс задать как параметр).

Задание 7: Создать функцию printTable, которая выводит таблицу умножения для числа от 1 до 10.

Задание 8: Написать функцию findStudent, которая ищет студента по фамилии в массиве структур:

```
struct Student {
    char surname[30];
    int grade;
};
```

Задание 9: Создать функцию calculateDiscount, которая рассчитывает скидку:

- Если сумма покупки > 10000, скидка 10%
- Если > 5000, скидка 5%
- Иначе без скидки

Задание 10: Написать функцию sortGrades, которая сортирует массив оценок по убыванию.

Критерии оценки:

Критерий	Баллы	Что проверяем
1. Синтаксис	30	Правильное объявление и вызов функций
2. Логика	30	Функции выполняют требуемую задачу
3. Параметры	20	Правильная передача данных в функции
4. Тестирование	10	Проверка с разными входными данными
5. Оформление	10	Комментарии, отступы, названия

Оценки:

- **5 (отлично):** 90-100 баллов - выполнено 8-10 задач
- **4 (хорошо):** 75-89 баллов - выполнено 6-7 задач
- **3 (удовл.):** 60-74 балла - выполнено 4-5 задач
- **2 (неуд.):** менее 60 баллов - менее 4 задач

ПЗ №17. Применение рекурсивных функций.

Тема: Применение рекурсивных функций

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Понять принцип работы рекурсивных функций.
2. Научиться создавать простые рекурсивные функции.
3. Освоить решение задач с помощью рекурсии.

Задачи:

- Изучить структуру рекурсивной функции.
- Научиться определять базовый случай рекурсии.
- Решать математические задачи с помощью рекурсии.
- Понимать ограничения рекурсии.

2. Теоретические сведения

Рекурсивная функция — это функция, которая вызывает сама себя для решения задачи.

Структура рекурсивной функции:

```
тип функция(параметры) {  
    if (базовый случай) {  
        return результат; // Остановка рекурсии  
    } else {  
        return функция(измененные параметры); // Рекурсивный вызов  
    }  
}
```

Важные понятия:

1. **Базовый случай** — условие, при котором рекурсия останавливается
2. **Рекурсивный случай** — вызов функции с измененными параметрами
3. **Глубина рекурсии** — количество вложенных вызовов
4. **Стек вызовов** — память для хранения вызовов функций

Пример простой рекурсии:

```
#include <iostream>  
using namespace std;  
  
void countdown(int n) {  
    if (n == 0) { // Базовый случай  
        cout << "Старт!" << endl;  
    } else {  
        cout << n << "... ";  
        countdown(n - 1); // Рекурсивный вызов  
    }  
}  
  
int main() {  
    countdown(5); // 5... 4... 3... 2... 1... Старт!  
    return 0;  
}
```

Внимание! Рекурсия должна всегда иметь базовый случай, иначе будет бесконечный цикл.

3. Контрольные вопросы

1. Что такое рекурсивная функция?
2. Чем рекурсивная функция отличается от обычной?
3. Что такое базовый случай в рекурсии?
4. Почему важно определять базовый случай?
5. Что произойдет, если забыть базовый случай?
6. Что такое глубина рекурсии?
7. Какой максимальной глубины рекурсии можно достичь?
8. В чем недостатки рекурсии?
9. Когда рекурсия полезна, а когда нет?
10. Как работает стек вызовов при рекурсии?

4. Задания к практической работе (10 задач)

Общие требования:

- Все задачи решать с помощью рекурсивных функций
- Обязательно определять базовый случай
- Проверять корректность входных данных
- Избегать слишком глубокой рекурсии

Задание 1: Написать рекурсивную функцию для вычисления суммы цифр числа.

```
int sumDigits(int number) {  
    // Например: sumDigits(123) = 1+2+3 = 6  
}
```

Задание 2: Создать рекурсивную функцию для возведения числа в степень.

```
double power(double base, int exponent) {  
    // Например: power(2, 3) = 2*2*2 = 8  
}
```

Задание 3: Написать рекурсивную функцию для подсчета количества цифр в числе.

```
int countDigits(int number) {  
    // Например: countDigits(12345) = 5  
}
```

Задание 4: Создать рекурсивную функцию для вычисления НОД (наибольшего общего делителя) двух чисел.

```
int gcd(int a, int b) {  
    // Алгоритм Евклида  
}
```

Задание 5: Написать рекурсивную функцию для проверки, является ли число простым.

```
bool isPrime(int n, int divisor = 2) {  
    // Число простое, если делится только на 1 и само себя  
}
```

Задание 6: Создать рекурсивную функцию для вывода всех чисел от A до B.

```
void printRange(int a, int b) {  
    // Например: printRange(3, 7) выводит 3 4 5 6 7  
}
```

Задание 7: Написать рекурсивную функцию для реверса (переворачивания) числа.

```
int reverseNumber(int n, int reversed = 0) {
```

```
    // Например: reverseNumber(123) = 321
}
```

Задание 8: Создать рекурсивную функцию для проверки, является ли строка палиндромом.

```
bool isPalindrome(char str[], int start, int end) {
    // Например: "казак" - палиндром
}
```

Задание 9: Написать рекурсивную функцию для вычисления биномиального коэффициента $C(n, k)$.

```
int binomial(int n, int k) {
    //  $C(n, k) = C(n-1, k-1) + C(n-1, k)$ 
    //  $C(n, 0) = C(n, n) = 1$ 
}
```

Задание 10: Создать рекурсивную функцию для нахождения максимального элемента в массиве.

```
int findMax(int arr[], int size) {
    // Сравниваем первый элемент с максимумом остального массива
}
```

Как проверить рекурсивную функцию:

1. Проверьте базовый случай
2. Проверьте, что параметры изменяются при рекурсивном вызове
3. Протестируйте на малых значениях
4. Протестируйте на граничных случаях

Советы:

- Начните с самых простых задач (1-3)
- Используйте отладчик для пошагового выполнения
- Рисуите схему вызовов на бумаге для сложных задач
- Помните, что рекурсия не всегда лучше цикла!

Эта практическая работа поможет понять, когда и как использовать рекурсию, и какие задачи она помогает решить эффективно.

ПЗ №18. Программирование модуля.

Тема: Программирование модуля

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи

Цель: Научиться создавать программы из нескольких файлов

Задачи:

- Разделять код на логические модули
- Создавать заголовочные файлы
- Работать с проектами в среде разработки

2. Теоретические сведения

Структура проекта:

main. - основная программа

calculator.h - объявления функций

calculator. - реализация функций

calculator.h:

```
#ifndef CALCULATOR_H
#define CALCULATOR_H
```

```
int add(int a, int b);
int subtract(int a, int b);
float divide(int a, int b);
```

```
#endif
```

3. Контрольные вопросы

1. Что хранится в .h файле?
2. Для чего нужна директива #ifndef?
3. Как подключить свой заголовочный файл?
4. Можно ли писать код в .h файле?
5. Как создать новый . файл в проекте?

4. Задания к практической работе (10+ заданий)

Задание 1: Создать проект "Математические функции":

- math.h - объявления
- math. - факториал, степень, НОД
- main. - меню выбора операции

Задание 2: Проект "Работа со строками":

- string_tools.h
- string_tools. (длина, сравнение, поиск)
- main.

Задание 3: Проект "Учет книг":

- book.h - структура книги
- library. - функции работы с массивом книг
- main. - интерфейс

Задание 4: Проект "Геометрические фигуры":

- geometry.h - объявления
- geometry. - площади и периметры
- main. - выбор фигуры

Задание 5: Проект "Конвертер валют":

- converter.h
- converter. (рубль-доллары, рубль-евро)

- main.

Задание 6: Проект "Шифрование":

- crypto.h
- crypto. (шифр Цезаря)
- main.

Задание 7: Проект "Работа с датой":

- date.h - структура даты
- date. - проверка корректности, вывод
- main.

Задание 8: Проект "Статистика":

- stats.h
- stats. (среднее, максимум, минимум массива)
- main.

Задание 9: Проект "База данных студентов" (3 файла):

- Файл 1: структуры
- Файл 2: функции ввода/вывода
- Файл 3: главное меню

Задание 10: Проект "Игра Угадай число":

- game.h - константы
- game. - логика игры
- main. - запуск

Задание 11: Проект "Заметки":

- note.h - структура заметки
- notes. - сохранение/загрузка из файла
- main. - интерфейс

Задание 12: Проект "Тестирование" (опросник):

- question.h - структура вопроса
- test. - загрузка вопросов, проверка
- main. - проведение теста

ПЗ №19. Создание библиотеки подпрограмм

Тема: Создание библиотеки подпрограмм

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться создавать и использовать библиотеки подпрограмм.
2. Освоить организацию кода в многофайловых проектах.
3. Понять принципы повторного использования кода.

Задачи:

- Создать заголовочный файл (.h) с объявлениями функций.
- Создать файл реализации (.) с определениями функций.
- Научиться подключать собственную библиотеку к программе.
- Организовать проект из нескольких файлов.

2. Теоретические сведения

Библиотека подпрограмм - это набор функций, объединенных в отдельные файлы для многократного использования.

Структура библиотеки:

text

my_library/

```
├── mylib.h    ← заголовочный файл (объявления)
├── mylib.    ← файл реализации (определения)
└── main.     ← основная программа
```

Заголовочный файл (.h):

```
#ifndef MYLIB_H    // Защита от повторного включения
#define MYLIB_H
```

```
// Объявления функций
```

```
void printHello();
int sum(int a, int b);
```

```
#endif
```

Файл реализации (.):

```
#include "mylib.h"
#include <iostream>
using namespace std;
```

```
// Определения функций
```

```
void printHello() {
    cout << "Hello from library!" << endl;
}
```

```
int sum(int a, int b) {
    return a + b;
}
```

Основная программа:

```
#include "mylib.h"
#include <iostream>
```

```
using namespace std;
```

```
int main() {  
    printHello();  
    cout << "Sum: " << sum(5, 3) << endl;  
    return 0;  
}
```

Как собрать проект в Code::Blocks:

1. Создать новый проект
2. Добавить все три файла в проект
3. Нажать Build → Build and Run

3. Контрольные вопросы

1. Что такое заголовочный файл (.h)?
2. Для чего нужна директива `#ifndef` в заголовочном файле?
3. Чем отличается объявление функции от ее определения?
4. Как подключить свою библиотеку к программе?
5. Почему нельзя определять функции в заголовочном файле?
6. Как правильно называть файлы библиотеки?
7. Что будет, если подключить один заголовочный файл дважды?
8. Как организовать проект из нескольких файлов в Code::Blocks?
9. Можно ли создать несколько библиотек в одном проекте?
10. Зачем нужны библиотеки подпрограмм?

