**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. И. РАЗЗАКОВА**

**Кафедра «Программное обеспечение компьютерных систем»**

**Отчет**

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ С++**

**Лабораторная работа №1**

**Выполнила: Тойчубекова Айчурок ПИ-2-21**

**Проверил: Мусабаев Э.Б.**

**Бишкек 2024**

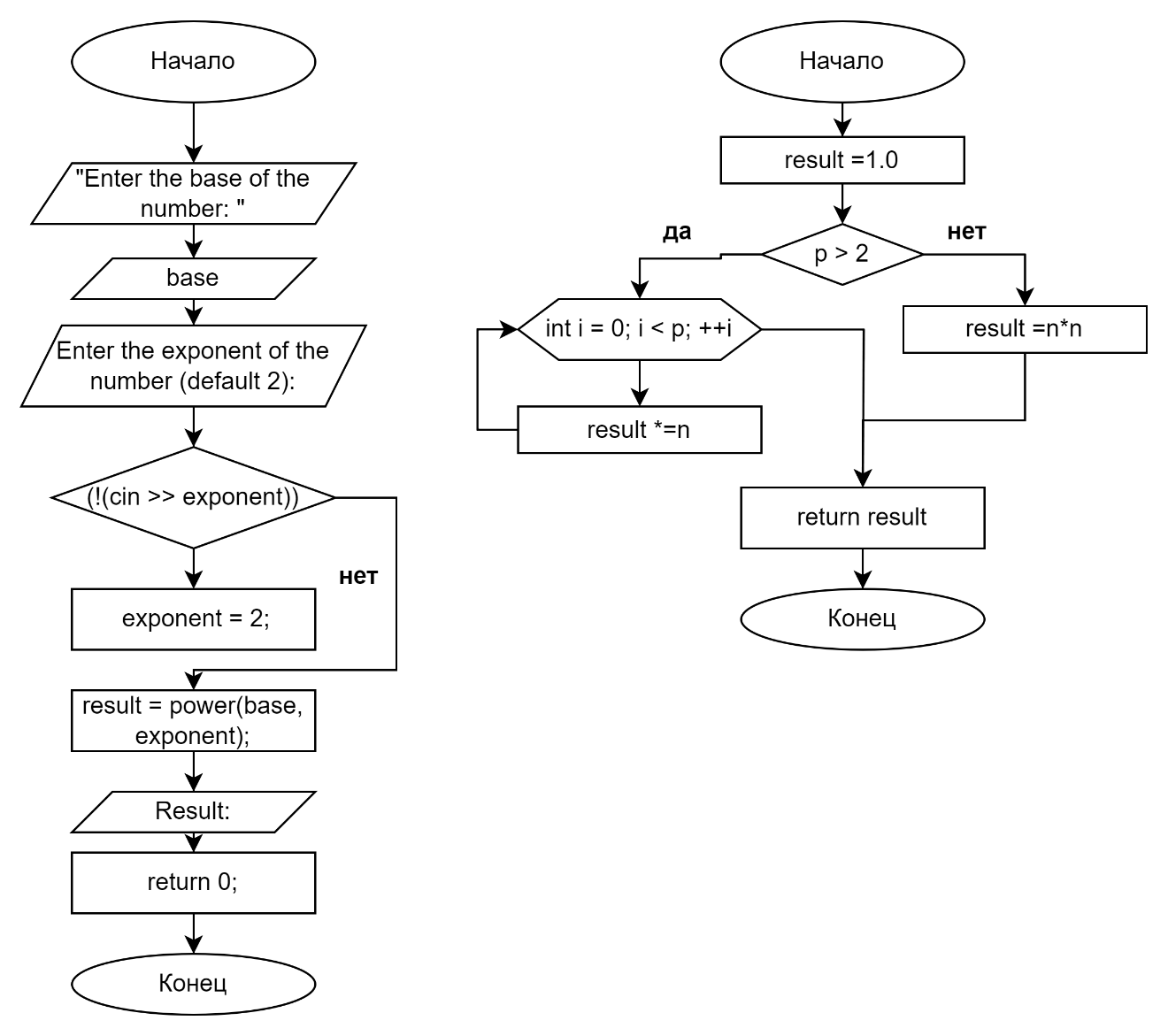
1. **Возведение числа n в степень p – это умножение числа n на себя p раз.**

**Напишите функцию (не библиотечную) с именем power (), которая в качестве аргументов принимает значение типа double для n и значение типа int для p и возвращает значение типа double.**

