

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тульский государственный университет»

Институт прикладной математики и компьютерных наук
Кафедра информационной безопасности

Программирование

Лабораторная работа № 4

Тема

Изучение принципов построения и использования функций в языке
C++.

Выполнил
Студент гр. Б260221
Воробьёв А.А.
Проверила
Сафронова М.А.

Тула 2023 г.

Цель работы: Изучение принципов построения и использования функций в языке C++ является важным этапом в процессе обучения программированию. Функции в языке программирования представляют собой отдельные блоки кода, которые выполняют определенные задачи. Функции улучшают читаемость, повторное использование кода и делают программы более модульными.

В контексте языка C++, вы можете изучать следующие концепции:

- **Определение функции:** Вы должны понять, как определить функцию в C++, что включает в себя имя функции, тип возвращаемого значения, параметры, и тело функции.
- **Объявление функции:** Объявление функции в C++ является важным шагом для обеспечения правильной связи между различными частями программы.
- **Вызов функции:** После определения и объявления функции, вы должны знать, как правильно вызвать функцию в вашей программе.
- **Передача параметров:** В C++, параметры могут быть переданы по значению, по ссылке или по указателю. Каждый из этих методов имеет свои особенности и может быть использован в зависимости от ситуации.
- **Возвращаемые значения:** Функции в C++ могут возвращать значения, которые затем могут быть использованы в других частях программы. Вы должны понимать, как возвращать значения из функций и как их использовать.
- **Перегрузка функций:** В C++ можно иметь несколько функций с одинаковым именем, но разными параметрами. Это называется перегрузкой функций, и это мощный инструмент в C++.
- **Шаблонные функции:** Шаблоны в C++ позволяют определить общую функцию, которая будет работать с различными типами данных.
- **Lambda-функции:** Lambda-функции в C++ представляют собой анонимные функции, которые можно использовать в качестве аргументов для других функций или для создания объектов функций на лету.

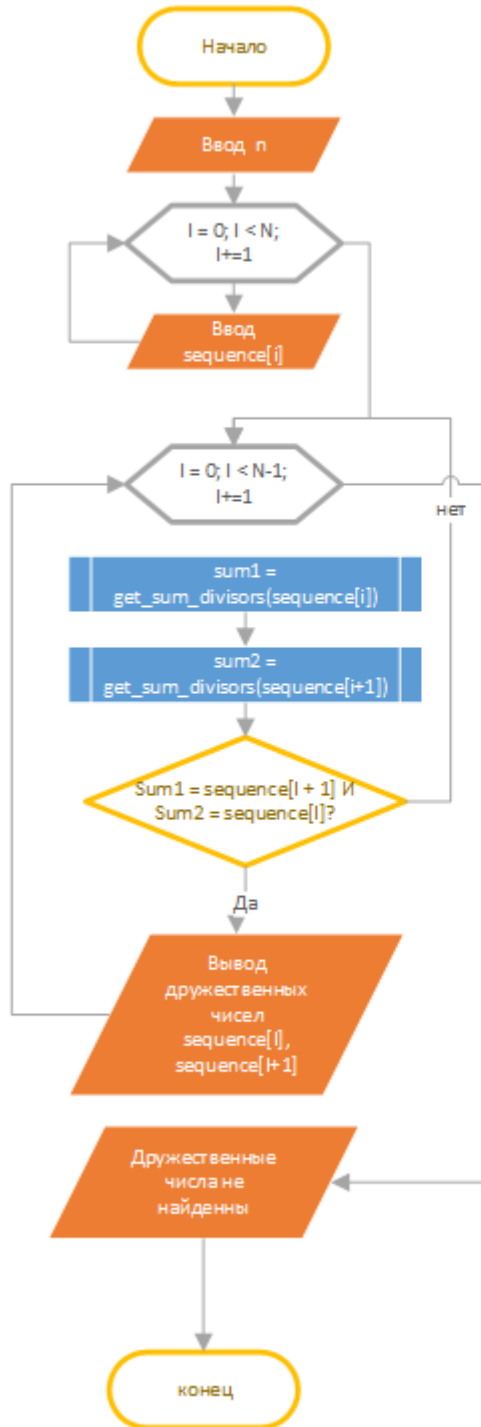
Каждая из этих тем важна для разработки компетенции в области программирования на C++. Понимание и умение использовать функции помогут вам создавать более эффективные, модульные и переиспользуемые программы.

