Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им. И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление: 710400 «Программная инженерия»

Дисциплина: «Объектно – ориентированное программирование»

**ОТЧЕТ**

По лабораторной работе №1.

Тема: «Повторение»

                                                                       Выполнила: студент группы

ПИ(б)-2-21 Марат кызы Жылдыз

Проверила: Мусабаев Э. Б.

**Бишкек – 2022**

**Лабораторная работа №1.**

**Тема: Повторение**

**Задания:**

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 ().

**Код программы:**

#include <iostream>

// Function prototype for power

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

int main() {

// Variables for input values

double base;

int exponent;

// Input the base number

std::cout << "Enter a number: ";

std::cin >> base;

// Check if the user wants to input the exponent

std::cout << "Do you want to enter the exponent? (1 - Yes, 0 - No): ";

int userChoice;

std::cin >> userChoice;

if (userChoice == 1) {

// Input the exponent

std::cout << "Enter the exponent (an integer greater than 2): ";

std::cin >> exponent;

// Call the power function with user-input values

double result = power(base, exponent);

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

}

else {

// Call the power function with the default exponent value

double result = power(base);

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

}

return 0;

}

// Definition of the power function

double power(double n, int p) {

double result = 1.0;

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

result \*= n;

}

return result;

}

**Результат программы:**

Рис.1 Если возвести, то возводит на заданную нам число в данном случае 2 на 3

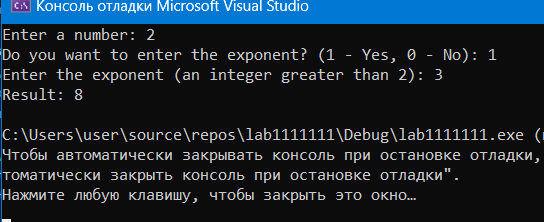
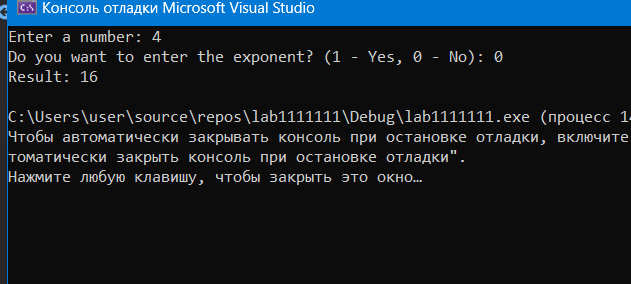


Рис.2 Если не возводить то, он умножает число



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 ().

**Код программы:**

#include <iostream>

// Перегруженные функции power

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

double power(char n, int p = 2);

double power(short int n, int p = 2);

double power(long int n, int p = 2);

double power(float n, int p = 2);

double power(int n, int p = 2); // Новая перегрузка для двух целочисленных аргументов

int main() {

// Примеры вызова функции power() с разными типами аргументов

std::cout << "power(2.5) = " << power(2.5) << std::endl;

std::cout << "power('A') = " << power('A') << std::endl;

std::cout << "power(3, 3) = " << power(3, 3) << std::endl;

std::cout << "power(12345L, 2) = " << power(12345L, 2) << std::endl;

std::cout << "power(4.7f, 3) = " << power(4.7f, 3) << std::endl;

return 0;

}

// Определение перегруженных функций power

double power(double n, int p) {

double result = 1.0;

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

result \*= n;

}

return result;

}

double power(char n, int p) {

// Простейшая реализация для char - приведение к double и вызов первой функции

return power(static\_cast<double>(n), p);

}

double power(short int n, int p) {

// Простейшая реализация для short int - приведение к double и вызов первой функции

return power(static\_cast<double>(n), p);

}

double power(long int n, int p) {

// Простейшая реализация для long int - приведение к double и вызов первой функции

return power(static\_cast<double>(n), p);

}

double power(float n, int p) {

// Простейшая реализация для float - приведение к double и вызов первой функции

return power(static\_cast<double>(n), p);

}

double power(int n, int p) {

// Реализация для двух целочисленных аргументов

double result = 1.0;

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

result \*= n;

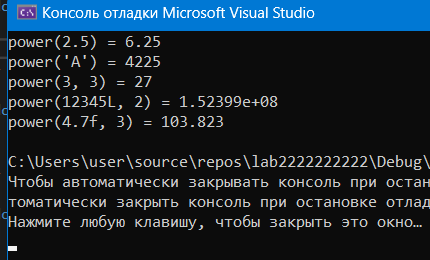
}

return result;

}

**Результат программы:**

Рис.1 Это основная функция программы, где вызываются различные варианты функции **power** с разными типами аргументов.



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.

**Код программы:**

#include <iostream>

#include <string>

// Structure representing information about an employee

struct Employee {

std::string surname;

int age;

std::string specialty;

double average\_salary;

};

int main() {

const int N = 2; // Number of factories

const int M = 3; // Maximum number of employees at each factory (adjust as needed)

// Create an array of structures for each factory

Employee factories[N][M];

// Input information about employees at each factory

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

std::cout << "Enter information for employees at Factory " << i + 1 << ":" << std::endl;

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

std::cout << "Employee " << j + 1 << ":" << std::endl;

std::cout << "Surname: ";

std::cin >> factories[i][j].surname;

std::cout << "Age: ";

std::cin >> factories[i][j].age;

std::cout << "Specialty (locksmith/turner): ";

std::cin >> factories[i][j].specialty;

std::cout << "Average salary: ";

std::cin >> factories[i][j].average\_salary;

}

}

// Count the number of locksmiths and turners

int locksmith\_count = 0;

int turner\_count = 0;

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

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

if (factories[i][j].specialty == "locksmith") {

++locksmith\_count;

}

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

++turner\_count;

}

}

}

// Output the results

std::cout << "Number of locksmiths: " << locksmith\_count << std::endl;

std::cout << "Number of turners: " << turner\_count << std::endl;

return 0;

}

**Результат программы:**

Рис.1 Здесь введены 3 сотрудника по 2 фабрикам, их имя, возраст, специальность по выбору (слесарь или токарь) и их средний оклад

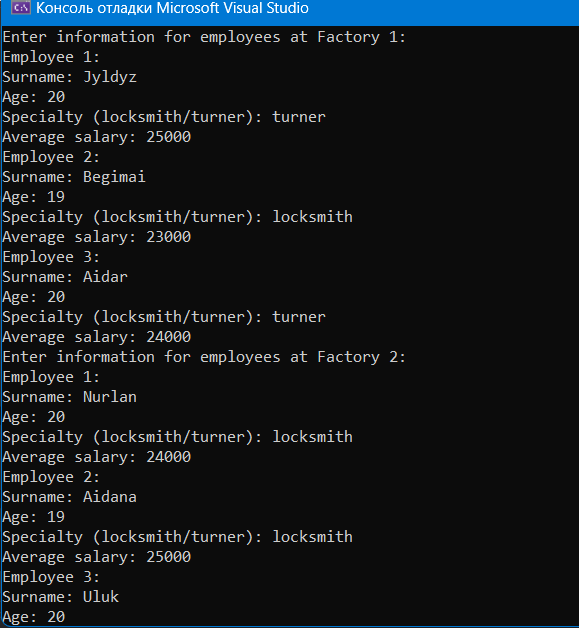


Рис.2 Здесь подсчитываем количество слесарей и токарей на основе введенных данных для каждого завода. В данном случае, у вас есть по три слесаря и токаря на каждом заводе.