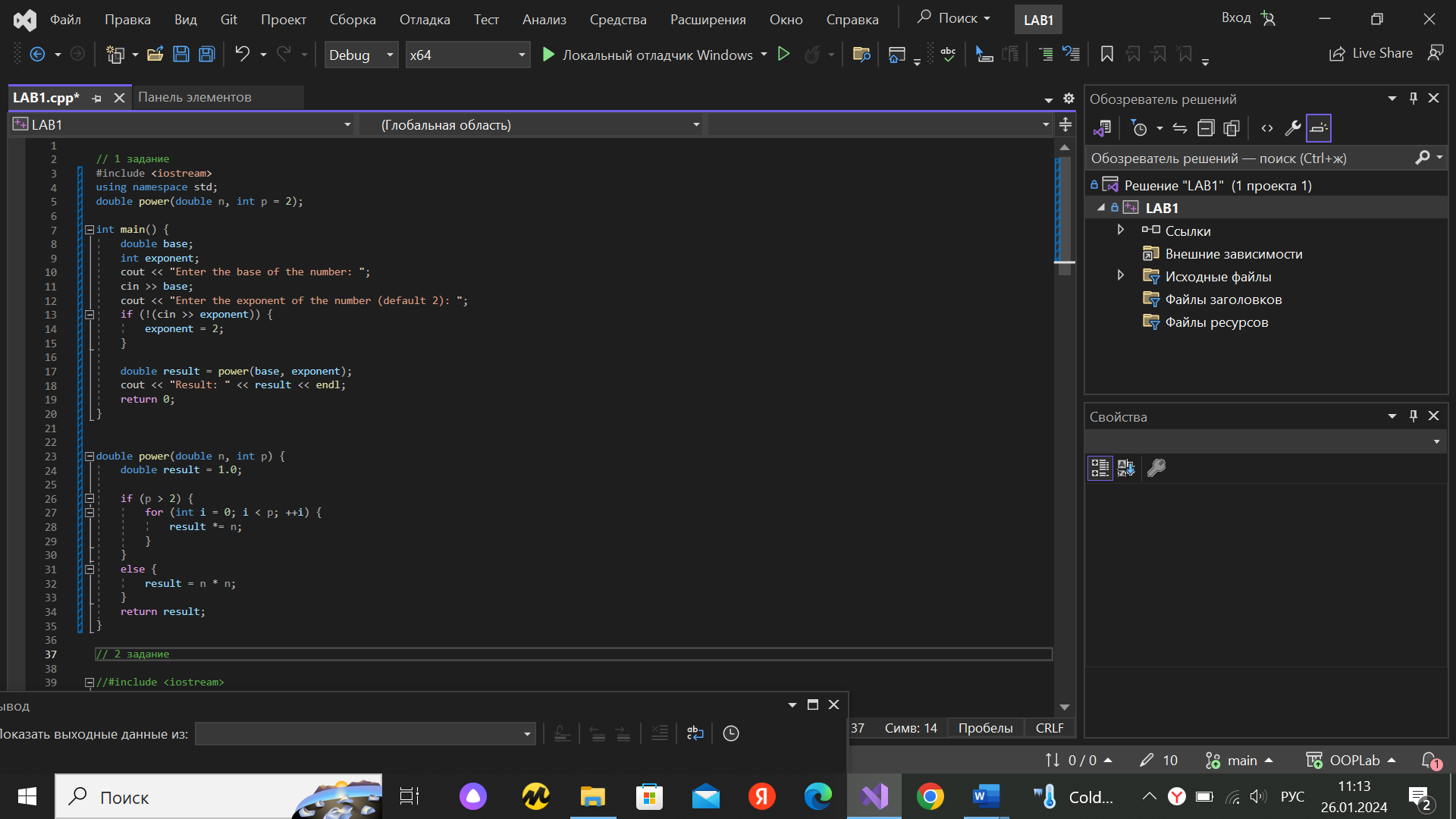
**Для аргумента, соответствующего степени числа, задать значение по умолчанию, равное 2, чтобы при отсутствии показателя степени при вызове функции число n возводилось в квадрат.**

**Сделайте проверку, будет ли пользователь вводить степень числа. Если будет, то эта степень – число, большее 2. Напишите функцию main(), которая запрашивает у пользователя ввод аргументов для функции power () и отображает на экране результаты её работы. Напишите код первого задания с *объявлением* (прототипом) функции power ().**