Задание 1. Применение функций при работе с последовательностями чисел

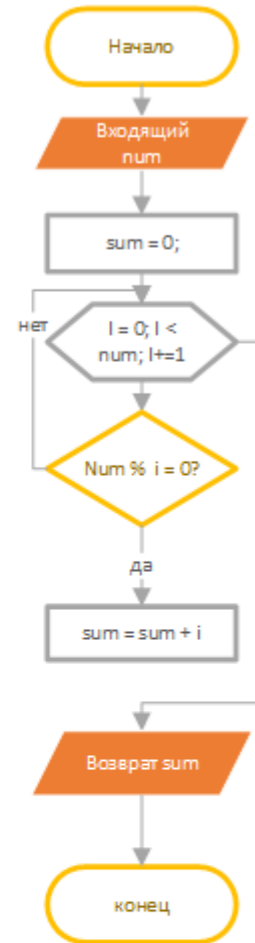
Вводится последовательность из N целых положительных элементов. Проверить, содержит ли последовательность хотя бы одну пару соседних дружественных чисел. Два различных натуральных числа являются дружественными, если сумма всех делителей первого числа (кроме самого числа) равна второму числу. Например, 220 и 284, 1184 и 1210, 2620 и 2924, 5020 и 5564.

Блок-схема:

Основной блок



Функция get_sum_divisors



Код программы:

```
void Task1()
{
    cout << endl
        << endl
        << "Task 1" << endl;
    cout << "FOR" << endl;
    int n;
    cout << "Введите количество элементов: ";
    cin >> n;

    int sequence[n];
    cout << "Введите последовательность чисел:" << endl;
    for (int i = 0; i < n; i++)
    {
        cin >> sequence[i];
    }

    for (int i = 0; i < n - 1; i++)
    {
        int sum1 = get_sum_divisors(sequence[i]);
        int sum2 = get_sum_divisors(sequence[i + 1]);
        if (sum1 == sequence[i + 1] && sum2 == sequence[i])
        {
            cout << "Найдена пара дружественных чисел: " << sequence[i]
                << " и " << sequence[i + 1] << endl;
        }
    }

    cout << "Пара дружественных чисел не найдена." << endl;
}

int get_sum_divisors(int num)
{
    int sum = 0;
    for (int i = 1; i < num; i++)
    {
```

```

        if (num % i == 0)
        {
            sum += i;
        }
    }
    return sum;
}

```

Вывод программы:

Run: CodeSolution x

```

Введите номер лабораторной работы (1-5) или 'q' для выхода: 4
Введите номер задания (1-3): 1

Task 1
FOR
Введите количество элементов: 10
Введите последовательность чисел:
15
220
284
500
600
700
800
900
1000
1100
Найдена пара дружественных чисел: 220 и 284
Пара дружественных чисел не найдена.