4. Задания к практической работе (10 задач)

Общие требования:

- Каждую задачу оформить как отдельный проект
- Создать заголовочный файл и файл реализации
- Написать основную программу для тестирования
- Проверить работу всех функций

Задание 1: Создать библиотеку `baselib` с базовыми функциями:

- `printLine()` - выводит линию из 20 символов '-'
- `printSquare(int n)` - выводит квадрат числа
- `isEven(int n)` - проверяет четность числа
- `absValue(int n)` - возвращает абсолютное значение

Задание 2: Создать библиотеку `converter` для конвертации величин:

- `celsiusToFahrenheit(float c)` - Цельсий → Фаренгейт
- `fahrenheitToCelsius(float f)` - Фаренгейт → Цельсий
- `rublesToDollars(float rub, float rate)` - рубли → доллары
- `dollarsToRubles(float usd, float rate)` - доллары → рубли

Задание 3: Создать библиотеку `geometry` для геометрических расчетов:

- `circleArea(float r)` - площадь круга
- `circlePerimeter(float r)` - длина окружности
- `rectangleArea(float a, float b)` - площадь прямоугольника
- `triangleArea(float base, float height)` - площадь треугольника

Задание 4: Создать библиотеку `studentlib` для работы со студентами:

```
struct Student {  
    char name[30];  
    int group;  
    float grades[5];  
};
```

Функции:

- `inputStudent(Student &s)` - ввод данных
- `printStudent(Student s)` - вывод данных
- `calculateAverage(Student s)` - средний балл

Задание 5: Создать библиотеку `arraytools` для работы с массивами:

- `fillRandom(int arr[], int size, int min, int max)` - заполнение случайными
- `bubbleSort(int arr[], int size)` - сортировка пузырьком
- `linearSearch(int arr[], int size, int value)` - линейный поиск
- `countEven(int arr[], int size)` - количество четных

Задание 6: Создать библиотеку `timelib` для работы со временем:

```
struct Time {  
    int hours;  
    int minutes;  
    int seconds;  
};
```

Функции:

- `addTime(Time t1, Time t2)` - сложение времени
- `timeToSeconds(Time t)` - перевод в секунды
- `secondsToTime(int sec)` - перевод из секунд
- `printTime(Time t)` - вывод времени

Задание 7: Создать библиотеку `calculator` для различных расчетов:

- `calculateDiscount(float price, float percent)` - расчет скидки
- `calculateTax(float income, float rate)` - расчет налога
- `compoundInterest(float principal, float rate, int years)` - сложный процент
- `loanPayment(float amount, float rate, int months)` - платеж по кредиту

Задание 8: Создать библиотеку `stringoperations`:

- `stringLength(char str[])` - длина строки
- `stringCopy(char dest[], char src[])` - копирование
- `stringConcatenate(char str1[], char str2[])` - конкатенация
- `findSubstring(char str[], char sub[])` - поиск подстроки

Задание 9: Создать библиотеку `filelib` для работы с файлами:

- `createFile(char filename[])` - создание файла
- `writeToFile(char filename[], char text[])` - запись в файл
- `readFromFile(char filename[])` - чтение из файла
- `countLines(char filename[])` - подсчет строк

Задание 10: Создать библиотеку `gametools` для простых игр:

- `generateRandomNumber(int min, int max)` - случайное число
- `checkGuess(int secret, int guess)` - проверка догадки
- `calculateScore(int attempts, int maxAttempts)` - расчет очков
- `printHangman(int mistakes)` - вывод виселицы

Критерии оценки:

1. Правильная структура файлов - 30%
2. Работоспособность всех функций - 40%
3. Корректное подключение библиотеки - 20%
4. Качество кода и комментарии - 10%

Эта работа поможет освоить организацию кода в реальных проектах!

ПЗ №20. Использование указателей для организации связанных списков. Создание и удаление динамических переменных

Тема: Использование указателей для организации связанных списков. Создание и удаление динамических переменных

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Понять, что такое указатели и зачем они нужны.
2. Научиться работать с динамической памятью.
3. Создавать простые динамические структуры.

Задачи:

- Создавать динамические переменные.
- Работать с динамическими массивами.
- Правильно освобождать память.
- Избегать утечек памяти.

2. Теоретические сведения

Указатель - переменная, которая хранит адрес другой переменной.

```
int x = 10;    // обычная переменная
int* p = &x;   // указатель на x
cout << *p;    // получаем значение через указатель
```

Динамическая память:

// Выделение памяти для одного числа

```
int* num = new int;
```

// Выделение памяти для массива

```
int* arr = new int[5];
```

// Освобождение памяти

```
delete num;
```

```
delete[] arr;
```

Важно: Всегда освобождайте память, которую выделили!

3. Контрольные вопросы

1. Что такое указатель?
2. Как получить адрес переменной?
3. Как получить значение через указатель?
4. Что делает оператор new?
5. Что делает оператор delete?
6. Чем отличается delete от delete[]?
7. Что такое динамическая память?
8. Почему важно освобождать память?
9. Что будет, если не освободить память?
10. Как создать динамический массив?

4. Задания к практической работе (10 задач)

Задание 1: Создать динамическую переменную для возраста человека. Запросить возраст, вывести его дважды (обычным способом и через указатель), освободить память.

Задание 2: Создать динамическую переменную для цены товара. Запросить цену, увеличить ее на 10% через указатель, вывести новую цену.

Задание 3: Создать динамический массив для 3 оценок студента. Заполнить, найти средний балл.

Задание 4: Создать динамический массив для N чисел (N вводит пользователь). Заполнить, найти сумму всех чисел.

Задание 5: Создать динамический массив температур за неделю (7 дней). Заполнить случайными значениями от -10 до +10, найти самую высокую температуру.

Задание 6: Создать структуру Book в динамической памяти:

```
struct Book {  
    char title[50];  
    char author[50];  
    int year;  
};
```

Заполнить и вывести информацию о книге.

Задание 7: Создать динамический массив для хранения 5 товаров. Каждый товар имеет название и цену. Вывести самый дорогой товар.

Задание 8: Создать программу для учета студентов. Использовать динамический массив структур Student (имя, группа, оценка). Реализовать:

- Добавление студента
- Вывод всех студентов
- Поиск студента по имени

Задание 9: Создать два динамических массива одинакового размера. Заполнить их, создать третий массив, который содержит сумму соответствующих элементов первых двух.

Задание 10: Создать динамическую квадратную матрицу N×N. Заполнить ее, вывести на экран, найти сумму диагональных элементов.

Критерии оценки:

1. Программа компилируется - 30%
2. Правильно работает - 40%
3. Нет утечек памяти - 20%
4. Есть комментарии - 10%

ПЗ №25. События компонентов (элементов управления), их сущность и назначение.

Тема: События компонентов (элементов управления), их сущность и назначение.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Понять, что такое события в программировании и как они работают.
2. Научиться обрабатывать события элементов управления.
3. Закрепить навыки работы с динамической памятью.

Задачи:

- Изучить основные события компонентов (кнопки, поля ввода).
- Научиться создавать обработчики событий.
- Практиковаться в создании и удалении динамических переменных.
- Создать простые программы с графическим интерфейсом.

2. Теоретические сведения

Событие - это действие пользователя или системы, на которое программа может реагировать.

Основные события в C++/QT:

1. **clicked()** - нажатие на кнопку
2. **textChanged()** - изменение текста в поле
3. **valueChanged()** - изменение значения
4. **pressed()** / **released()** - нажатие/отпускание кнопки

Создание обработчика события:

// В заголовочном файле

```
private slots:  
    void on_button_clicked();
```

// В файле реализации

```
void MyClass::on_button_clicked() {  
    // Действия при нажатии кнопки  
}
```

Связь события с обработчиком:

```
connect(кнопка, SIGNAL(clicked()), this, SLOT(обработчик()));
```

Динамическая память (повторение):

// Создание

```
int* число = new int;  
string* текст = new string;
```

// Удаление

```
delete число;  
delete текст;
```

3. Контрольные вопросы

1. Что такое событие в программировании?
2. Какие основные события есть у кнопки?
3. Как создать обработчик события?
4. Что делает функция connect()?
5. Как обработать нажатие кнопки?
6. Как обработать изменение текста в поле ввода?

7. Как создать динамическую переменную?
8. Как освободить динамическую память?
9. Что произойдет, если не освободить память?
10. Как связать несколько событий с разными обработчиками?

4. Задания к практической работе (10 задач)

Задание 1: Создать программу с кнопкой, которая при нажатии выводит сообщение "Привет, мир!" в консоль.

Задание 2: Создать программу с полем ввода и кнопкой. При нажатии кнопки выводить текст из поля в консоль.

Задание 3: Создать счетчик нажатий. Кнопка увеличивает счетчик на 1 и показывает текущее значение.

Задание 4: Создать программу с двумя кнопками: "Увеличить" и "Уменьшить". Они изменяют значение счетчика.

Задание 5: Создать программу для ввода имени. При нажатии кнопки выводить приветствие: "Привет, [имя]!".

Задание 6: Создать калькулятор с двумя полями ввода и кнопками операций (+, -, *, /). Результат выводить в третье поле.

Задание 7: Создать программу для изменения цвета фона по нажатию кнопки.

Задание 8: Создать программу с динамическим массивом для хранения списка имен. Добавлять имена по нажатию кнопки.

Задание 9: Создать программу для конвертации валют (рубли в доллары). Использовать динамическую переменную для курса.

Задание 10: Создать программу учета студентов с динамическим массивом. Добавлять студентов по нажатию кнопки.

Критерии оценки:

1. Программа компилируется и работает - 40%
2. Правильная обработка "событий" - 25%
3. Корректная работа с динамической памятью - 25%
4. Комментарии и оформление - 10%

ПЗ №26. Создание процедур на основе событий.

Тема: Создание процедур на основе событий.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться создавать процедуры, которые вызываются при наступлении событий.
2. Закрепить навыки работы с динамической памятью в контексте событий.
3. Освоить организацию кода с использованием событийно-ориентированного подхода.

Задачи:

- Создавать процедуры-обработчики событий.
- Связывать события с процедурами.
- Управлять динамической памятью в обработчиках событий.
- Реализовывать практические задачи с использованием событий.

2. Теоретические сведения

Процедура-обработчик события - это процедура, которая вызывается автоматически при наступлении определенного события.

