**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И. РАЗЗАКОВА**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**ОТЧЕТ**

**по лабораторной работе №1**

**Выполнила:** студент группы ПИ-3-21

Джанышбекова Акмарал

**Проверила:** Мусабаев Э. Б.

**Бишкек 2024**

**Задание №1**

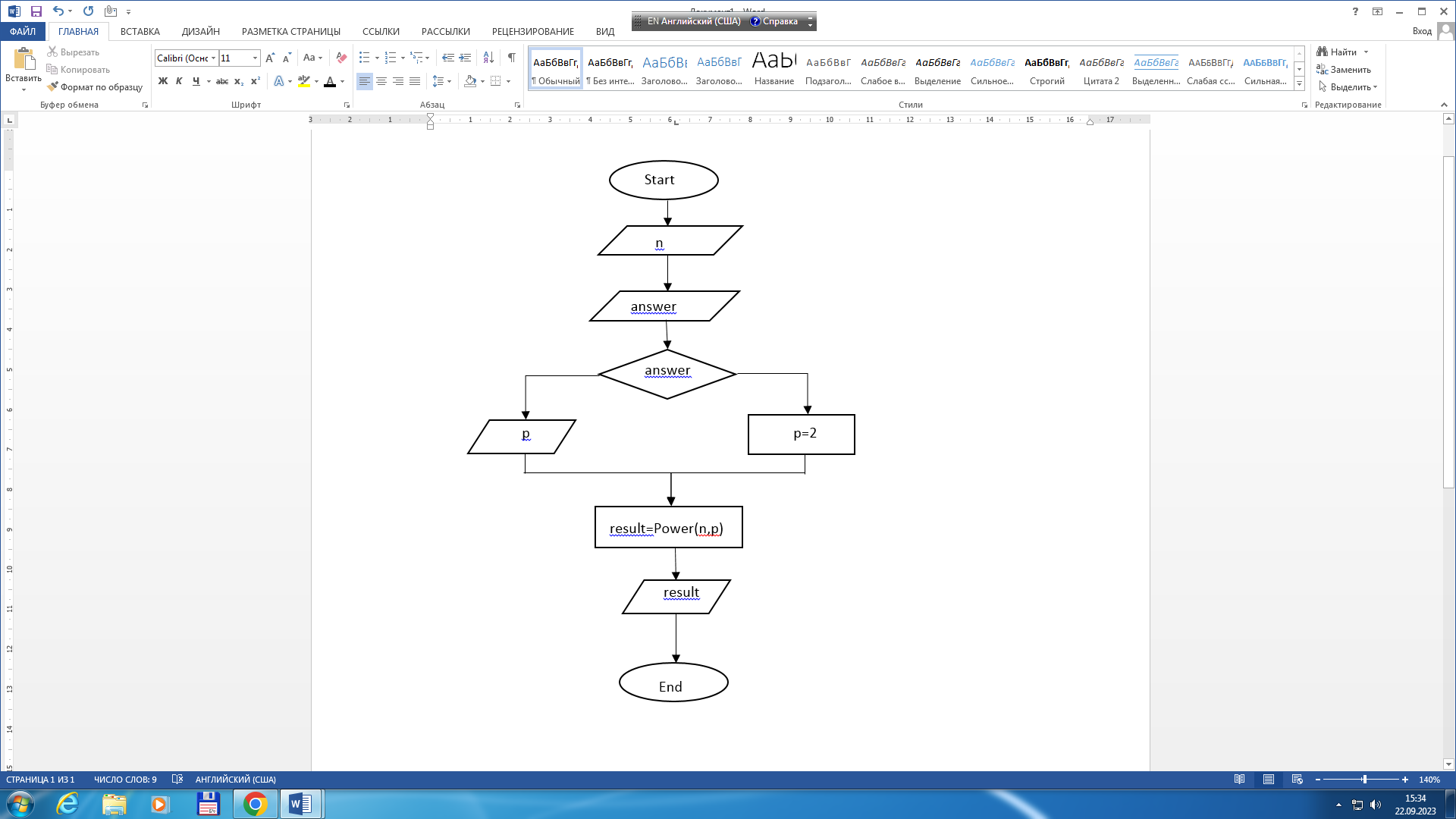
Возведение числа **n** в степень **p** – это умножение числа **n** на себя **p** раз. Напишите функцию (не библиотечную) с именем **power** (), которая в качестве аргументов принимает значение типа **double** для **n** и значение типа **int** для **p** и возвращает значение типа **double**. Для аргумента, 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 аргуменыыыыыытов для функции **power** () и отображает на экране результаты её работы. Напишите код первого задания с *объявлением* (прототипом) функции power ().