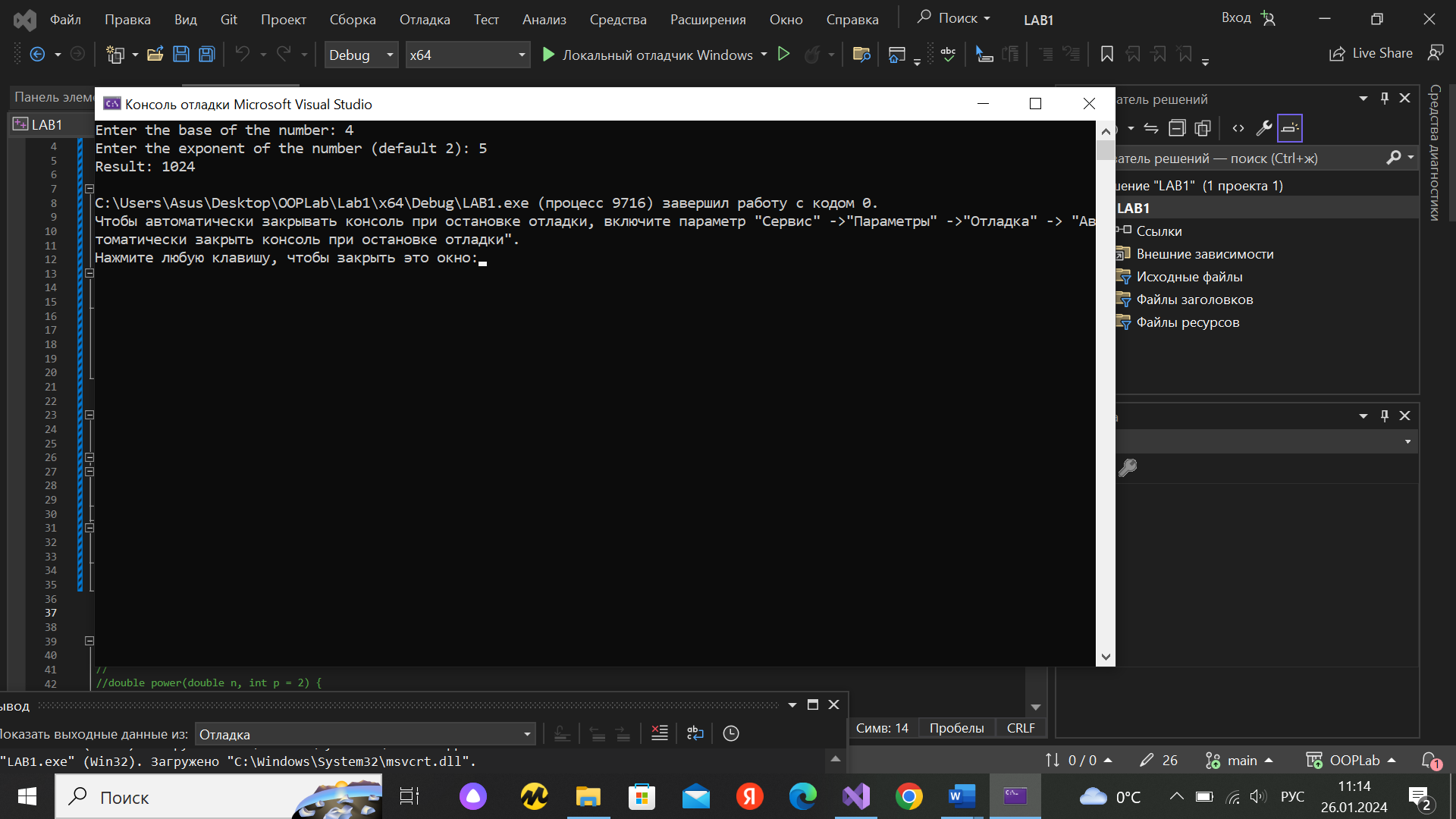
**Блок-схема:**

****

**Код:**

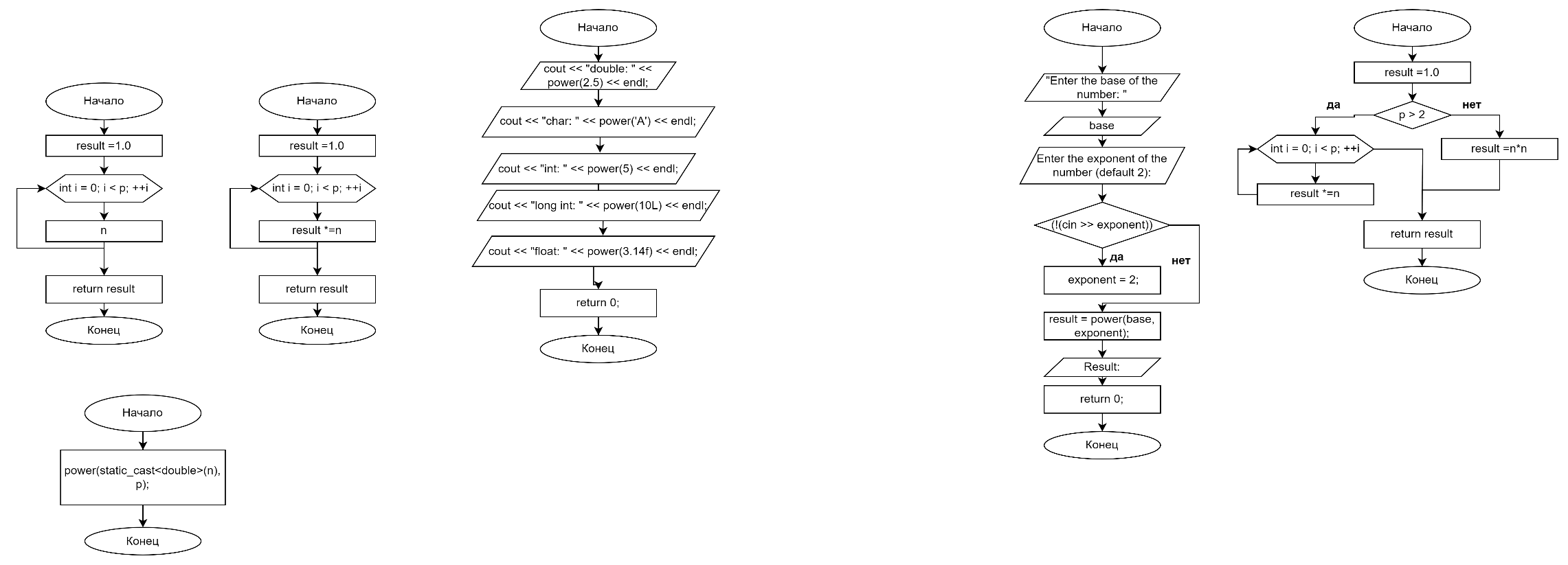


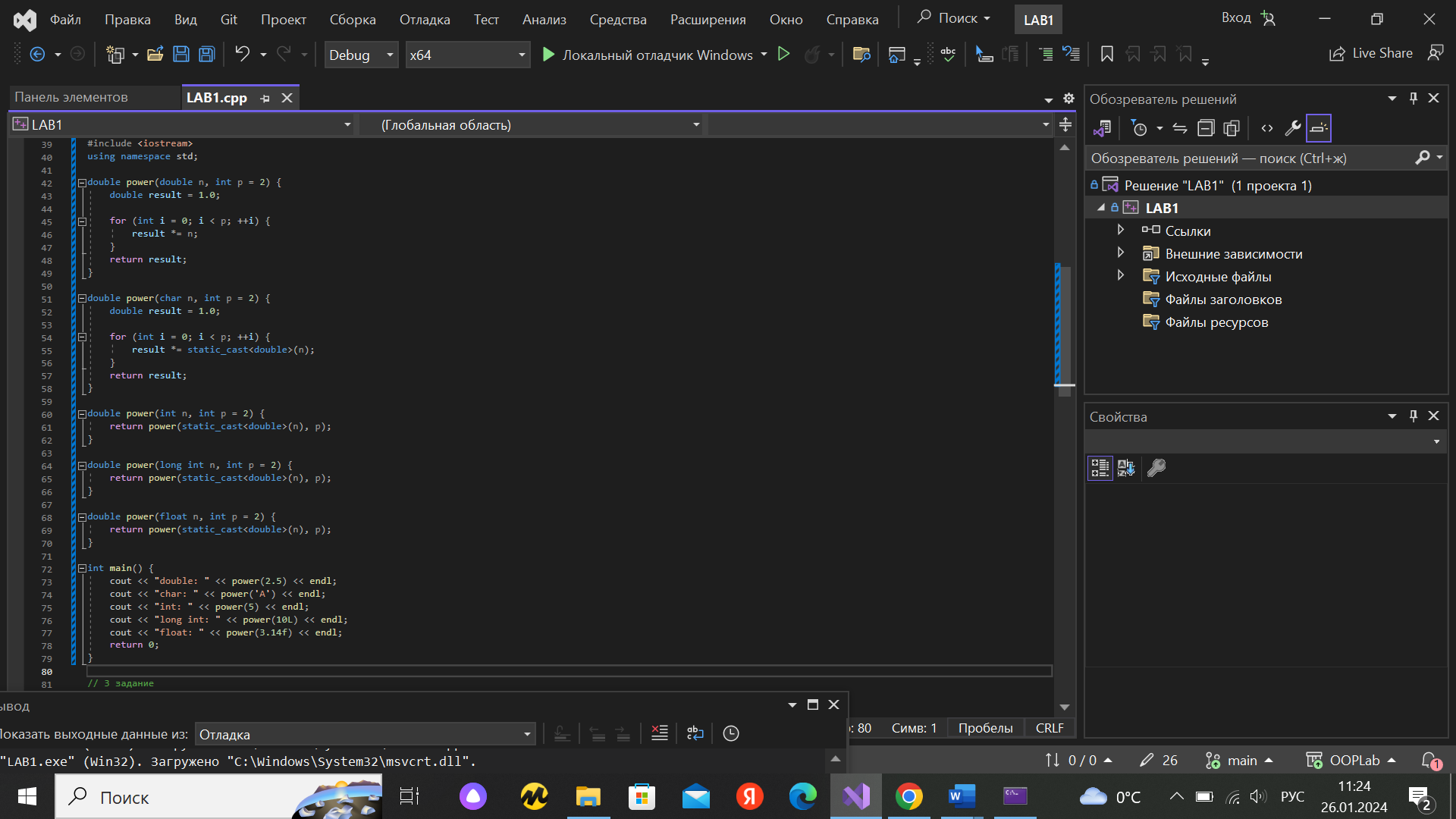
**Результат:**



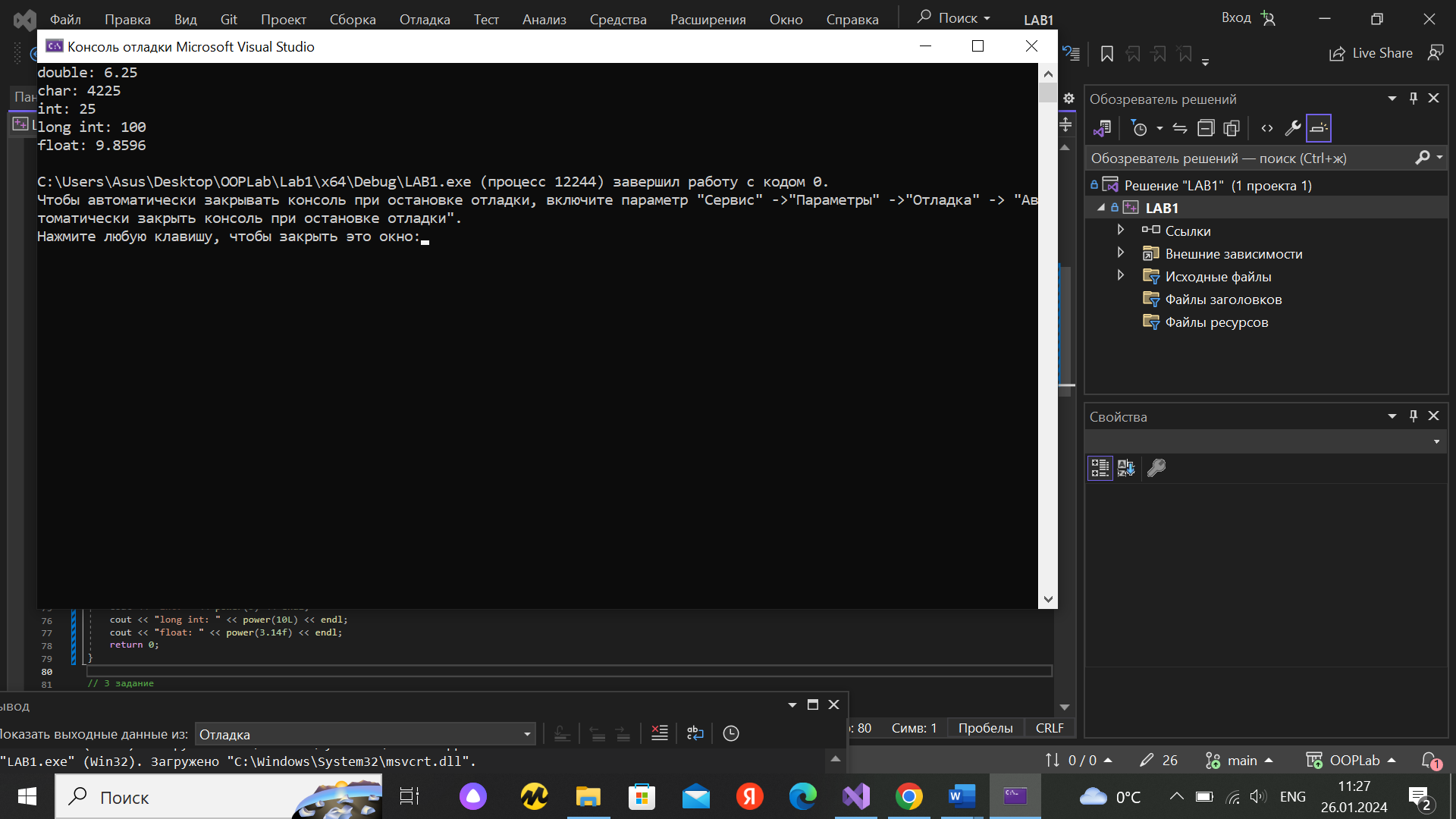
1. **Взяв в качестве основы функцию power () из первого задания, работающую только со значением типа double. Создайте перегруженные функции с этим же именем, принимающими в качестве аргумента значения типа char, short int, long int, float. Напишите программу, вызывающую функцию power () со всеми возможными типами аргументов. Напишите код второго задания без *объявления* функции power ().**