Структура в Qt:

```
class MyWidget : public QWidget {
    Q_OBJECT

private slots: // Процедуры-обработчики
    void on_button_clicked(); // при нажатии кнопки
    void on_text_changed(); // при изменении текста
};
```

```
// Определение процедуры
void MyWidget::on_button_clicked() {
    // Действия при нажатии
}
```

Связывание событий с процедурами:

```
// Способ 1: auto-connect (сигнатура on_объект_событие)
connect(button, SIGNAL(clicked()), this, SLOT(on_button_clicked()));
```

```
// Способ 2: lambda-выражение
connect(button, &QPushButton::clicked, [=]() {
    // код обработки
});
```

Динамическая память в обработчиках:

```
void on_button_clicked() {
    // Создание динамической переменной
    int* data = new int(100);

    // Использование...

    // Важно: освободить память, если она больше не нужна
    delete data;
}
```

Основные события в Qt:

- `clicked()` - нажатие кнопки
- `textChanged()` - изменение текста
- `valueChanged()` - изменение значения
- `pressed()`, `released()` - нажатие/отпускание
- `returnPressed()` - нажатие Enter

3. Контрольные вопросы

1. Что такое процедура-обработчик события?
2. Как связать событие с процедурой в Qt?
3. Какие секции используются для объявления обработчиков в классе Qt?
4. Как создать динамическую переменную в обработчике события?
5. Почему важно освобождать динамическую память в деструкторе?
6. Как обработать событие изменения текста в `QLineEdit`?
7. Можно ли одной процедурой обрабатывать несколько разных событий?
8. Что такое слот (slot) в Qt?
9. Как передать параметры в процедуру-обработчик?
10. Что делать, если нужно создать много динамических объектов?

4. Задания к практической работе (10 задач)

Задание 1: Создать программу с кнопкой, которая при нажатии создает динамическую переменную-счетчик и увеличивает ее значение. Отображать текущее значение на форме.

Задание 2: Создать форму с полем ввода и кнопкой. При нажатии кнопки создавать динамическую строку с текстом из поля и выводить ее в `QLabel`.

Задание 3: Создать калькулятор с двумя полями ввода. При нажатии кнопок "+", "-", "*", "/" выполнять соответствующую операцию над динамически созданными переменными.

Задание 4: Создать программу для ввода списка покупок. По кнопке "Добавить" создавать динамический объект покупки и добавлять в список.

Задание 5: Создать форму смены цвета фона. Три кнопки "Красный", "Зеленый", "Синий" изменяют цвет фона, используя динамически созданные объекты `QColor`.

Задание 6: Создать программу для конвертации валют. При изменении текста в поле "Курс" создавать динамическую переменную с курсом и пересчитывать сумму.

Задание 7: Создать простое текстовое приложение. Кнопка "Сохранить" сохраняет текст в динамически созданный файловый объект, кнопка "Загрузить" - загружает.

Задание 8: Создать программу учета времени. Кнопка "Старт" создает динамическую переменную времени начала, кнопка "Стоп" - вычисляет разницу.

Задание 9: Создать форму для работы с динамическим массивом чисел. Кнопки: "Добавить число", "Удалить число", "Найти сумму", "Найти среднее".

Задание 10: Создать программу для работы с базой студентов. Использовать динамический массив структур. Кнопки: "Добавить студента", "Удалить студента", "Показать всех".

Критерии оценки:

1. Правильная структура процедур-обработчиков - 30%
2. Корректная работа с динамической памятью - 30%
3. Функциональность программы - 25%
4. Оформление кода и комментарии - 15%

ПЗ №27. Создание проекта с использованием кнопочных компонентов.

Тема: Создание проекта с использованием кнопочных компонентов.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться создавать графические интерфейсы с использованием кнопок в Qt.
2. Освоить основные свойства и методы кнопочных компонентов.
3. Научиться организовывать взаимодействие пользователя с программой через кнопки.

Задачи:

- Создавать различные типы кнопок.
- Настраивать внешний вид кнопок.
- Привязывать обработчики к событиям кнопок.
- Создавать практические приложения с кнопочным управлением.

2. Теоретические сведения

Основные кнопочные компоненты в Qt:

1. **QPushButton** - обычная кнопка
2. **QToolButton** - кнопка с иконкой
3. **QRadioButton** - переключатель (выбор одного из нескольких)
4. **QCheckBox** - флажок (множественный выбор)

Создание кнопки:

// Простая кнопка

```
QPushButton *button = new QPushButton("Текст кнопки", родитель);
```

// Кнопка с иконкой

```
QPushButton *buttonWithIcon = new QPushButton(  
    QIcon("путь/к/иконке.png"),  
    "Текст",  
    родитель  
);
```

Основные свойства кнопок:

```
button->setText("Новый текст"); // Установка текста  
button->setEnabled(false);      // Блокировка кнопки  
button->setVisible(false);      // Скрытие кнопки  
button->setFixedSize(100, 50);  // Фиксированный размер
```

События кнопок:

// Основные события

```
clicked()    // Нажатие и отпускание  
pressed()   // Нажатие (удержание)  
released()  // Отпускание  
toggled(bool) // Изменение состояния (для переключателей)
```

Связывание событий с обработчиками:

// Способ 1: Слоты

```
connect(button, SIGNAL(clicked()), this, SLOT(название_слота()));
```

// Способ 2: Lambda-выражение

```
connect(button, &QPushButton::clicked, [=]() {
```

// Действия при нажатии

});

3. Контрольные вопросы

1. Какие основные типы кнопок есть в Qt?
2. Как создать кнопку с текстом?
3. Как создать кнопку с иконкой?
4. Какие основные события есть у кнопки?
5. Как заблокировать кнопку?
6. Как скрыть кнопку?
7. Как изменить размер кнопки?
8. Как изменить текст кнопки?
9. Чем отличается QRadioButton от QCheckBox?
10. Как обработать нажатие кнопки?

4. Задания к практической работе (10 задач)

Задание 1: Создать программу с одной кнопкой. При нажатии на кнопку меняется ее текст (например, с "Нажми меня" на "Спасибо!").

Задание 2: Создать программу с тремя кнопками разных цветов. При нажатии на кнопку цвет фона окна меняется на цвет кнопки.

Задание 3: Создать простой калькулятор с кнопками цифр 0-9 и кнопками операций +, -, *, /. Результат выводить в поле над кнопками.

Задание 4: Создать программу для выбора любимого цвета с помощью QRadioButton. При нажатии кнопки "Выбрать" показывать выбранный цвет.

Задание 5: Создать форму регистрации с кнопками: "Зарегистрироваться", "Очистить", "Выйти". Проверять заполнение полей.

Задание 6: Создать программу-таймер с кнопками "Старт", "Стоп", "Сброс". Отображать время в секундах.

Задание 7: Создать программу для работы со списком дел. Кнопки: "Добавить", "Удалить", "Отметить выполненным".

Задание 8: Создать программу для конвертации единиц измерения (например, см в дюймы). Кнопки для разных направлений конвертации.

Задание 9: Создать простую игру "Угадай число" с кнопками "Новая игра", "Проверить", "Подсказка".

Задание 10: Создать программу для учета расходов с кнопками: "Добавить расход", "Показать статистику", "Очистить историю".

Критерии оценки:

1. Правильное создание и настройка кнопок - 30%
2. Работоспособность обработчиков событий - 30%
3. Логика и полезность приложения - 25%
4. Внешний вид и удобство интерфейса - 15%

ПЗ №28. Создание проекта с использованием кнопочных компонентов.

Тема: Создание проекта с использованием кнопочных компонентов.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться создавать простые программы с кнопками в Qt.
2. Понять, как кнопки реагируют на нажатия.
3. Создать свои первые графические программы.

Задачи:

- Добавлять кнопки в окно программы.
- Делать так, чтобы кнопки что-то делали при нажатии.
- Менять внешний вид кнопок.
- Создавать полезные программы с кнопками.

2. Теоретические сведения

Как создать кнопку в Qt:

// Простая кнопка

```
QPushButton *button = new QPushButton("Моя кнопка", this);
```

// Кнопка с иконкой (если нужна картинка)

```
QPushButton *button2 = new QPushButton(this);
```

```
button2->setText("Кнопка 2");
```

Что делает кнопка при нажатии:

// Подключаем действие к кнопке

```
connect(button, SIGNAL(clicked()), this, SLOT(название_функции()));
```

// Или проще:

```
connect(button, &QPushButton::clicked, [=]() {
```

```
    // Здесь пишем, что должно произойти
```

```
    cout << "Кнопка нажата!" << endl;
```

```
});
```

Как изменить кнопку:

```
button->setText("Новый текст"); // Меняем надпись
```

```
button->setEnabled(false); // Делаем кнопку серой (нельзя нажать)
```

```
button->setStyleSheet("background: red;"); // Меняем цвет
```

3. Контрольные вопросы

1. Как создать кнопку в Qt?
2. Как сделать, чтобы кнопка что-то делала при нажатии?
3. Как изменить текст на кнопке?
4. Как сделать кнопку неактивной (серой)?
5. Как добавить несколько кнопок в окно?
6. Что такое Layout и зачем он нужен?

4. Задания к практической работе (6 задач)

Задание 1: Создать программу с кнопкой, которая при нажатии показывает сообщение "Привет!" в отдельном окошке.

Задание 2: Создать программу с двумя кнопками: "Показать" и "Скрыть". При нажатии на "Показать" появляется текст "Вот я!", при нажатии на "Скрыть" - текст исчезает.

Задание 3: Создать простой калькулятор с кнопками для сложения. Два поля для ввода чисел, кнопка "=", показывает сумму.

Задание 4: Создать программу для выбора цвета. Три кнопки: "Красный", "Зеленый", "Синий". При нажатии цвет фона окна меняется.

Задание 5: Создать программу-таймер. Кнопка "Старт" начинает отсчет секунд, кнопка "Стоп" останавливает. Показывать прошедшее время.

Задание 6: Создать программу для списка дел. Поле для ввода дела, кнопка "Добавить" добавляет дело в список, кнопка "Удалить" удаляет выбранное дело.

Что проверяет преподаватель:

1. Работает ли программа ✓
2. Нажимаются ли кнопки ✓
3. Делают ли кнопки то, что нужно ✓
4. Выглядит ли программа аккуратно ✓

ПЗ №29. Создание проекта с использованием компонентов стандартных диалогов и системы меню.

Тема: Создание проекта с использованием компонентов стандартных диалогов и системы меню.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться создавать простые меню в программах.
2. Понять, как показывать сообщения пользователю.
3. Сделать программы удобнее с помощью диалоговых окон.

Задачи:

- Добавить меню "Файл" и "Справка" в программу.
- Научиться показывать сообщения об ошибках, предупреждения, вопросы.
- Открывать диалоги выбора файлов и цветов.
- Создать простые, но полезные программы.