```

Run: CodeSolution x

```

C:\Users\AzerQ\Documents\SessionProg2023\CodeSolution\cmake-build-
Active code page: 65001
Введите номер лабораторной работы (1-5) или 'q' для выхода: 4
Введите номер задания (1-3): 1

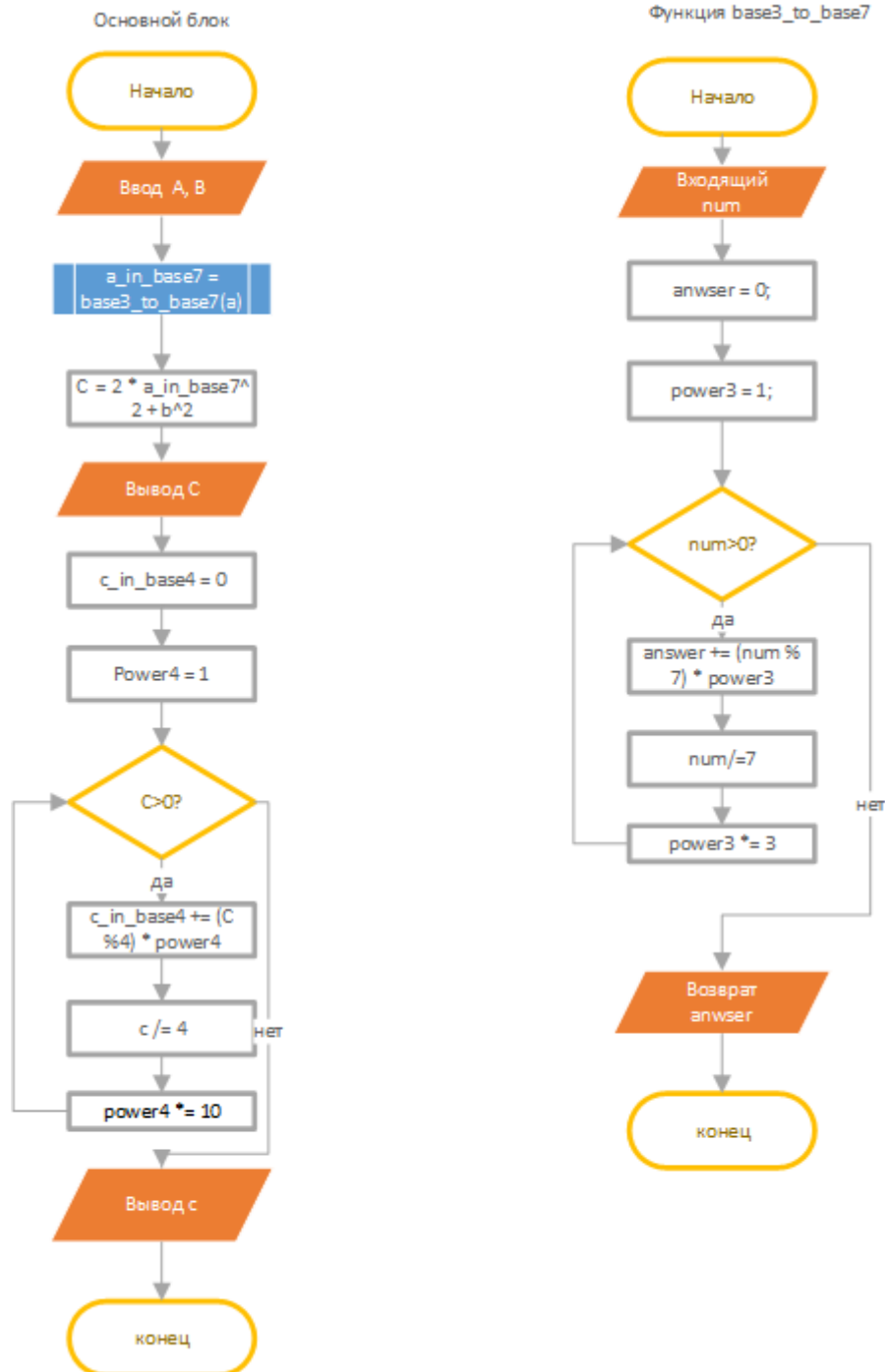
Task 1
FOR
Введите количество элементов: 5
Введите последовательность чисел:
15
25
35
45
55
Пара дружественных чисел не найдена.

```

Задание 2. Применение функций для вычислений в различных системах счисления

Разработать программу на языке C++ для решения следующей задачи. Заданы два числа — А и В, первое в системе счисления с основанием р, второе в системе счисления с основанием q. Вычислить значение С по указанной формуле и вывести его на экран в десятичной системе счисления и системе счисления с основанием r.