**Блок-схема:**

****

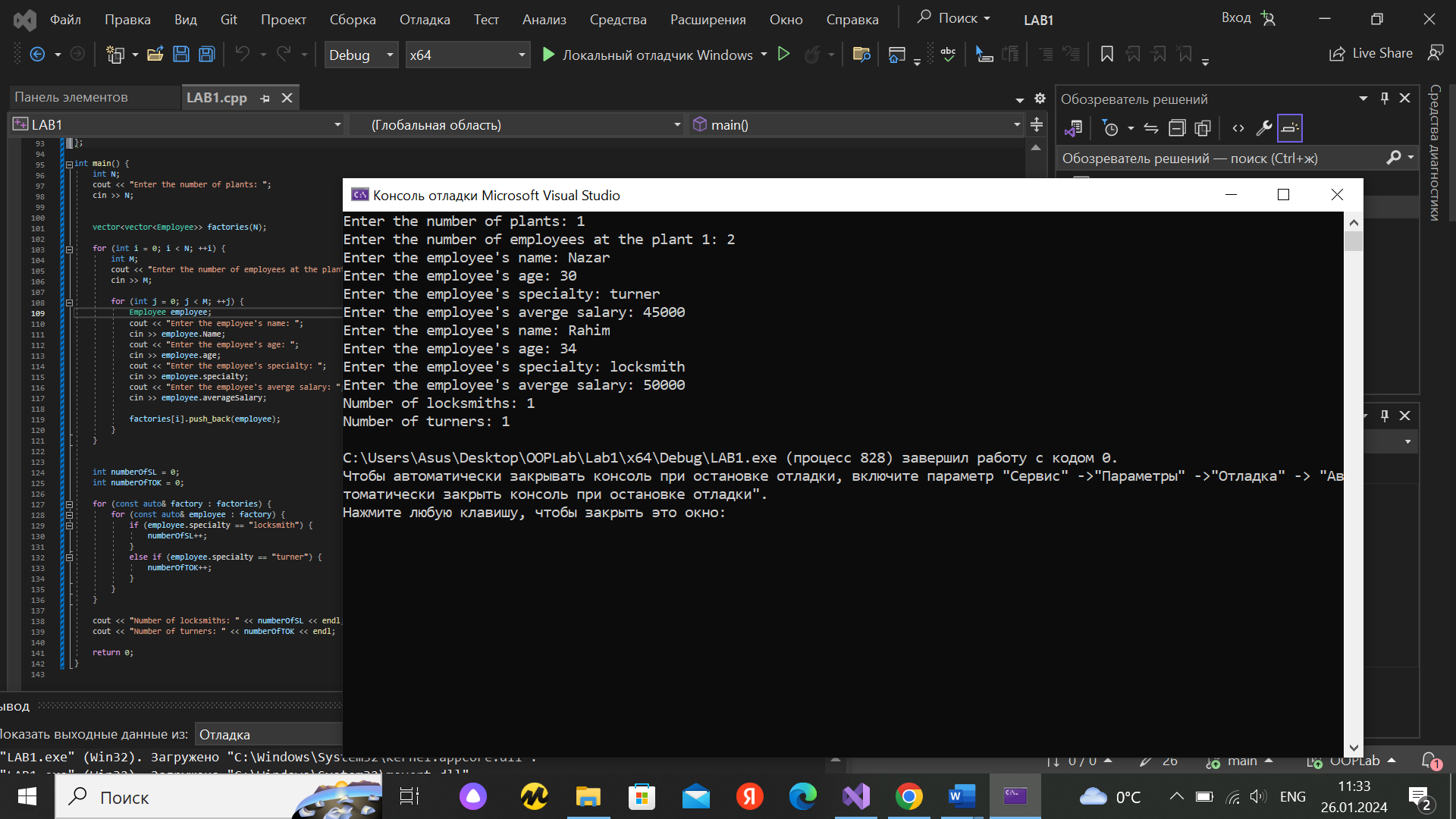


**Результат:**

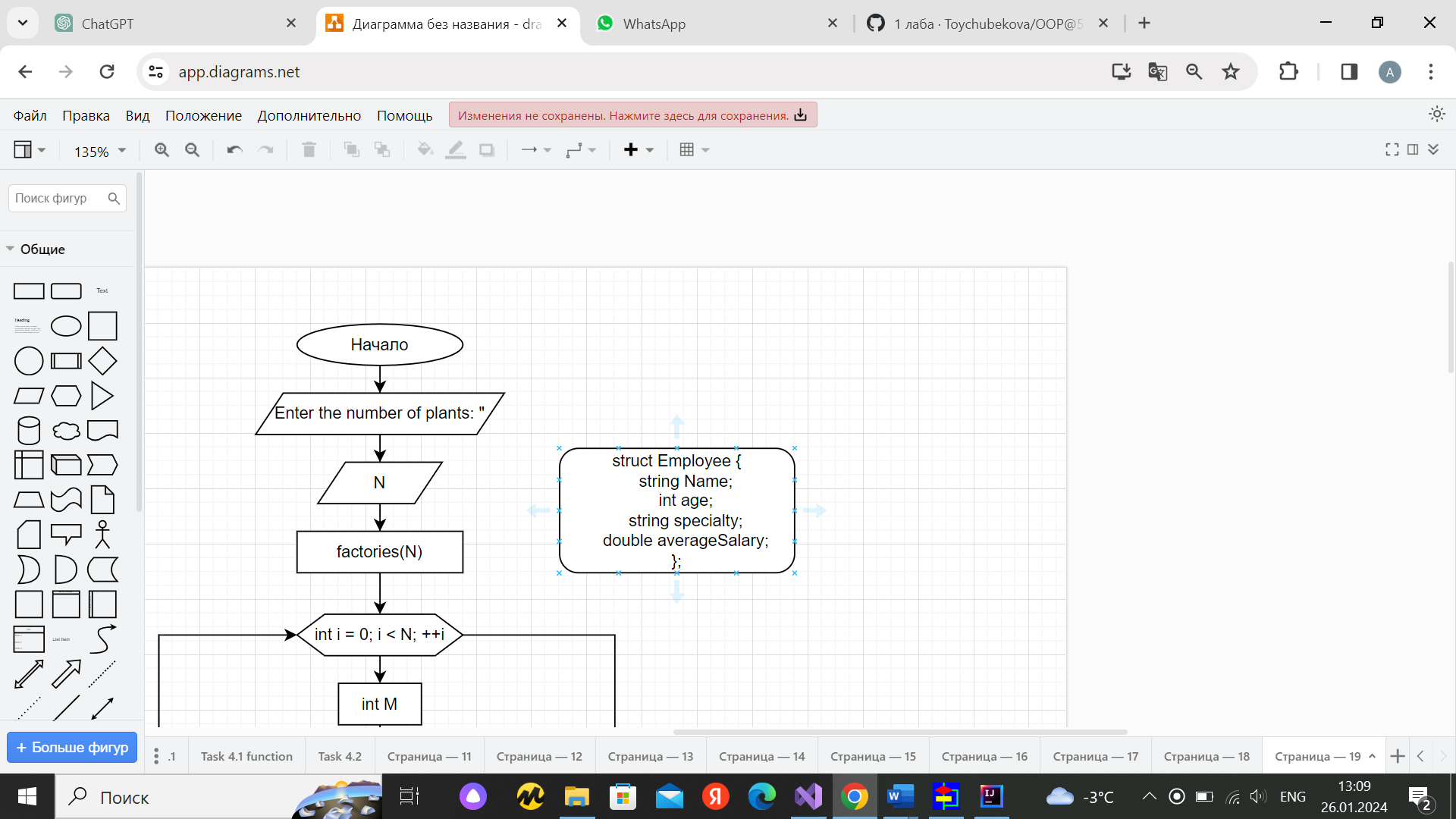


1. **Информация по работникам N заводов задается структурой с полями: фамилия, возраст, специальность, средний оклад. Ввести информацию по заводам, посчитать количество слесарей и токарей. Вывести эти значения на консоль.**
2. #include <iostream>
3. #include <vector>
4. #include <string>
5. using namespace std;
6. struct Employee {
7. string Name;
8. int age;
9. string specialty;
10. double averageSalary;
11. };
12. int main() {
13. int N;
14. cout << "Enter the number of plants: ";
15. cin >> N;
17. vector<vector<Employee>> factories(N);
18. for (int i = 0; i < N; ++i) {
19. int M;
20. cout << "Enter the number of employees at the plant " << i + 1 << ": ";
21. cin >> M;
22. for (int j = 0; j < M; ++j) {
23. Employee employee;
24. cout << "Enter the employee's name: ";
25. cin >> employee.Name;
26. cout << "Enter the employee's age: ";
27. cin >> employee.age;
28. cout << "Enter the employee's specialty: ";
29. cin >> employee.specialty;
30. cout << "Enter the employee's averge salary: ";
31. cin >> employee.averageSalary;
32. factories[i].push\_back(employee);
33. }
34. }
35. int numberOfSL = 0;
36. int numberOfTOK = 0;
37. for (const auto& factory : factories) {
38. for (const auto& employee : factory) {
39. if (employee.specialty == "locksmith") {