2. Теоретические сведения

Как показать сообщение:

// Простое сообщение

```
QMessageBox::information(this, "Заголовок", "Текст сообщения");
```

// Спросить что-то

```
int answer = QMessageBox::question(this, "Вопрос", "Вы уверены?");
```

```
if (answer == QMessageBox::Yes) {
```

```
    // Пользователь сказал "Да"
```

```
}
```

// Предупреждение

```
QMessageBox::warning(this, "Внимание", "Будьте осторожны!");
```

// Ошибка

```
QMessageBox::critical(this, "Ошибка", "Что-то сломалось!");
```

Как создать меню:

// В конструкторе класса, наследующего QMainWindow

```
QMenu *fileMenu = menuBar()->addMenu("Файл");
```

```
QAction *exitAction = fileMenu->addAction("Выход");
```

```
connect(exitAction, &QAction::triggered, this, &QMainWindow::close);
```

Как выбрать файл:

```
QString fileName = QFileDialog::getOpenFileName(this,
```

```
    "Открыть файл", "", "Текстовые файлы (*.txt)");
```

```
if (!fileName.isEmpty()) {
```

```
    // Файл выбран
```

```
}
```

Как выбрать цвет:

```
QColor color = QColorDialog::getColor(Qt::white, this, "Выберите цвет");
```

```
if (color.isValid()) {
```

```
    // Цвет выбран
```

```
}
```

3. Контрольные вопросы

1. Как показать простое сообщение пользователю?
2. Как спросить у пользователя "Да" или "Нет"?
3. Как создать меню "Файл" с пунктом "Выход"?
4. Как открыть диалог для выбора файла?
5. Как выбрать цвет с помощью диалога?
6. Какие типы сообщений есть в QMessageBox?

4. Задания к практической работе (6 задач)

Задание 1: Создать программу с меню "Справка". При выборе "О программе" показывать окно с информацией о программе.

Задание 2: Создать программу с кнопкой "Спросить". При нажатии задавать вопрос "Вы довольны программой?" и показывать ответ.

Задание 3: Создать программу для выбора цвета фона. Кнопка "Выбрать цвет" открывает диалог выбора цвета.

Задание 4: Создать программу с меню "Файл" → "Открыть". При выборе показывать диалог выбора файла и его имя.

Задание 5: Создать программу-калькулятор с кнопкой "Сброс". При нажатии спрашивать подтверждение.

Задание 6: Создать простой текстовый редактор с меню:

- Файл → Выход
- Правка → Очистить (спрашивает подтверждение)
- Справка → О программе

Что проверяет преподаватель:

1. Есть ли меню в программе ✓
2. Работают ли пункты меню ✓
3. Показываются ли диалоги ✓
4. Выглядит ли программа аккуратно ✓

ПЗ №30. Создание проекта с использованием компонентов стандартных диалогов и системы меню.

Тема: Создание проекта с использованием компонентов стандартных диалогов и системы меню.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться добавлять меню в программы на Qt.
2. Понять, как показывать разные типы сообщений пользователю.
3. Использовать стандартные диалоги для работы с файлами и цветами.

Задачи:

- Создать простое меню "Файл" и "Справка".
- Показать информационное сообщение.
- Открыть диалог выбора файла.
- Выбрать цвет с помощью диалога.

2. Теоретические сведения

Стандартные сообщения:

// Простое сообщение

```
QMessageBox::information(parent, "Заголовок", "Текст");
```

// Вопрос с ответами Да/Нет

```
QMessageBox::question(parent, "Вопрос", "Вы уверены?");
```

// Предупреждение

```
QMessageBox::warning(parent, "Внимание", "Осторожно!");
```

// О программе

```
QMessageBox::about(parent, "О программе", "Информация");
```

Выбор файла:

```
QString fileName = QFileDialog::getOpenFileName(parent,  
    "Выберите файл", "", "Все файлы (*.*)");
```

Выбор цвета:

```
QColor color = QColorDialog::getColor(Qt::white, parent, "Выберите цвет");
```

Создание меню:

// В программе с QMainWindow

```
QMenu *fileMenu = menuBar()->addMenu("Файл");  
QAction *openAction = fileMenu->addAction("Открыть");
```

3. Контрольные вопросы

1. Как показать простое сообщение пользователю?
2. Как создать меню "Файл" с пунктом "Выход"?
3. Как открыть диалог для выбора файла?
4. Как спросить у пользователя "Да" или "Нет"?
5. Как выбрать цвет фона окна?
6. Что делает QMainWindow?

4. Задания к практической работе (6 задач)

Задание 1: Создать программу с меню "Справка". При выборе "О программе" показывать окно с вашим именем и группой.

Задание 2: Создать программу с кнопкой "Спросить". При нажатии задавать вопрос "Вам нравится программирование?" и показывать ответ.

Задание 3: Создать программу для смены цвета фона. Кнопка "Изменить цвет" открывает диалог выбора цвета.

Задание 4: Создать программу с меню "Файл" → "Открыть файл". При выборе показывать диалог и выводить имя выбранного файла.

Задание 5: Создать программу-блокнот с кнопкой "Очистить". При нажатии спрашивать "Очистить весь текст?".

Задание 6: Создать программу калькулятор с меню:

- Файл → Выход
- Операции → Сложить (спрашивает два числа)
- Справка → О программе

Что проверяет преподаватель:

1. Компилируется ли программа ✓
2. Есть ли меню (для заданий 1,4,6) ✓
3. Работают ли диалоги (сообщения, выбор файла/цвета) ✓
4. Программа делает то, что требуется ✓
5. Код аккуратно оформлен ✓

Советы:

- Начинайте с копирования примеров
- Проверяйте каждую часть программы отдельно
- Не бойтесь делать ошибки - на них учатся
- Сохраняйте код часто (Ctrl+S)
- Запускайте программу после каждого изменения (Ctrl+R)

ПЗ №31. Разработка функциональной схемы работы приложения.

Тема: Разработка функциональной схемы работы приложения.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться планировать программу перед написанием кода.
2. Понять, как разбивать задачу на простые части.
3. Создать схему работы программы.

Задачи:

- Нарисовать блок-схему программы.
- Разделить программу на функции.
- Спланировать интерфейс пользователя.
- Создать простую программу по готовой схеме.

2. Теоретические сведения

Функциональная схема - это план работы программы, который показывает:

1. Что программа делает
2. Как она это делает
3. Какие функции используются
4. Как пользователь взаимодействует с программой

Основные элементы схемы:

text

Начало → Ввод данных → Обработка → Вывод результата → Конец

Пример простой схемы для калькулятора:

text

1. Начало программы
2. Показать меню операций
3. Спросить первое число
4. Спросить второе число
5. Спросить операцию (+, -, *, /)
6. Выполнить вычисление
7. Показать результат
8. Спросить "Повторить?"
9. Если ДА → вернуться к шагу 2
10. Если НЕТ → Конец

Как рисовать схему:

- Прямоугольник - действие
- Ромб - решение (да/нет)
- Стрелки - переход к следующему шагу
- Овал - начало/конец

3. Контрольные вопросы

1. Что такое функциональная схема программы?
2. Зачем нужна схема перед написанием кода?
3. Какой значок используется для действия?
4. Какой значок используется для решения (да/нет)?
5. Что показывает стрелка в схеме?
6. Как разделить сложную программу на простые части?

4. Задания к практической работе (6 задач)

Задание 1: Нарисовать схему для программы, которая спрашивает имя и говорит "Привет, [имя]!". Написать программу по схеме.

Задание 2: Нарисовать схему для программы "Проверка пароля". Программа спрашивает пароль, если пароль правильный - показывает "Доступ разрешен", если нет - "Доступ запрещен". Написать программу.

Задание 3: Нарисовать схему для программы "Калькулятор + -". Программа складывает или вычитает два числа по выбору пользователя. Написать программу.

Задание 4: Нарисовать схему для программы "Угадай число". Компьютер загадывает число от 1 до 10, пользователь угадывает. Написать программу.

Задание 5: Нарисовать схему для программы "Список дел". Можно добавлять дела в список и показывать весь список. Написать программу.

Задание 6: Нарисовать схему для программы "Конвертер валют". Пользователь вводит сумму в рублях, программа показывает сумму в долларах (курс задать). Написать программу.

Что проверяет преподаватель:

1. Есть ли схема программы ✓
2. Соответствует ли код схеме ✓
3. Работает ли программа ✓
4. Понятен ли код ✓

ПЗ №32. Разработка оконного приложения с несколькими формами.

Тема: Разработка оконного приложения с несколькими формами.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться создавать программы с несколькими окнами.
2. Понять, как переключаться между окнами.
3. Научиться передавать данные между окнами.

Задачи:

- Создать главное окно с кнопкой для открытия второго окна.
- Создать второе окно с кнопкой для возврата.
- Передавать простые данные между окнами.
- Сделать программу с логином и основным окном.

2. Теоретические сведения

Как создать новое окно:

// 1. Создаем класс для второго окна

```
class SecondWindow : public QWidget {  
    // ... код окна  
};
```

// 2. В главном окне создаем и показываем второе окно

```
SecondWindow *second = new SecondWindow();  
second->show();
```

Как закрыть окно:

// Закрыть текущее окно

```
this->close();
```

// Или

```
close();
```

Как передать данные между окнами:

// Через конструктор

```
SecondWindow *second = new SecondWindow("текст");
```

// Или через функцию

```
second->setData("текст");
```

Основные виды окон:

- QWidget - обычное окно
- QDialog - диалоговое окно (модальное)
- QMainWindow - главное окно с меню

3. Контрольные вопросы

1. Как создать второе окно в программе?
2. Как закрыть текущее окно?
3. Как открыть новое окно при нажатии кнопки?
4. Что такое модальное окно?
5. Как передать текст из одного окна в другое?
6. Чем отличается QWidget от QDialog?

4. Задания к практической работе (6 задач)

Задание 1: Создать программу с главным окном и кнопкой "Открыть". При нажатии открывается второе окно с надписью "Привет!".

Задание 2: Создать программу с окном настроек. Главное окно имеет кнопку "Настройки", которая открывает второе окно для выбора цвета фона.

Задание 3: Создать программу "Вход в систему". Окно логина запрашивает имя, при правильном имени ("студент") открывает основное окно.

Задание 4: Создать программу с главным окном и окном справки. Кнопка "Справка" в главном окне открывает окно с текстом помощи.

Задание 5: Создать программу калькулятор. Главное окно - ввод чисел, кнопка "Результат" открывает второе окно с ответом.

Задание 6: Создать программу "Анкета". Первое окно для ввода имени, второе для выбора группы, третье для подтверждения.

Что проверяет преподаватель:

1. Есть ли несколько окон в программе ✓
2. Переключаются ли окна правильно ✓
3. Передаются ли данные между окнами (для задач 5-6) ✓
4. Работает ли программа без ошибок ✓

Советы:

- Начинайте с самой простой программы с двумя окнами
- Добавляйте функциональность постепенно
- Тестируйте каждое окно отдельно
- Используйте готовые примеры как шаблон

Эта работа научит вас создавать настоящие приложения с навигацией между окнами!