Блок-схема:



Код программы:

```
void Task2()
{
    int a, b;
    cout << "Число a в 3-ной системе: ";
    cin >> a;
    cout << "Число b в 7-ной системе: ";
    cin >> b;
    int c = calculate_c(a, b);
    cout << "C in decimal system: " << c << endl;

    // Перевод числа c в систему с основанием 4
    int c_in_base4 = 0;
    int power4 = 1;
    while (c > 0)
    {
        c_in_base4 += (c % 4) * power4;
        c /= 4;
        power4 *= 10;
    }

    cout << "C in base 4: " << c_in_base4 << endl;
}

// Функция для вычисления значения C
int calculate_c(int a, int b)
{
    int a_in_base7 = base3_to_base7(a);
    return 2 * (pow(a_in_base7, 2) + pow(b, 2));
}

// Функция перевода числа из системы с основанием 3 в систему с
основанием 7
int base3_to_base7(int num)
{
    int answer = 0;
    int power3 = 1;
```



```

while (num > 0)
{
    answer += (num % 7) * power3;
    num /= 7;
    power3 *= 3;
}
return answer;
}

```

Вывод программы:

```

Run: CodeSolution x
Active code page: 65001
Введите номер лабораторной работы (1-5) или 'q' д
Введите номер задания (1-3): 2
Число a в 3-ной системе: 12222
Число b в 7-ной системе: 653
C in decimal system: 1257818
C in base 4: 1713076530
Process finished with exit code 0

```

```

Run: CodeSolution x
Active code page: 65001
Введите номер лабораторной работы (1-5) или 'q' для выхода: 4
Введите номер задания (1-3): 2
Число a в 3-ной системе: 200000
Число b в 7-ной системе: 1234
C in decimal system: 13224584
C in base 4: 1482511300

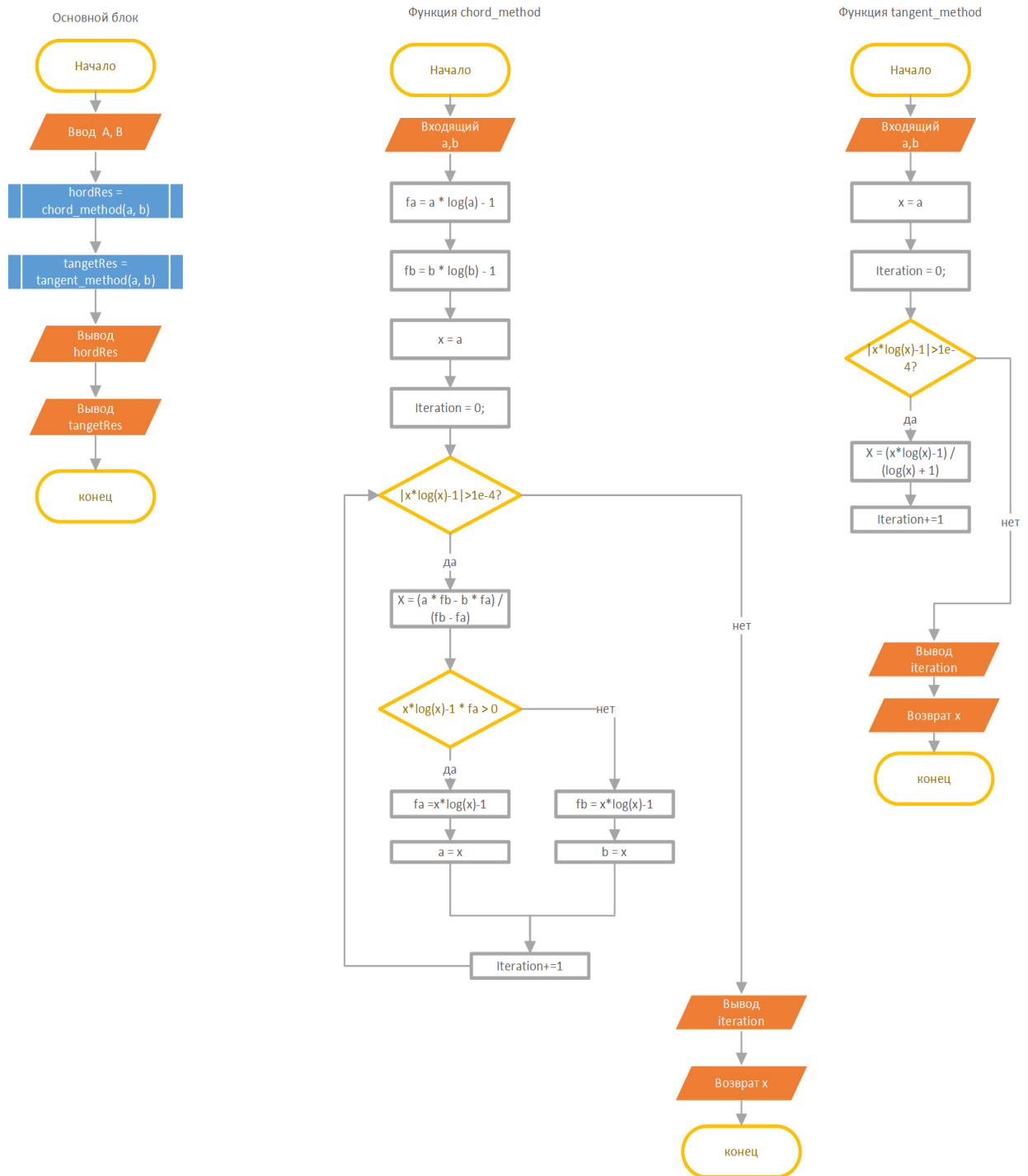
```