**Problem statement:**

*Входные данные:* число n, ответ (0 или 1), степень этого числа p

*Выходные данные:* число возведённое в степень

**Блок-схема:**



**Код программы:**

#include <iostream>

using namespace std;

double Power(double n, int p = 2);

int main()

{

double n;

int p, answer;

cout << "Input N:\n";

cin >> n;

cout << "Will you input the power? (0-No; 1-Yes)\n";

cin >> answer;

if (answer) {

cout << "Input number >=2 : \n";

cin >> p;

}

else p = 2;

double result = Power(n, p);

cout << "Result : " << result;

return 0;

}

double Power(double n, int p = 2) {

double result = 1;

for (int i = 0; i < p; i++) {

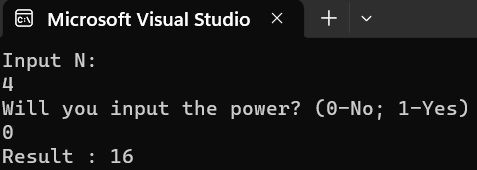
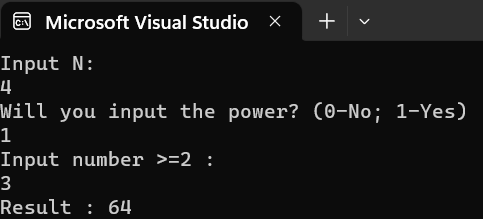
result = result \* n;

}

return result;

}

**Результат работы программы:**



**Задание №2**

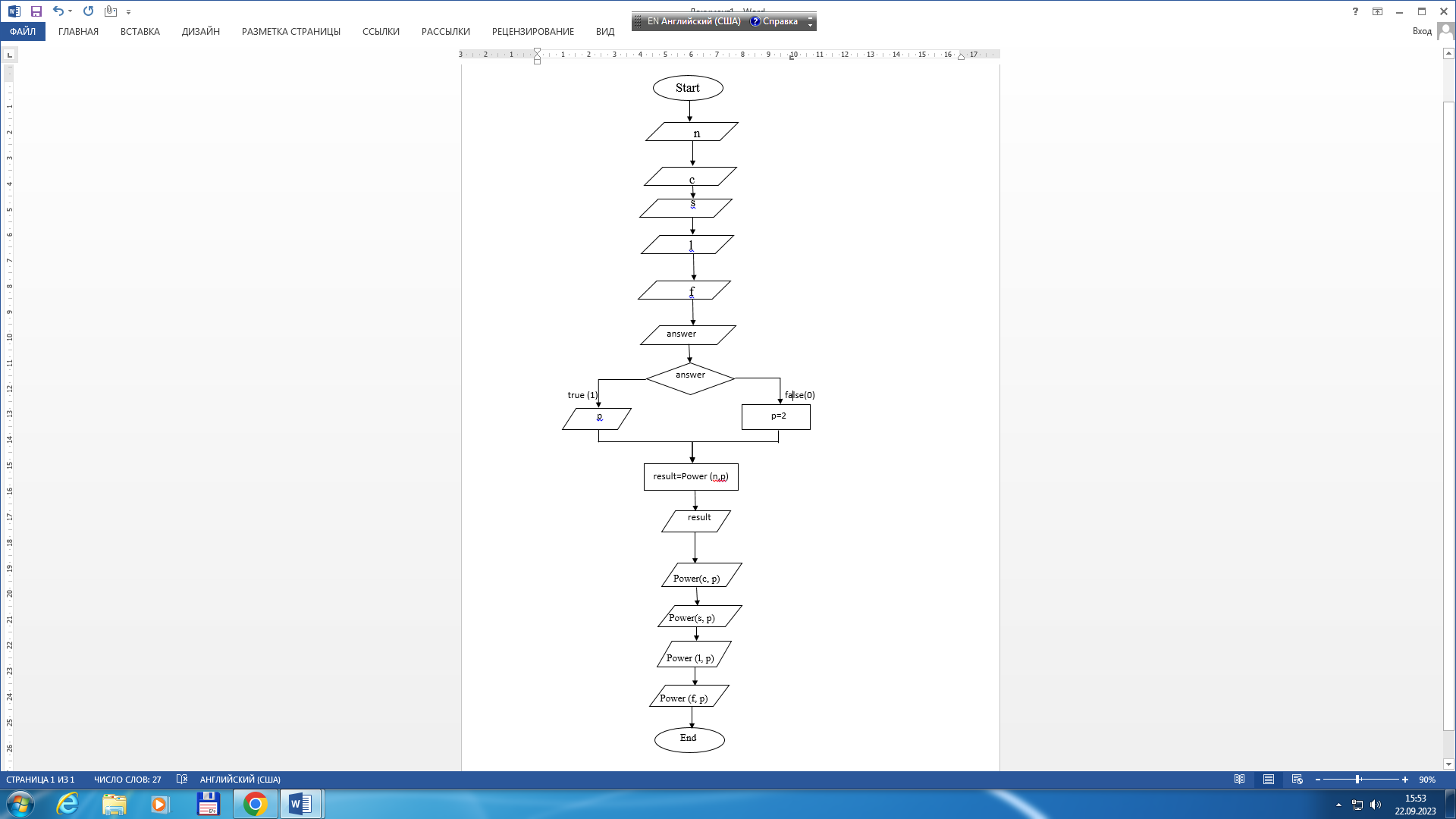
Взяв в качестве основы функцию **power** () из первого задания, работающую только со значением типа **double**. Создайте перегруженные функции с этим же именем, принимающими в качестве аргумента значения типа **char**, short **int**, **long int, float**. Напишите программу, вызывающую функцию **power**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