ПЗ №33. Разработка игрового приложения.

Тема: Разработка игрового приложения.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться создавать простые игры на Qt.
2. Понять основы игрового цикла.
3. Научиться обрабатывать действия пользователя в играх.

Задачи:

- Создать простую игру с кнопками.
- Реализовать обработку нажатий клавиш.
- Сделать игру с таймером.
- Создать игру с подсчетом очков.

2. Теоретические сведения

Основы создания игры:

1. **Игровой цикл** - повторяющиеся действия в игре
2. **Обработка событий** - реакция на действия игрока
3. **Таймер** - для обновления игры
4. **Счетчик очков** - для отслеживания прогресса

Игровой таймер в Qt:

```
QTimer *timer = new QTimer(this);  
connect(timer, &QTimer::timeout, this, &MyGame::updateGame);  
timer->start(100); // Обновлять каждые 100 мс
```

Обработка клавиш:

```
void keyPressEvent(QKeyEvent *event) {  
    if (event->key() == Qt::Key_Space) {  
        // Действие при нажатии пробела  
    }  
}
```

Случайные числа:

```
#include <QRandomGenerator>
```

```
// Случайное число от 1 до 10
```

```
int randomNumber = QRandomGenerator::global()->bounded(1, 11);
```

3. Контрольные вопросы

1. Что такое игровой цикл?
2. Как создать таймер в Qt?
3. Как обработать нажатие клавиши?
4. Как получить случайное число?
5. Что такое QTimer?
6. Как обновлять состояние игры?

4. Задания к практической работе (6 задач)

Задание 1: Создать игру "Счетчик кликов". Кнопка увеличивает счетчик. При достижении 10 очков - победа.

Задание 2: Создать игру "Угадай число". Компьютер загадывает число от 1 до 10, игрок угадывает за 3 попытки.

Задание 3: Создать игру "Быстрый счет". Появляются случайные числа от 1 до 9, нужно нажать кнопку с правильным числом.

Задание 4: Создать игру "Змейка" (упрощенная). Змейка двигается по кнопкам, игрок управляет стрелками.

Задание 5: Создать игру "Поймай шарик". Шарик движется по экрану, нужно нажать на него.

Задание 6: Создать игру "Викторина". Вопрос с 4 вариантами ответов, подсчет правильных ответов.

Что проверяет преподаватель:

1. Работает ли игра ✓
2. Есть ли цель и правила ✓
3. Считаются ли очки ✓
4. Интересно ли играть ✓

Советы:

- Начинайте с самой простой игры (Кликер)
- Добавляйте функции постепенно
- Тестируйте после каждого изменения
- Просите друзей поиграть и дать feedback

Игры - это весело! И отличный способ научиться программированию!

ПЗ №34. Создание процедур обработки событий. Компиляция и запуск приложения.

Тема: Создание процедур обработки событий. Компиляция и запуск приложения.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Научиться создавать процедуры для обработки событий в Qt.
2. Понять, как компилировать и запускать программы.
3. Научиться отлаживать простые программы.

Задачи:

- Создать процедуры, которые вызываются при нажатии кнопок.
- Научиться компилировать программу в Qt Creator.
- Запускать программу и находить ошибки.
- Создать несколько программ с обработкой событий.

2. Теоретические сведения

Событие - это действие пользователя или системы (нажатие кнопки, ввод текста).

Процедура обработки события - функция, которая выполняется при наступлении события.

Как создать процедуру:

// 1. Объявить процедуру

```
void onButtonClicked();
```

// 2. Определить процедуру

```
void MyClass::onButtonClicked() {  
    // Действия при нажатии кнопки  
}
```

// 3. Связать событие с процедурой

```
connect(button, SIGNAL(clicked()), this, SLOT(onButtonClicked()));
```

Или проще с lambda:

```
connect(button, &QPushButton::clicked, [=]() {  
    // Действия при нажатии кнопки  
});
```

Как компилировать в Qt Creator:

1. Нажмите **Ctrl+R** или кнопку ► (зеленый треугольник)
2. Qt Creator сам скомпилирует и запустит программу
3. Если есть ошибки - они покажутся внизу окна

Основные события:

- `clicked()` - нажатие кнопки
- `textChanged()` - изменение текста
- `returnPressed()` - нажатие Enter
- `valueChanged()` - изменение значения

3. Контрольные вопросы

1. Что такое событие в программировании?
2. Как создать процедуру для обработки нажатия кнопки?
3. Как скомпилировать программу в Qt Creator?
4. Как запустить программу после компиляции?
5. Что делать, если программа не компилируется?
6. Как обработать нажатие клавиши Enter в поле ввода?

4. Задания к практической работе (6 задач)

Задание 1: Создать программу с кнопкой. При нажатии кнопки меняется текст на метке.

Задание 2: Создать программу с полем ввода и кнопкой. При нажатии кнопки текст из поля показывается в метке.

Задание 3: Создать программу с двумя кнопками: "Показать" и "Скрыть". Они показывают/скрывают текст.

Задание 4: Создать программу калькулятор. Кнопки "+" и "-" выполняют операции над числами из двух полей.

Задание 5: Создать программу с кнопкой, которая считает, сколько раз ее нажали.

Задание 6: Создать программу для ввода имени. При нажатии Enter или кнопки "Привет" показывать приветствие.

Что проверяет преподаватель:

1. Компилируется ли программа без ошибок ✓
2. Запускается ли программа ✓
3. Работают ли все кнопки и процедуры ✓
4. Правильно ли обрабатываются события ✓
5. Аккуратно ли оформлен код ✓

ПЗ №35. Перегрузка методов. Тестирование и отладка приложения. Решение задач

Тема: Перегрузка методов. Тестирование и отладка приложения. Решение задач приложения.

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цели:

1. Понять, что такое перегрузка методов.
2. Научиться тестировать программы.
3. Находить и исправлять ошибки в коде.

Задачи:

- Создать методы с одинаковым именем, но разными параметрами.
- Протестировать работу программы.
- Найти ошибки в готовом коде и исправить их.
- Написать простые программы с перегруженными методами.

2. Теоретические сведения

Перегрузка методов - это когда у методов одинаковое имя, но разные параметры.

Пример:

// Два метода с одинаковым именем add

```
int add(int a, int b) {  
    return a + b;  
}  
  
double add(double a, double b) {  
    return a + b;  
}
```

// Использование:

```
int result1 = add(5, 3);    // Вызовется первая функция  
double result2 = add(5.5, 3.2); // Вызовется вторая функция
```

Методы могут отличаться:

1. Количеством параметров
2. Типами параметров
3. Порядком параметров

Тестирование программы - проверка, что программа работает правильно.

Как тестировать:

1. Запустить программу
2. Проверить разные варианты ввода
3. Убедиться, что результаты правильные

Отладка - поиск и исправление ошибок.

Простые способы отладки:

// 1. Выводить значения переменных

```
cout << "Значение x = " << x << endl;
```

// 2. Проверять условия

```
if (x > 0) {  
    cout << "x положительное" << endl;  
}
```

3. Контрольные вопросы

1. Что такое перегрузка методов?
2. Чем должны отличаться перегруженные методы?
3. Как проверить, что программа работает правильно?
4. Что делать, если программа не компилируется?
5. Как найти ошибку в программе?
6. Можно ли перегружать методы с разным типом возвращаемого значения?

4. Задания к практической работе (6 задач)

Задание 1: Создать перегруженные методы `print` для вывода: числа, строки и двух чисел.

Задание 2: Создать перегруженные методы `max` для нахождения максимума: двух чисел, трех чисел, массива чисел.

Задание 3: Найти и исправить ошибки в программе с перегрузкой методов.

Задание 4: Создать программу калькулятор с перегруженными методами для сложения разных типов чисел.

Задание 5: Написать тесты для проверки работы программы из задания 4.

Задание 6: Создать программу с перегруженными методами для вычисления площади разных фигур.

Что проверяет преподаватель:

1. Понимаете ли вы перегрузку методов ✓
2. Умеете ли тестировать программы ✓
3. Можете ли найти и исправить ошибки ✓
4. Работают ли все программы без ошибок ✓

Советы:

1. **Начинайте с простого** - одна перегрузка
2. **Тестируйте каждый метод отдельно**
3. **Не бойтесь ошибок** - их исправление учит
4. **Сравнивайте свой код с примерами**
5. **Спрашивайте**, если не понимаете

Важно запомнить:

- Перегрузка методов = одинаковое имя, разные параметры
- Тестирование = проверка, что все работает правильно
- Отладка = поиск и исправление ошибок

1. Цели и задачи практической работы

Цели:

1. Научиться создавать методы с одинаковым именем, но разными параметрами.
2. Понять, как проверять работу программы.
3. Научиться находить и исправлять ошибки в коде.

Задачи:

- Создать несколько методов с одним именем.
- Протестировать, что методы работают правильно.
- Найти ошибки в коде и исправить их.
- Создать простые полезные программы.

2. Теоретические сведения

Перегрузка методов - это когда несколько методов имеют одинаковое имя, но разные параметры.

Пример:

// Три метода с именем "показать"

```
void показать(int число) {  
    cout << "Число: " << число << endl;  
}  
  
void показать(string текст) {  
    cout << "Текст: " << текст << endl;  
}  
  
void показать(int a, int b) {  
    cout << "Числа: " << a << " и " << b << endl;  
}
```

// Использование:

```
показать(10);           // Вызов первого метода  
показать("Привет");    // Вызов второго метода  
показать(5, 7);        // Вызов третьего метода
```

Как программа выбирает метод:

- По количеству параметров
- По типам параметров
- По порядку параметров

Тестирование - проверка, что программа работает правильно.

Как тестировать:

1. Запустить программу
2. Ввести разные данные
3. Проверить, что результат правильный
4. Исправить ошибки, если есть

Поиск ошибок:

1. Прочитать сообщение об ошибке
2. Найти строку с ошибкой
3. Понять, что не так
4. Исправить ошибку

3. Контрольные вопросы

1. Что такое перегрузка методов?
2. Могут ли два метода иметь одинаковое имя?
3. Чем должны отличаться перегруженные методы?
4. Как проверить, что программа работает правильно?
5. Что делать, если программа выдает ошибку?
6. Как найти ошибку в программе?

4. Задания к практической работе (6 задач)

Задание 1: Создать методы привет для вывода: имени, имени и возраста, имени и города.

Задание 2: Создать методы **стоимость** для расчета: стоимости товара по цене за штуку, стоимости со скидкой, стоимости нескольких товаров.