Задание 3. Применение функций для решения нелинейных уравнений

Разработать программу на языке C++ для вычисления одного из корней уравнения $f(x)=0$ методами, указанными в задании. Для решения задачи предварительно определить интервал изоляции корня графическим методом. Вычисления проводить с точностью $\varepsilon=10^{-4}$. Оценить степень точности путём подсчёта количества итераций, выполненных для достижения заданной точности.

9	$x \cdot \ln(x) - 1 = 0$	метод хорд, метод касательных
---	--------------------------	-------------------------------

Блок-схема:



Код программы:

```

void Task3()
{
    cout << endl
        << endl
        << "Task 3" << endl;
    double a, b;

```

```

    cout << "Введите начало и конец интервала: ";
    cin >> a >> b;
    double hordRes = chord_method(a, b);
    double tangetRes = tangent_method(a, b);
    cout << "Hord result = " << hordRes << endl;
    cout << "Tangent result = " << tangetRes << endl;
}

// метод касательных
double tangent_method(double a, double b)
{
    double x = a;
    int iteration = 0;
    while (abs(f(x)) > 1e-4)
    {
        x = x - f(x) / df(x);
        iteration++;
    }
    cout << "Метод касательных: ";
    cout << "Корень = " << x << ", достигнут за " << iteration << "
итераций" << endl;
    return x;
}

// метод хорд
double chord_method(double a, double b)
{
    double fa = f(a), fb = f(b), x = a;
    int iteration = 0;
    while (abs(f(x)) > 1e-4)
    {
        x = (a * fb - b * fa) / (fb - fa);
        if (f(x) * fa > 0)
        {
            fa = f(x);
            a = x;
        }
        else