**Problem statement:**

*Входные данные:* число n (**double)**, число s (**short int),** число l (**long int)**, символ c (**char)**, ответ (0 или 1), степень этого числа p

*Выходные данные:* число или символ, возведённый в степень

**Блок-схема:**



**Код программы:**

#include <iostream>

using namespace std;

double Power(double n, int p = 2) {

double result = 1;

for (int i = 0; i < p; i++) {

result = result \* n;

}

return result;

}

short int Power(short int n, int p = 2) {

short int result = 1;

for (int i = 0; i < p; i++) {

result = result \* n;

}

return result;

}

long int Power(long int n, int p = 2) {

long int result = 1;

for (int i = 0; i < p; i++) {

result = result \* n;

}

return result;

}

float Power(float n, int p = 2) {

float result = 1;

for (int i = 0; i < p; i++) {

result = result \* n;

}

return result;

}

double Power(char n, int p = 2) {

double result = 1;

static\_cast<double>(n);

for (int i = 0; i < p; i++) {

result = result \* n;

}

return result;

}

int main() {

double n;

int p, answer;

char c;

short int s;

long int l;

float f;

cout << "Input double:\n";

cin >> n;

cout << "Input char:\n";

cin >> c;

cout << "Input short:\n";

cin >> s;

cout << "Input long:\n";

cin >> l;

cout << "Input float:\n";

cin >> f;

cout << "Will you input the power? (0-No; 1-Yes)\n";

cin >> answer;

if (answer) {

cout << "Input number >= 2 : \n";

cin >> p;

}

else {

p = 2;

}

double result = Power(n, p);

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

cout << "Result for char: " << Power(c, p) << endl;

cout << "Result for short int: " << Power(s, p) << endl;

cout << "Result for long int: " << Power(l, p) << endl;

cout << "Result for float: " << Power(f, p) << endl;

return 0;

}

**Результат работы программы:**



**Задание №3**

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.

**Problem statement:**

*Входные данные:* число работников **numEmployees** (**double)**, данные о сотрудниках (**struct Employee)**: фамилия (**string)**, возраст (**int)**, специальность (**string)**, средний оклад (**double).**

*Выходные данные:* количество слесарей и токарей (**int**, **int)**.