Задание 3: Найти и исправить ошибки в программе с методами расчета площади.

Задание 4: Создать методы **информация** для вывода информации о: студенте (имя, группа), книге (название, автор), машине (марка, год).

Задание 5: Протестировать работу методов из задания 4.

Задание 6: Создать программу-калькулятор с методами для разных операций (сложение, вычитание, умножение, деление).

Что проверяет преподаватель:

1. Понимаете ли вы перегрузку методов ✓
2. Можете ли найти и исправить ошибки ✓
3. Умеете ли тестировать программы ✓
4. Работают ли все программы ✓

Советы:

1. **Начинайте с самого простого** задания
2. **Копируйте код** из примеров и меняйте его
3. **Тестируйте каждый метод отдельно**
4. **Не бойтесь делать ошибки** - это нормально
5. **Спрашивайте помощи**, если не получается

Помните: Программирование - это как сборка конструктора. Сначала учитесь собирать простые блоки, потом сложные!

ПЗ №37. Создание наследованного класса.

Тема: Создание наследованного класса

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Научиться создавать иерархии классов на языке C++ с использованием механизма наследования.

Задачи:

- Закрепить понимание принципов наследования.
- Научиться создавать производные (дочерние) классы.
- Освоить использование спецификаторов доступа (`public`, `protected`, `private`) при наследовании.
- Научиться переопределять методы базового класса в производных классах.
- Научиться использовать указатели и ссылки базового класса для работы с объектами производных классов.

2. Теоретические сведения

Наследование – это механизм объектно-ориентированного программирования, позволяющий создавать новый класс (производный, дочерний) на основе уже существующего (базового, родительского). Новый класс автоматически включает все поля и методы базового класса и может добавлять свои собственные или изменять унаследованные.

Синтаксис объявления производного класса:

```
class ИмяПроизводногоКласса : [спецификатор_доступа] ИмяБазовогоКласса {  
    // Члены производного класса  
};
```

Спецификаторы доступа при наследовании:

- `public`: Открытые (`public`) и защищенные (`protected`) члены базового класса сохраняют свой уровень доступа в производном классе. **Наиболее часто используется.**
- `protected`: Открытые и защищенные члены базового класса становятся защищенными (`protected`) в производном классе.
- `private`: Открытые и защищенные члены базового класса становятся приватными (`private`) в производном классе.

Переопределение методов: Производный класс может определить свою собственную версию метода, уже существующего в базовом классе. Для вызова переопределенного метода базового класса используется оператор области видимости `::` (например, `БазовыйКласс::Метод()`).

Конструкторы и деструкторы: Конструкторы не наследуются. При создании объекта производного класса сначала вызывается конструктор базового класса, затем конструктор производного. Деструкторы вызываются в обратном порядке: сначала производного, затем базового.

3. Контрольные вопросы

1. Что такое наследование и для чего оно применяется?
2. Какой синтаксис используется для создания производного класса?
3. В чем разница между спецификаторами `public`, `protected` и `private` при наследовании?
4. Какой спецификатор доступа при наследовании используется чаще всего и почему?
5. Что такое переопределение метода? Как вызвать переопределенный метод базового класса из производного?
6. В каком порядке вызываются конструкторы и деструкторы при создании и уничтожении объекта производного класса?
7. Наследуются ли конструкторы, деструкторы и операторы присваивания?

4. Задания к практической работе

Общее условие: Для всех заданий необходимо создать проект в среде **Qt Creator** (консольное приложение, проект Qt Widgets или QML — на усмотрение преподавателя). Каждую задачу оформить в виде отдельного файла `.cpp` или в разных функциях `main` с переименованием проекта.

Задача 1 (Базовый уровень).

Создайте базовый класс `Работник` с полями: `имя` (строка) и `зарплата` (целое число). Создайте производный класс `Менеджер`, который добавляет поле `бонус` (целое число). В обоих классах создайте метод `показатьИнфо()`, выводящий данные на экран. Протестируйте работу классов в `main()`.

Задача 2 (Базовый уровень).

Создайте базовый класс `Фигура` с защищенным полем `цвет` (строка) и чисто виртуальным методом `площадь()` (возвращает `double`). Создайте производный класс `Прямоугольник` с полями `ширина` и `высота` (оба `double`), который переопределяет метод `площадь()`. Создайте объект прямоугольника и вычислите его площадь.

Задача 3 (Средний уровень).

Создайте иерархию классов для представления банковских счетов.

- Базовый класс `БанковскийСчет`: `номерСчета` (строка), `баланс` (`double`). Методы: `положить(double сумма)`, `снять(double сумма)`, `показатьБаланс()`.
- Производный класс `ДепозитныйСчет`: добавляет поле `процентнаяСтавка` (`double`). Добавьте метод `начислитьПроценты()`, который увеличивает баланс на соответствующий процент. Создайте объекты обоих типов счетов и протестируйте все методы.

Задача 4 (Средний уровень).

Создайте базовый класс `УстройствоВвода` с полем `интерфейсПодключения` (например, `"USB"`, `"Bluetooth"`) и методом `вести()` (выводит сообщение `"Ввод данных..."`).

Создайте производные классы: `Клавиатура` (добавляет поле `количествоКлавиш`) и `Мышь` (добавляет поле `количествоКнопок`). Переопределите в них метод `вести()` так, чтобы он выводил специфичное для устройства сообщение (например, `"Ввод текста с клавиатуры"` или `"Щелчок мышью"`).

В функции `main()` создайте массив указателей на `УстройствоВвода` и поместите в него адреса объектов `Клавиатура` и `Мышь`. Вызовите метод `вести()` через эти указатели (протестируйте полиморфное поведение).

Задача 5 (Средний/Сложный уровень).

Создайте иерархию классов для представления простой геометрической библиотеки.

- Базовый класс `Точка` с полями `x`, `y` (`double`).
- Производный от `Точки` класс `Окружность`: добавляет поле `радиус`. Добавляет методы `площадь()` и `длинаОкружности()`.
- Производный от `Точки` класс `Прямоугольник`: добавляет поля `ширина`, `высота`. Добавляет методы `площадь()` и `периметр()`.

Протестируйте создание объектов окружности и прямоугольника, задав их позицию (координаты центра/левого верхнего угла) и размеры, и выведите их параметры.

Задача 6 (Сложный уровень — целостная программа).

Реализуйте простую систему учета техники.

- Базовый класс `Оборудование`: `инвентарныйНомер` (`int`), `модель` (`string`), `статус` (`исправен/неисправен` — `bool` или `enum`). Методы для изменения/получения статуса, вывода полной информации.
- Производный класс `Компьютер`: добавляет поля `процессор` (`string`), `объемОЗУ` (`int`). Переопределяет метод вывода информации.
- Производный класс `Принтер`: добавляет поля `тип` (`струйный/лазерный`, `string`), `цветной` (`bool`). Переопределяет метод вывода информации.
- В функции `main()` создайте массив (или `std::vector`) указателей на `Оборудование`. Добавьте в него несколько компьютеров и принтеров. Напишите цикл, который выводит информацию о всей технике в списке. Реализуйте меню, позволяющее изменить статус выбранного по инвентарному номеру оборудования.

Критерии оценки:

- Корректность синтаксиса и логики программы.
- Правильное применение принципов наследования.
- Корректное использование спецификаторов доступа.

- Наличие переопределения методов, где это необходимо.
- Читаемость и комментирование кода (по необходимости).
- Работоспособность программы в среде Qt.

ПЗ №38. Классы и объекты.

Тема: Классы и объекты в C++

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Сформировать практические навыки создания и использования классов и объектов в языке программирования C++.

Задачи:

- Закрепить понимание принципов объектно-ориентированного программирования (ООП).
- Научиться объявлять классы, определять их поля (данные-члены) и методы (функции-члены).
- Освоить создание объектов (экземпляров классов).
- Научиться использовать спецификаторы доступа (`public`, `private`, `protected`).
- Освоить написание конструкторов и деструкторов.
- Научиться реализовывать методы класса.

2. Теоретические сведения

Класс – это пользовательский тип данных, который объединяет данные (поля) и функции (методы) для работы с этими данными в одну логическую единицу. Класс является описанием (чертежом) будущих объектов.

Объект – это конкретный экземпляр класса, созданный в памяти программы. У каждого объекта есть свои собственные значения полей.

Спецификаторы доступа:

- `private` (закрытые): Члены класса доступны только из методов этого же класса. Используются для сокрытия внутренней реализации (инкапсуляция).
- `public` (открытые): Члены класса доступны из любого места программы, где виден объект.
- `protected` (защищенные): Аналогичны `private`, но доступны также в производных классах (используются при наследовании).

Конструктор – это специальный метод, который вызывается автоматически при создании объекта. Имя конструктора совпадает с именем класса. Используется для инициализации полей объекта.

- Конструктор по умолчанию (без параметров).
- Конструктор с параметрами.
- Конструктор копирования.

Деструктор – это специальный метод, который вызывается автоматически при уничтожении объекта. Имя деструктора: `~ИмяКласса()`. Используется для освобождения ресурсов (памяти, файлов и т.д.).

3. Контрольные вопросы

1. Что такое класс и объект? В чем между ними разница?
2. Для чего нужны спецификаторы доступа `public`, `private` и `protected`?
3. Что такое конструктор? Какие типы конструкторов вы знаете?
4. Что такое деструктор и когда он вызывается?
5. Что такое инкапсуляция и как она реализуется в C++?
6. Для чего нужны методы-геттеры (`get`) и сеттеры (`set`)?
7. Как создать массив объектов?
8. Что происходит, если не написать конструктор для класса?

4. Задания к практической работе

Общее условие: Для всех заданий необходимо создать проект в среде **Qt Creator** (рекомендуется консольное приложение `Qt Console Application`). Каждую задачу оформить в виде отдельного файла или в разных функциях `main` с переименованием проекта.

Задача 1 (Базовый уровень). Книга

Создайте класс `Book`. Закрытые поля класса: `title` (название, строка), `author` (автор, строка), `year` (год издания, целое число).

1. Реализуйте конструктор с параметрами для инициализации всех полей.
2. Реализуйте метод `display()`, который выводит информацию о книге в удобном формате.

3. Создайте 2-3 объекта класса Book в функции main() и продемонстрируйте работу метода display().

Задача 2 (Базовый уровень). Счет в банке

Создайте класс BankAccount. Закрытые поля: accountNumber (номер счета, строка), balance (баланс, вещественное число).

1. Реализуйте конструктор по умолчанию (устанавливает номер "Не указан", баланс 0.0).
2. Реализуйте конструктор с параметрами.
3. Реализуйте методы: deposit(double sum) (положить деньги), withdraw(double sum) (снять деньги, с проверкой, что сумма не больше баланса), printBalance() (вывести текущий баланс).
4. Создайте объект, положите на счет 1000, снимите 300 и выведите итоговый баланс.