40. numberOfSL++;
41. }
42. else if (employee.specialty == "turner") {
43. numberOfTOK++;
44. }
45. }
46. }
48. cout << "Number of locksmiths: " << numberOfSL << endl;
49. cout << "Number of turners: " << numberOfTOK << endl;
50. return 0;
51. }

**Результат:**

****

**Блок-схема:**

****

**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ №1**

1. **Какие стандартные типы используются в С++? Сколько под них резервируется памяти?**

В C++, стандартные типы данных включают базовые типы и модификаторы, которые позволяют изменять свойства базовых типов. Вот некоторые из наиболее распространенных стандартных типов данных в C++:

**Базовые целочисленные типы:**

int: целое число, обычно 4 байта (32 бита).

short: короткое целое число, обычно 2 байта (16 бит).

long: длинное целое число, обычно 4 байта (32 бита).

long long: очень длинное целое число, обычно 8 байт (64 бита).

char: символ, обычно 1 байт.

**Базовые вещественные типы:**

float: число с плавающей точкой одинарной точности, обычно 4 байта.

double: число с плавающей точкой двойной точности, обычно 8 байт.

long double: число с плавающей точкой с расширенной точностью.

**Логический тип:**

bool: логический тип, обычно 1 байт.

**Символьные типы:**

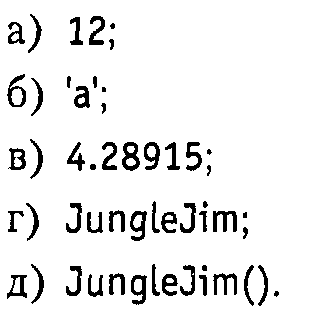
char: символ, обычно 1 байт.

wchar\_t: широкий символ, размер может варьироваться в зависимости от компилятора и платформы.