```

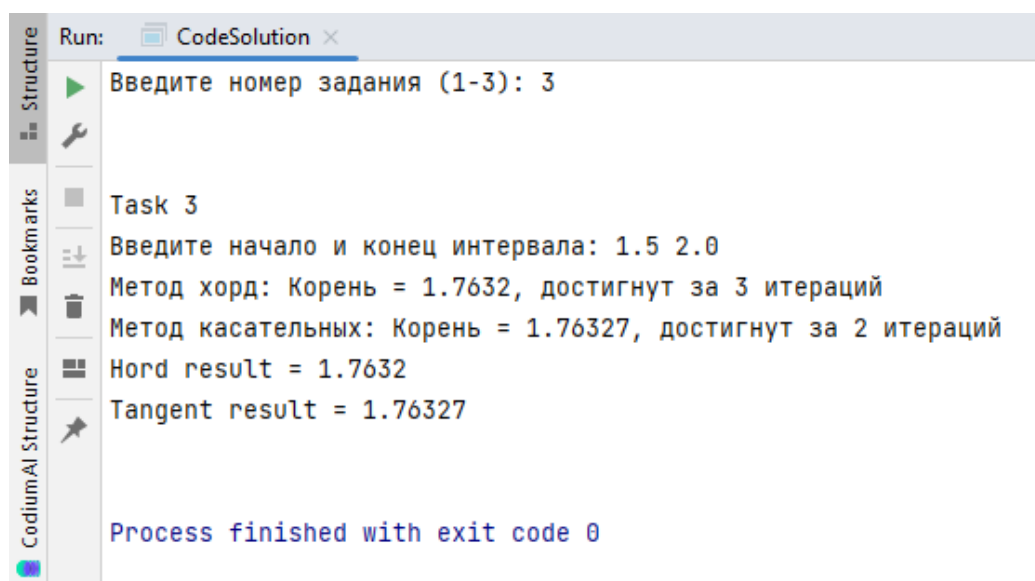
```

        {
            fb = f(x);
            b = x;
        }
        iteration++;
    }
    cout << "Метод хорд: ";
    cout << "Корень = " << x << ", достигнут за " << iteration << "
итераций" << endl;
    return x;
}
// функция f(x)
double f(double x)
{
    return x * log(x) - 1;
}

// производная функции f(x)
double df(double x)
{
    return log(x) + 1;
}

```

Вывод программы:



The screenshot shows a C++ IDE window titled "Run: CodeSolution x". The output console displays the following text:

```

Введите номер задания (1-3): 3

Task 3
Введите начало и конец интервала: 1.5 2.0
Метод хорд: Корень = 1.7632, достигнут за 3 итераций
Метод касательных: Корень = 1.76327, достигнут за 2 итераций
Hord result = 1.7632
Tangent result = 1.76327

Process finished with exit code 0

```

The IDE interface includes a sidebar with icons for Structure, Bookmarks, and Codium AI Structure.

Structure

Run: CodeSolution x

Введите номер задания (1-3): 3

Task 3

Введите начало и конец интервала: 0.4 0.7

Метод хорд: Корень = 1.76321, достигнут за 9 итераций

Метод касательных: Корень = 1.76322, достигнут за 6 итераций

Hor d result = 1.76321

Tangent result = 1.76322

Process finished with exit code 0

Вывод

После изучения принципов построения и использования функций в C++, можно сделать следующие выводы:

1. Функции играют ключевую роль в структурировании кода на языке C++. Они позволяют разделить сложную задачу на более простые подзадачи, что делает код более понятным и поддерживаемым.
2. Функции в C++ способствуют повторному использованию кода. Один и тот же блок кода можно выделить в функцию и вызывать его из разных частей программы, что сокращает объем кода и уменьшает вероятность ошибок.
3. Передача параметров в функцию может происходить разными способами: по значению, по ссылке или по указателю. Каждый метод имеет свои особенности и может быть использован в зависимости от ситуации.
4. В C++ возможна перегрузка функций, что позволяет использовать одно и то же имя функции для разных задач.
5. Шаблонные функции в C++ позволяют создавать функции, которые могут работать с различными типами данных. Это делает код более гибким и универсальным.
6. Lambda-функции в C++ предоставляют возможность создавать анонимные функции, которые могут быть использованы для передачи в другие функции или для создания функциональных объектов на лету.
7. Понимание принципов работы функций в C++ является важным шагом в освоении языка и развитии навыков программирования. Это знание позволит эффективно использовать функции для создания модульного, эффективного и переиспользуемого кода.