Задача 3 (Средний уровень). Прямоугольник

Создайте класс Rectangle. Закрытые поля: width (ширина), height (высота) - вещественные числа.

1. Реализуйте два конструктора: по умолчанию (ширина=1, высота=1) и с параметрами.
2. Реализуйте методы-геттеры и сеттеры для полей. В сеттерах добавьте проверку, чтобы значения были положительными.
3. Добавьте методы: calculateArea() (вычисление площади), calculatePerimeter() (вычисление периметра).
4. В main() создайте несколько прямоугольников, измените размеры одного из них через сеттеры и для всех выведите площадь и периметр.

Задача 4 (Средний уровень). Автомобиль

Создайте класс Car. Закрытые поля: brand (марка), model (модель), year (год выпуска), fuelLevel (уровень топлива, от 0 до 100, %), isEngineOn (двигатель включен - булево значение).

1. Создайте конструктор для инициализации марки, модели и года. Уровень топлива установите в 100, двигатель - выключен.
2. Реализуйте методы: startEngine() (включает двигатель, если топливо > 0), stopEngine() (выключает двигатель), refuel(int amount) (заправка), drive() (если двигатель включен и топливо > 0, уменьшает уровень топлива на 10 и выводит сообщение "Едем...", иначе выводит сообщение об ошибке).
3. Создайте объект, заправьте его, заведите, несколько раз вызовите drive(), затем заглушите.

Задача 5 (Средний/Сложный уровень). Магазин (массив объектов)

Создайте класс Product. Закрытые поля: id (уникальный код, целое число), name (название), price (цена), quantity (количество на складе).

1. Реализуйте необходимые конструкторы, геттеры и сеттеры.
2. Добавьте метод getTotalCost() (возвращает общую стоимость товара на складе = цена * количество).
3. В функции main() создайте массив из 3-5 объектов Product.
4. Напишите функцию (или код в main), которая выводит информацию о всех товарах в виде таблицы.
5. Дополнительно: Вычислите и выведите общую стоимость всех товаров на складе.

Задача 6 (Сложный уровень - меню и управление). Студенческая группа

Создайте класс Student (как в примере, можно добавить поле grades - массив оценок).

Создайте класс StudentGroup. Его закрытое поле - массив (например, из 10 элементов) объектов Student и переменная count для учета текущего количества студентов в группе.

1. Реализуйте в классе StudentGroup методы:
 - o addStudent(Student s) - добавление нового студента в группу (если есть место).
 - o removeStudent(int id) - удаление студента по номеру билета.
 - o findStudent(int id) - поиск студента по номеру билета (вывод информации).
 - o displayAll() - вывод информации о всех студентах в группе.
2. В функции main() создайте объект StudentGroup.

3. Реализуйте простое консольное меню с использованием `switch-case`, позволяющее пользователю выбирать действия: добавить студента, удалить, найти, показать всех, выйти. Данные можно вводить с клавиатуры.

ПЗ №39. Составление начальной иерархии и структуры классов.

Тема: Составление начальной иерархии и структуры классов

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Научиться проектировать простые иерархии классов и структурировать программу на основе объектно-ориентированного подхода.

Задачи:

- Научиться выделять сущности предметной области и определять отношения между ними.
- Освоить создание базовых и производных классов.
- Научиться правильно выбирать уровни доступа для членов классов.
- Сформировать навыки проектирования простой иерархии классов "сверху вниз".
- Научиться распределять функциональность между классами в иерархии.

2. Теоретические сведения

Иерархия классов - это система классов, организованная по принципу "общее-частное" с использованием механизма наследования. В иерархии выделяют:

1. **Базовый (родительский) класс** - содержит общие характеристики и поведение для всех производных классов.
2. **Производный (дочерний) класс** - наследует свойства и методы базового класса и добавляет свои специфические особенности.

Принципы построения иерархии:

- **Принцип "является" (is-a)** - если можно сказать, что объект класса В "является" объектом класса А, то В должен наследоваться от А.
- **Инкапсуляция** - сокрытие внутренней реализации классов.
- **Спецификаторы доступа в иерархии:**
 - **public** наследование: сохраняет уровни доступа
 - **protected** наследование: public → protected
 - **private** наследование: public/protected → private

Проектирование иерархии:

1. Выделить общие характеристики для группы объектов.
2. Создать базовый класс с этими характеристиками.
3. Определить специфические особенности для каждого типа объектов.
4. Создать производные классы, добавляющие эти особенности.

3. Контрольные вопросы

1. Что такое иерархия классов и для чего она используется?
2. Какой принцип ("является" или "имеет") используется для построения иерархии наследования?
3. Что такое базовый и производный классы? Приведите примеры.
4. Как правильно выбрать, какие поля и методы выносить в базовый класс?
5. Что такое полиморфизм и как он связан с иерархией классов?
6. Какие спецификаторы доступа используются при наследовании и чем они отличаются?
7. Почему часто делают деструктор виртуальным в базовом классе?

4. Задания к практической работе

Общее условие: Создайте проект в Qt Creator. Каждую задачу можно оформить как отдельный проект или как разные -файлы в одном проекте.

Задача 1: Иерархия "Сотрудники предприятия"

Спроектируйте и реализуйте иерархию классов для сотрудников предприятия:

1. Базовый класс **Сотрудник** (поля: ФИО, должность, оклад)
2. Производный класс **Руководитель** (добавляет: количество подчиненных, премия)
3. Производный класс **Инженер** (добавляет: категория, проект)
4. Производный класс **ТехническийПерсонал** (добавляет: смена, участок)

Создайте по одному объекту каждого типа и выведите информацию о них.

Задача 2: Иерархия "Геометрические фигуры"

Создайте иерархию классов для геометрических фигур:

1. Базовый класс **Фигура** (поля: цвет, название; методы: показатьИнфо(), площадь() - виртуальный)
2. Производный класс **Круг** (добавляет: радиус)
3. Производный класс **Прямоугольник** (добавляет: ширина, высота)
4. Производный класс **Треугольник** (добавляет: сторонаА, сторонаВ, сторонаС)

Реализуйте вычисление площади для каждой фигуры.

Задача 3: Иерархия "Учебные заведения"

Спроектируйте иерархию для учебных заведений:

1. Базовый класс **УчебноеЗаведение** (название, адрес, годОснования)
2. Производный класс **Школа** (количествоКлассов, профиль)
3. Производный класс **Колледж** (отделения, срокОбучения)
4. Производный класс **Университет** (факультеты, количествоСтудентов)

Добавьте метод провестиДеньОткрытыхДверей() с разной реализацией для каждого типа.

Задача 4: Иерархия "Библиотека"

Создайте иерархию классов для библиотечных материалов:

1. Базовый класс **БиблиотечныйМатериал** (шифр, название, годИздания)
2. Производный класс **Книга** (автор, количествоСтраниц, жанр)
3. Производный класс **Журнал** (номер, месяцИздания, тематика)
4. Производный класс **ЭлектронныйРесурс** (ссылка, форматФайла, размер)

Реализуйте систему, где можно выводить информацию в разном формате для каждого типа материалов.

Задача 5: Иерархия "Спортивные соревнования"

Спроектируйте иерархию для спортивных мероприятий:

1. Базовый класс **Соревнование** (видСпорта, дата, место)
2. Производный класс **КомандноеСоревнование** (количествоКоманд, размерКоманды)
3. Производный класс **ИндивидуальноеСоревнование** (количествоУчастников, системаПодсчетаОчков)
4. Производный от **КомандноеСоревнование** класс **ФутбольныйМатч** (длительность, стадион)
5. Производный от **ИндивидуальноеСоревнование** класс **ШахматныйТурнир** (количествоТуров, рейтинг)

Создайте объекты разных уровней иерархии.

Задача 6: Иерархия "Техника" с трехуровневой структурой

Создайте трехуровневую иерархию:

1. Уровень 1: **ЭлектронноеУстройство** (производитель, мощность, цена)
2. Уровень 2: **БытоваяТехника** (гарантия, энергопотребление)
 - Производный: **Холодильник** (объем, классЭнергосбережения)
 - Производный: **СтиральнаяМашина** (загрузка, программ)
3. Уровень 2: **КомпьютернаяТехника** (оперативнаяПамять, процессор)
 - Производный: **Ноутбук** (диагональЭкрана, времяРаботыОтБатарей)
 - Производный: **НастольныйКомпьютер** (видеокарта, блокПитания)

Продемонстрируйте создание объектов самого нижнего уровня иерархии и вызов методов всех уровней.

ПЗ №40. Итоговый проект

Тема: Итоговый проект по основам объектно-ориентированного программирования

Дисциплина: Основы алгоритмизации и программирования

Курс: 2, СПО

1. Цели и задачи практической работы

Цель: Применить полученные знания по ООП для создания законченного приложения с использованием классов, объектов, наследования и полиморфизма.

Задачи:

- Закрепить навыки проектирования и реализации классов.
- Научиться создавать иерархии классов с использованием наследования.
- Применить принципы инкапсуляции, наследования и полиморфизма.
- Разработать консольное приложение с меню и взаимодействием с пользователем.
- Научиться структурировать код проекта.

2. Теоретические сведения (краткое повторение)

Ключевые концепции ООП:

1. **Инкапсуляция** - объединение данных и методов в один класс, сокрытие внутренней реализации.
2. **Наследование** - создание нового класса на основе существующего.
3. **Полиморфизм** - возможность объектов с одинаковой спецификацией иметь различную реализацию.

Структура проекта:

- Разделение на заголовочные файлы (.h) и файлы реализации (.)
- Организация классов в логические группы
- Создание пользовательского интерфейса (меню)

Этапы разработки:

1. Анализ задачи и проектирование классов
2. Реализация классов
3. Создание пользовательского интерфейса
4. Тестирование и отладка
3. Контрольные вопросы
 1. Какие три основных принципа ООП вы знаете?
 2. Как правильно организовать проект из нескольких файлов?
 3. Что такое конструктор и деструктор класса?
 4. Чем отличается public наследование от private?
 5. Как реализовать полиморфное поведение объектов?
 6. Что такое вектор (vector) и для чего он используется?
 7. Как организовать цикл меню в консольном приложении?
4. Задания к практической работе (Итоговые проекты)

Общее требование: Для каждого проекта создать отдельное приложение в Qt Creator (консольное или с графическим интерфейсом - по выбору преподавателя). Проект должен содержать не менее 3-4 классов, организованных в иерархию.

Задачи по вариантам

Проект 1: Система учета студентов

Разработайте систему для учета успеваемости студентов.