**Тип void:**

void: используется для указания на отсутствие типа.

1. **К каким элементам программы относятся следующие:**



a) 12; - это целочисленная константа.

б) 'a'; - это символьная константа, которая представляет собой литерал типа char.

c) 4.28915; - это вещественная константа, которая может быть литералом типа double.

г) JungleJim; - это идентификатор (имя переменной или другого объекта).

д) JungleJim(); - это вызов функции. Если JungleJim - это функция, то эта строка вызывает эту функцию.

1. **Что такое функция? Какова роль функций в языке С++?**

Функция эти блоки кода, выполняющие определенные операции. Функции в C++ являются основными строительными блоками программы, их использование способствует разделению кода на логические блоки, облегчает поддержку, повторное использование и улучшает читаемость кода.

1. **Напишите синтаксис функции.**

возвращаемый\_тип имя\_функции(параметры) {

// тело функции

// инструкции

// операторы

return значение; // (если функция возвращает значение)

}

1. **Опишите способы использования функций в программах с объявлением функций и без объявления функций.**

Объявление функции предваряет ее определение и дает компилятору информацию о том, как функция будет вызываться и какие типы параметров и возвращаемого значения она использует.

// Объявление функции

int add(int a, int b);

int main() {

int result = add(3, 4); // Использование функции

cout << "Result: " << result << endl;

return 0;

}

// Определение функции

int add(int a, int b) {

return a + b;

}

Если функция определена до ее использования в коде, то объявление может быть опущено.

int main() {

int result = add(3, 4); // Использование функции

// Определение функции перед использованием

int add(int a, int b) {

return a + b;

}cout << "Result: " << result << std::endl;

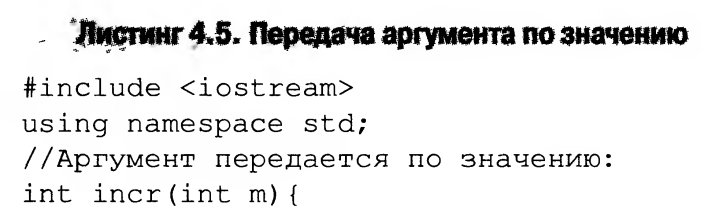
return 0;

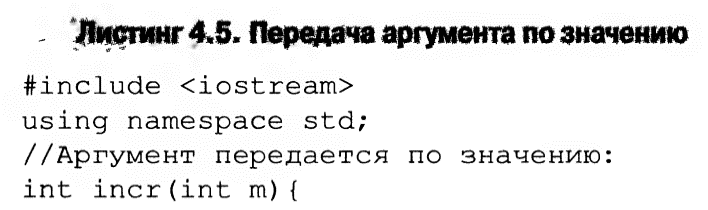
}

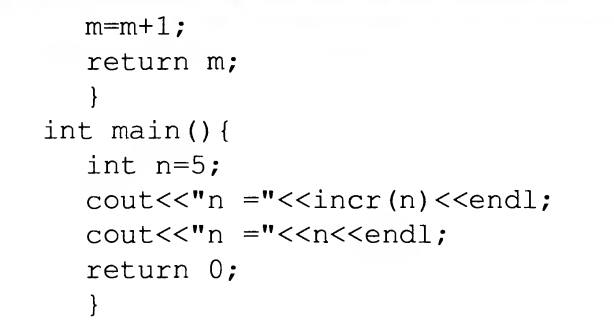
1. **Объясните механизмы передачи аргументов по значению и по ссылке в функцию.**

**Объясните результаты работы программ.**

1)







2))





1)По значению:

При передаче аргументов по значению в функцию создается копия значения аргумента, и функция работает с этой копией. Исходное значение переменной, переданной в функцию, не изменяется после завершения функции.

В этой программе используется передача по значению. Функция incr получает копию значения переменной n, увеличивает ее на 1, но это изменение не влияет на исходное значение n. Поэтому результатом будет вывод n = 6, но значение переменной n в main останется равным 5.

2)По ссылке:

При передаче аргументов по ссылке в функцию передается адрес памяти (ссылка) на переменную, а не ее копия. Функция работает с самой переменной, и изменения, внесенные внутри функции, затрагивают исходное значение переменной в вызывающей программе.

В этой программе используется передача по ссылке. Функция incr принимает переменную n по ссылке, и изменения, внесенные внутри функции, напрямую влияют на исходное значение n. Поэтому результатом будет вывод n = 6, и значение переменной n в main также станет равным 6.

1. **Назовите разновидности аргументов, которые могут быть переданы параметрам функции?**

В C++ аргументы могут быть переданы параметрам функции различными способами, включая:

1. По значению (pass by value):

- Значение аргумента копируется в параметр функции.

- Локальная копия параметра изменяется внутри функции, но не влияет на оригинальное значение.

```cpp

void func(int x) {

// x - копия аргумента

}

```

2. По ссылке (pass by reference):

- Адрес аргумента передается в параметр функции.

- Изменения, внесенные в параметр функции, отражаются на оригинальном значении аргумента.

```cpp

void func(int &x) {

// x - ссылка на аргумент

}

```

3. По указателю (pass by pointer):

- Указатель на аргумент передается в параметр функции.

- Функция имеет доступ к оригинальному значению через указатель.

```cpp

void func(int \*ptr) {

// ptr - указатель на аргумент

}

```

4.По константной ссылке (pass by const reference):

- Аналогично передаче по ссылке, но параметр объявлен как `const`, что предотвращает изменение значения аргумента внутри функции.

```cpp

void func(const int &x) {

// x - константная ссылка на аргумент

}

```

5. По значению по умолчанию:

- Параметру функции присваивается значение по умолчанию, если вызывающий код не предоставляет аргумент.