**Код программы:**

#include <iostream>

#include <string>

using namespace std;

struct Employee {

string lastName;

int age;

string specialty;

double averageSalary;

};

int main() {

const int N = 3;

Employee employees[N][100];

for (int i = 0; i < N; i++) {

int numEmployees;

cout << "Input number of employees for the factory " << i + 1 << ": ";

cin >> numEmployees;

for (int j = 0; j < numEmployees; j++) {

cout << "Input the informatio about employee " << j + 1 << " on the factory " << i + 1 << ":\n";

cout << "Last name: ";

cin >> employees[i][j].lastName;

cout << "Age: ";

cin >> employees[i][j].age;

cout << "Specialty: ";

cin >> employees[i][j].specialty;

cout << "Average salary: ";

cin >> employees[i][j].averageSalary;

}

}

int numMechanics = 0;

int numTurners = 0;

for (int i = 0; i < N; i++) {

for (int j = 0; j < 100; j++) {

if (employees[i][j].specialty == "mechanic") {

numMechanics++;

}

else if (employees[i][j].specialty == "turner") {

numTurners++;

}

}

}

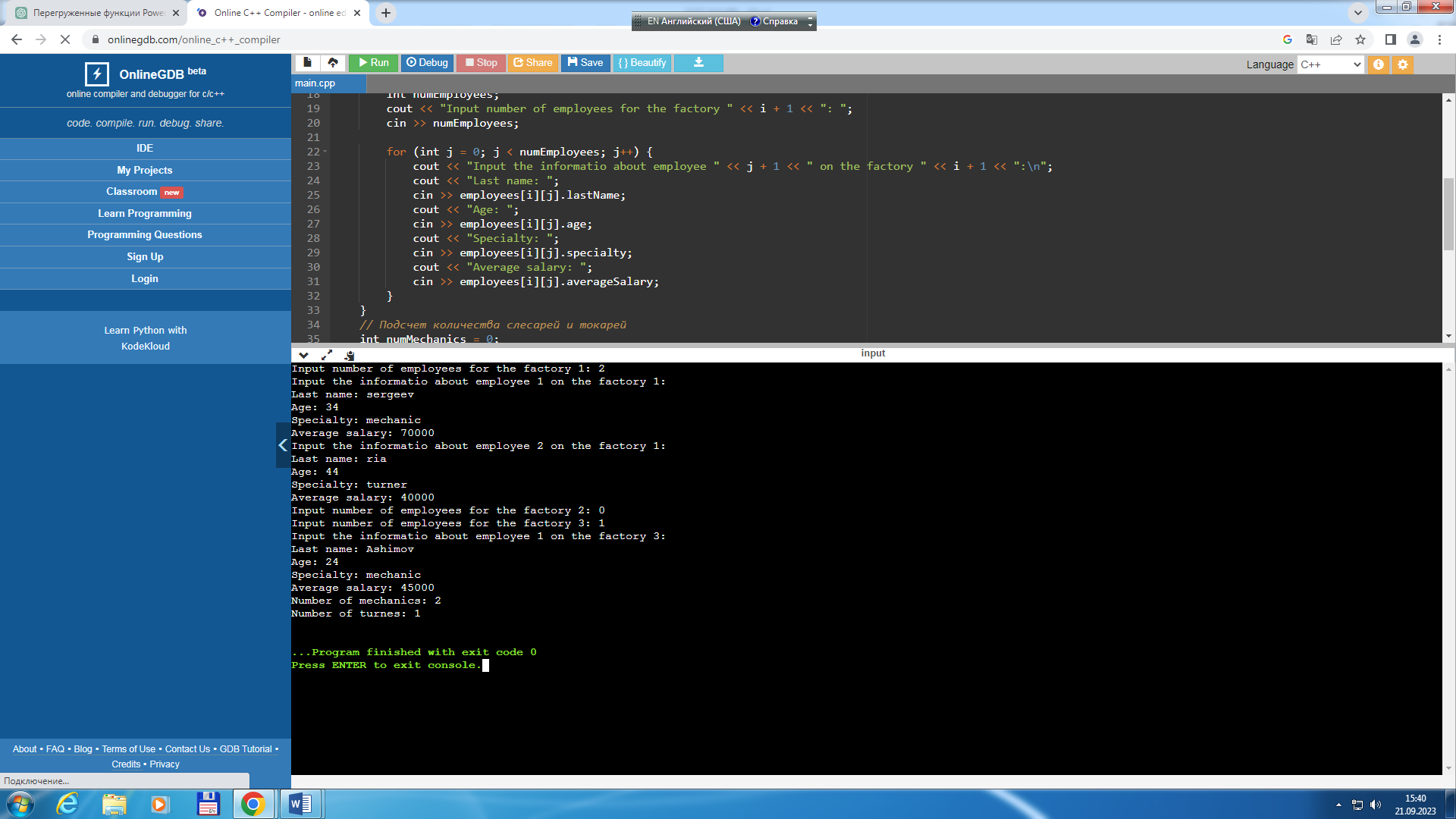
cout << "Number of mechanics: " << numMechanics << endl;