Требования:

1. Базовый класс `Person` (ФИО, возраст)
2. Класс `Student`, наследуемый от `Person` (номер зачетки, группа, массив оценок)
3. Класс `Teacher`, наследуемый от `Person` (должность, предмет)
4. Класс `Group` (название группы, массив студентов, куратор)
5. Функционал:
 - Добавление/удаление студентов

- Расчет среднего балла студента
- Вывод студентов с баллом выше заданного
- Сохранение/загрузка данных в файл
- 6. Консольное меню для управления

Проект 2: Магазин электроники

Создайте систему учета товаров в магазине.

Требования:

1. Базовый класс **Product** (название, цена, количество)
2. Производные классы: **Smartphone**, **Laptop**, **Tablet** (со специфичными полями)
3. Класс **ShoppingCart** (список товаров, общая стоимость)
4. Класс **Store** (массив товаров, методы поиска и фильтрации)
5. Функционал:
 - Добавление товаров разных категорий
 - Поиск товаров по критериям
 - Формирование чека покупки
 - Отчет по остаткам товаров
6. Простое консольное меню с имитацией покупок

Проект 3: Библиотека медиа-контента

Система для учета фильмов, книг и музыки.

Требования:

1. Базовый класс **Media** (название, год, рейтинг)
2. Производные классы: **Movie**, **Book**, **MusicAlbum**
3. Класс **Library** для хранения коллекции
4. Класс **Playlist** (для создания подборок)
5. Функционал:
 - Добавление медиа разных типов
 - Поиск по названию, году, рейтингу
 - Сортировка коллекции
 - Создание плейлистов
 - Расчет общей статистики
6. Меню с возможностью просмотра по категориям

Проект 4: Система бронирования отелей

Программа для управления бронированием номеров.

Требования:

1. Класс **Room** (номер, тип, цена, статус)
2. Класс **Client** (ФИО, паспорт, телефон)
3. Класс **Booking** (клиент, комната, даты, стоимость)
4. Класс **Hotel** (массив комнат, массив бронирований)
5. Функционал:
 - Просмотр свободных номеров
 - Бронирование на даты
 - Расчет стоимости проживания
 - Отмена бронирования
 - Отчет по занятости
6. Пошаговое меню бронирования

Проект 5: Игра "Зоопарк"

Простая система управления зоопарком.

Требования:

1. Базовый класс **Animal** (кличка, возраст, вид)
2. Производные классы для разных семейств: **Mammal**, **Bird**, **Reptile**
3. Класс **Employee** (смотритель)
4. Класс **Zoo** (массив животных, сотрудники)

5. Функционал:
 - Добавление животных с характеристиками
 - Назначение смотрителей
 - Имитация кормления/ухода
 - Поиск животных по критериям
 - Расчет расходов на содержание
6. Игровое меню с событиями

Проект 6: Система задач (To-Do List)

Персональный органайзер задач.

Требования:

1. Базовый класс Task (описание, статус, дата)
2. Производные классы: WorkTask, PersonalTask, StudyTask
3. Класс Category для группировки задач
4. Класс TaskManager (управление задачами)
5. Функционал:
 - Создание задач разных типов
 - Отметка выполнения
 - Сортировка по дате/приоритету
 - Поиск задач
 - Статистика выполнения
6. Цветное консольное меню (если используется консоль)

Критерии оценки итогового проекта:

1. **Корректность ООП-дизайна (20%)** - правильное использование классов, наследования, инкапсуляции
2. **Функциональность (30%)** - выполнение всех требований задания
3. **Качество кода (20%)** - читаемость, комментарии, структура
4. **Пользовательский интерфейс (15%)** - удобство работы с программой
5. **Обработка ошибок (15%)** - проверка вводимых данных, обработка исключительных ситуаций

Рекомендации по выполнению:

1. Начните с проектирования классов на бумаге
2. Реализуйте сначала базовые классы
3. Постепенно добавляйте функциональность
4. Тестируйте каждую новую функцию
5. Добавляйте комментарии по мере написания кода
6. Создайте простое меню сначала, потом усложняйте его

Время выполнения: 4-6 академических часов

Список литературы

Для студентов

1. Основы алгоритмизации и программирования / И.Г. Семакин, А.П. Шестаков. – Москва: Академия, 2021. – 304 с.
2. Огнева, М. В. Программирование на языке С++: практический курс : учебник для среднего профессионального образования / М. В. Огнева, Е. В. Кудрина, А. А. Казачкова. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 342 с. — (Профессиональное образование). — ISBN 978-5-534-18975-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 5 — URL: <https://urait.ru/bcode/563669/p.5> (дата обращения: 05.07.2025).

Для преподавателей

Трофимов, В. В. Основы алгоритмизации и программирования : учебник для среднего профессионального образования / В. В. Трофимов, Т. А. Павловская ; под редакцией В. В. Трофимова. — Москва : Издательство Юрайт, 2021. — 137 с. — (Профессиональное образование). — ISBN 978-5-534-07321-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/473347> (дата обращения: 13.12.2021).

Интернет-ресурсы

Кудрина, Е. В. Основы алгоритмизации и программирования на языке С# : учебное пособие для среднего профессионального образования / Е. В. Кудрина, М. В. Огнева. — Москва : Издательство Юрайт, 2021. — 322 с. — (Профессиональное образование). — ISBN 978-5-534-10772-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/475228> (дата обращения: 13.12.2021).

Отчет

по практической работе №1
на тему «???»

дисциплина **ОП.04 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ**
Специальность 09.02.07

Студент:
Иванов И.И.
Группа:
ИСП-24
Проверил преподаватель:
Комарова А.В.
Оценка:

«___» _____ 20__ г.

(подпись)

Новосибирск 2025 г.

Содержание отчета

1. **Цель работы**
2. **Задание**
3. **Теоретические сведения** (кратко)
4. **Блок-схема алгоритма** (если требуется)
5. **Листинг программы**
6. **Тестирование выполнения программы** (скриншоты)
7. **Ответы на контрольные вопросы**
8. **Выводы**

Страница 1. Цель работы

Цель работы: [Сформулируйте цель работы своими словами, можно взять из задания]

Задачи:

1. [Задача 1]
2. [Задача 2]
3. [Задача 3]

Страница 2. Задание

Вариант № [Ваш вариант, если есть]

Условие задачи:

[Полностью перепишите условие вашего задания]

Страница 3. Теоретические сведения

[Название темы, например: "Классы и объекты в C++"]

Основные понятия:

1. **[Понятие 1]:** [Определение]
2. **[Понятие 2]:** [Определение]
3. **[Понятие 3]:** [Определение]

Синтаксис:

[Пример синтаксиса, например:

```
class ИмяКласса {  
private:  
    // поля  
public:  
    // методы  
};]
```

Страница 4. Блок-схема алгоритма

(Если не требуется - пропустить)

[Место для вставки блок-схемы]

Рисунок 1 - Блок-схема алгоритма

Пояснение к блок-схеме:

[Краткое описание алгоритма]

Страница 5. Листинг программы

Проект в среде Qt Creator

- **Название проекта:** [ProjectName]
- **Тип проекта:** [Console App / Widgets App]
- **Файлы проекта:**

main.

[другие файлы, если есть]

Код программы:

// main.

[Вставьте полный код вашей программы]

Пояснения к коду:

1. [Пояснение к важному фрагменту 1]
2. [Пояснение к важному фрагменту 2]
3. [Пояснение к важному фрагменту 3]

Страница 6. Результаты выполнения программы

Тест 1: [Название теста]

[Вставьте скриншот 1]

Рисунок 2 - Результат выполнения программы (тест 1)

Тест 2: [Название теста]

[Вставьте скриншот 2]

Рисунок 3 - Результат выполнения программы (тест 2)

Анализ результатов:

[Кратко опишите, что делает программа и как она работает]

Страница 7. Ответы на контрольные вопросы

Вопрос 1: [Текст вопроса]

Ответ: [Ваш ответ]

Вопрос 2: [Текст вопроса]

Ответ: [Ваш ответ]

[и так далее...]

Страница 8. Выводы

Выводы по работе:

1. [Что нового узнали]
2. [Какие навыки приобрели]
3. [С какими трудностями столкнулись и как их преодолели]
4. [Как можно улучшить программу]

Оценка выполнения работы:

- Работа выполнена полностью/частично
- Все задачи решены верно/с ошибками
- Программа работает корректно/требуется доработка

Приложение А: Инструкция по оформлению

1. **Форматирование текста:**

- Шрифт: Times New Roman, 14 пт
- Междустрочный интервал: 1.5
- Поля: 2 см со всех сторон
- Выравнивание: по ширине

2. **Оформление кода:**

- Шрифт: Consolas или Courier New, 12 пт
- Фон: светлый
- Переносы: без переносов

3. **Скриншоты:**

- Размер: не более 10x15 см
- Подпись снизу по центру
- Качество: читаемый текст

Приложение Б: Чек-лист перед сдачей

Перед сдачей отчета проверьте:

- На титульном листе указаны ваши данные
- Номер работы соответствует заданию
- Код программы полный и соответствует заданию
- Скриншоты показывают РАЗНЫЕ тестовые случаи
- Ответы на контрольные вопросы даны полные
- Выводы написаны самостоятельно
- Все страницы пронумерованы
- Нет орфографических ошибок

Приложение В: Пример заполнения для конкретной работы

Пример для работы №38 "Классы и объекты"

На странице "Теоретические сведения":

Классы и объекты в C++

Основные понятия:

1. **Класс** - пользовательский тип данных, объединяющий данные и методы для работы с ними.
2. **Объект** - экземпляр класса, созданный в памяти программы.
3. **Конструктор** - специальный метод для инициализации объекта.

Синтаксис создания класса:

```
class Student {
private:
    string name;
    int age;
public:
    Student(string n, int a) {
        name = n;
        age = a;
    }
    void display() {
        cout << "Name: " << name << endl;
    }
};
```

На странице "Выводы":

Выводы по работе:

1. Научился создавать классы в C++ с использованием конструкторов
2. Освоил принципы инкапсуляции через private/public доступ

3. Столкнулся с ошибкой компиляции из-за отсутствия точки с запятой после класса
4. Программу можно улучшить, добавив проверку вводимых данных

Рекомендации для студентов:

1. **Сохраняйте все версии программы** - может пригодиться при доработках
2. **Делайте скриншоты сразу** при успешном запуске
3. **Комментируйте код** по мере написания - потом легче делать отчет
4. **Используйте один шаблон** для всех работ - это сэкономит время
5. **Храните отчеты в облаке** - чтобы не потерять

Этот шаблон подойдет для всех практических работ по программированию на C++/Qt и поможет студентам оформлять работы единообразно и профессионально.