- Обычно используется для определения необязательных параметров.

```cpp

void func(int x = 10) {

// x - значение аргумента или значение по умолчанию

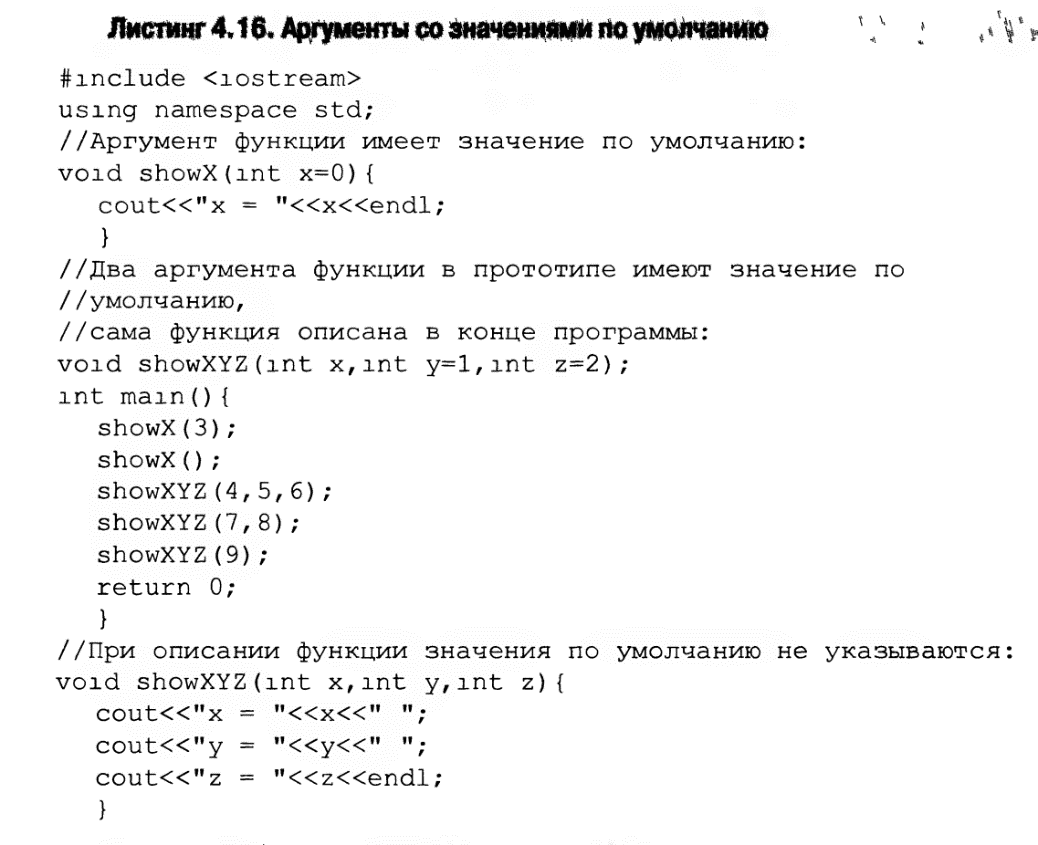
}

```

Выбор метода передачи аргументов зависит от требований задачи и необходимости изменения или защиты оригинальных значений.

1. **Что такое аргументы по умолчанию?**

**Объясните результат работы программы:**

,

Аргументы по умолчанию - это значения, которые предоставляются параметрам функции при ее объявлении, и эти значения используются, если вызывающий код не предоставил соответствующий аргумент при вызове функции. Они обеспечивают возможность определения параметров функции, которые необязательны для указания при вызове.

int main() {

showX(3); // Вывод: x = 3

showX(); // Вывод: x = 0

showXYZ(4, 5, 6); // Вывод: x = 4 y = 5 z = 6

showXYZ(7, 8); // Вывод: x = 7 y = 8 z = 2

showXYZ(9); // Вывод: x = 9 y = 1 z = 2

**Объяснение:**

showX(3);: Вызов функции showX с аргументом 3. Выводится "x = 3".

showX();: Вызов функции showX без аргумента. Используется значение по умолчанию (0). Выводится "x = 0".

showXYZ(4, 5, 6);: Вызов функции showXYZ с явными значениями для всех параметров. Выводится "x = 4 y = 5 z = 6".

showXYZ(7, 8);: Вызов функции showXYZ с явным значением для x и y, а значение z используется по умолчанию (2). Выводится "x = 7 y = 8 z = 2".

showXYZ(9);: Вызов функции showXYZ с явным значением только для x, значения y и z используются по умолчанию. Выводится "x = 9 y = 1 z = 2".

1. **Что такое перегрузка функций?**

Перегрузка функций - это возможность определения нескольких функций с одним и тем же именем в пределах одного области видимости, но с разными списками параметров. В языке программирования C++, функции могут быть перегружены по типам и/или количеству параметров.

Основные характеристики перегрузки функций:

То же имя, разные параметры: Перегруженные функции имеют одно и то же имя, но различаются по типам и/или количеству параметров.

Возвращаемый тип не является частью сигнатуры: То есть, функции с одним и тем же именем не могут отличаться только возвращаемым типом.

1. **Чем отличается структура (struct) от массива?**

Структура - это пользовательский тип данных, который может содержать переменные различных типов данных, объединенные под общим именем.

Массив - это упорядоченный набор элементов одного и того же типа данных.

**Отличия**:

Типы данных: В структуре элементы могут иметь разные типы данных, в массиве элементы имеют одинаковый тип.

Доступ к данным: В структуре доступ к данным осуществляется по именам полей, в массиве - по индексам.

Размер: Размер массива фиксирован, определен при объявлении. Размер структуры определяется суммой размеров ее полей.