cout << "Number of turnes: " << numTurners << endl;

return 0;

}

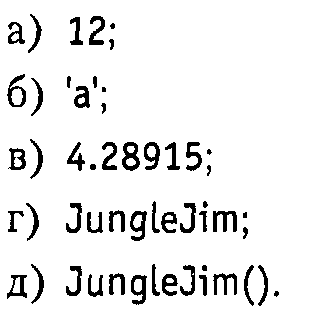
**Результат работы программы:**



**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ Р АБОТЫ №1**

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

* **int:** Тип данных для хранения целых чисел. 4 байта (32 бита)
* **float:** Тип данных для хранения чисел с плавающей точкой одинарной точности. Размер обычно составляет 4 байта.
* **double:** Тип данных для хранения чисел с плавающей точкой двойной точности. Размер обычно составляет 8 байт.
* **char:** Тип данных для хранения символов. Размер всегда равен 1 байту.
* **bool:** Тип данных для хранения логических значений *true* или *false*. Размер обычно равен 1 байту.
* **long:** Тип данных для хранения целых чисел большего размера, чем int. Размер зависит от архитектуры, но обычно 4 байта на 32-битных системах и 8 байт на 64-битных системах.

1. *К каким элементам программы относятся следующие:*

а) 12 - это литерал целого числа (integer literal) и представляет собой конкретное значение типа **int.**

б) 'a' - это символьный литерал (character literal) и представляет собой одиночный символ типа **char**.

в) 4.28915 - это литерал числа с плавающей точкой (floating-point literal) и представляет собой значение с плавающей точкой типа **double** по умолчанию. Если нужно использовать тип **float,** можно добавить суффикс f, например, 4.28915f.

г) JungleJim - это **идентификатор** (identifier), который может быть именем переменной, функции, класса или другого объекта в программе. Это зависит от контекста и объявления этого идентификатора в программе.

д) JungleJim() - это **конструктор**, который используется для создания объекта класса с именем JungleJim. Это объявление конструктора, и оно может быть частью класса.

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

*Функция* — это кусок кода, который выполняет определенную задачу. Она представляет собой набор инструкций, которые выполняются в определенном порядке при вызове функции. Роль функций в C++:

**Группировка кода:** Функции позволяют разделить программу на более мелкие, управляемые блоки кода. Это делает код более организованным и легким для понимания.

**Повторное использование кода:** Вы можете вызывать функции в разных частях программы, что позволяет избегать дублирования кода. Если вам нужно выполнить ту же задачу несколько раз, вы просто вызываете функцию снова, вместо того чтобы писать один и тот же код каждый раз.

1. *Напишите синтаксис функции.*
2. Прототип
3. Main()
4. Определение функции

возвращаемый\_тип имя\_функции(параметры) {

// Тело функции

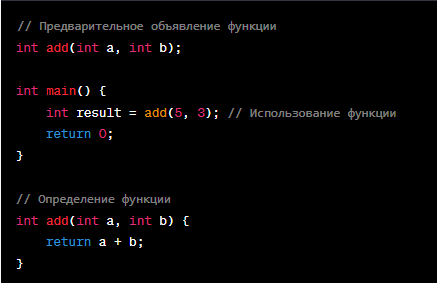
// Здесь размещаются инструкции, выполняемые функцией

return значение; // Необязательно, если функция возвращает значение

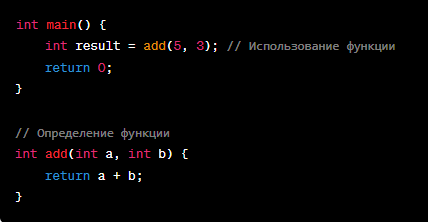
}

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

С объявлением функций мы предварительно объявляем функции перед их фактическим определением в программе:



При использовании функции без объявления компилятор анализирует функции в том порядке, в котором они появляются в коде:



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

*Объясните результаты работы программ.*

|  |  |
| --- | --- |
| 1. **Передача аргументов по значению:** | 1. **Передача аргументов по ссылке:** |
| В этом случае функция получает копию переданного аргумента. Внутри функции изменения параметра не влияют на оригинальное значение переменной. | В этом случае функция получает ссылку на оригинальную переменную, и изменения внутри функции влияют на саму переменную. |
|  |  |
| Результат работы этой программы будет: | |
|  |  |

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

**Аргументы по значению** (Pass by Value): Параметр функции получает копию аргумента, и изменения параметра не влияют на оригинальный аргумент. Это подход, при котором аргумент передается "по копии".

**Аргументы по указателю** (Pass by Pointer): Параметр функции принимает указатель на оригинальный аргумент, что позволяет функции изменять значение аргумента через указатель.

**Аргументы по ссылке** (Pass by Reference): Параметр функции принимает ссылку на оригинальный аргумент, и изменения параметра внутри функции отражаются на оригинальном аргументе. Этот метод позволяет функции модифицировать аргумент напрямую.

**Аргументы по константной ссылке** (Const Reference): Параметр функции принимает константную ссылку на оригинальный аргумент, что позволяет функции читать значение аргумента, но не изменять его.

1. *Что такое аргументы по умолчанию?*

**Аргументы по умолчанию** (Default Arguments): Параметры функции могут иметь значения по умолчанию, которые используются, если аргументы не переданы при вызове функции.

*Объясните результат работы программы:*

В начале программы объявлены две функции:

showX(int x=0): Эта функция принимает один параметр x, который имеет значение по умолчанию 0. Если аргумент не передан при вызове функции, то x будет равно 0.

showXYZ(int x, int y = 1, int z = 2): Эта функция принимает три параметра x, y и z, каждый из которых имеет значение по умолчанию, если не переданы.

В функции main выполняются вызовы этих функций:

showX(3);: Вызывается showX с аргументом 3, поэтому x внутри функции будет равно 3. Выводится "x = 3".

showX();: Вызывается showX без аргументов, и так как x имеет значение по умолчанию 0, то выводится "x = 0".

showXYZ(4, 5, 6);: Вызывается showXYZ с аргументами 4, 5 и 6, поэтому x равно 4, y равно 5 и z равно 6. Выводится "x = 4 y = 5 z = 6".

showXYZ(7, 8);: Вызывается showXYZ с аргументами 7 и 8, и так как z имеет значение по умолчанию 2, то x равно 7, y равно 8 и z равно 2. Выводится "x = 7 y = 8 z = 2".

showXYZ(9);: Вызывается showXYZ с аргументом 9, и так как y и z имеют значения по умолчанию, то x равно 9, y равно 1 (значение по умолчанию) и z равно 2 (значение по умолчанию). Выводится "x = 9 y = 1 z = 2".

Таким образом, программа демонстрирует, как можно использовать параметры по умолчанию, чтобы делать функции более гибкими и упрощать вызовы функций с различным количеством аргументов.

1. *Что такое перегрузка функций?*

Перегрузка функций - это возможность создавать несколько функций с одним и тем же именем, но с разными параметрами. Когда функция вызывается, компилятор определяет, какую из перегруженных функций использовать, основываясь на типах и количестве переданных аргументов, что позволяет создавать более удобные и гибкие функции.

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

Массив - это упорядоченный набор однотипных элементов, расположенных в памяти последовательно. Все элементы массива имеют один и тот же тип данных, и обращение к ним происходит по индексу.

Структура (struct) - это пользовательский тип данных, который может содержать разные типы данных, называемые полями. В отличие от массива, структура позволяет группировать данные разных типов в единый объект, что делает ее удобной для представления сложных структур данных.