**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСТИТЕТ**

**им. И.Раззакова**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра «**Программное обеспечение компьютерных систем**»

Направление: 710400 «**Программная инженерия**»

Дисциплина: «**Объектно-ориентированное программирование**»

ОТЧЕТ

Лабораторная работа №1

Тема: “ **Типы данных, функции и структуры в программировании** ”

Выполнила: Бактыбекова Айтунук

Группа: ПИ-2-21

Проверил: Мусабаев Э.Б.

Бишкек – 2024

Задание №1

Постановка задачи:

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 ().

Описание исходных данных:

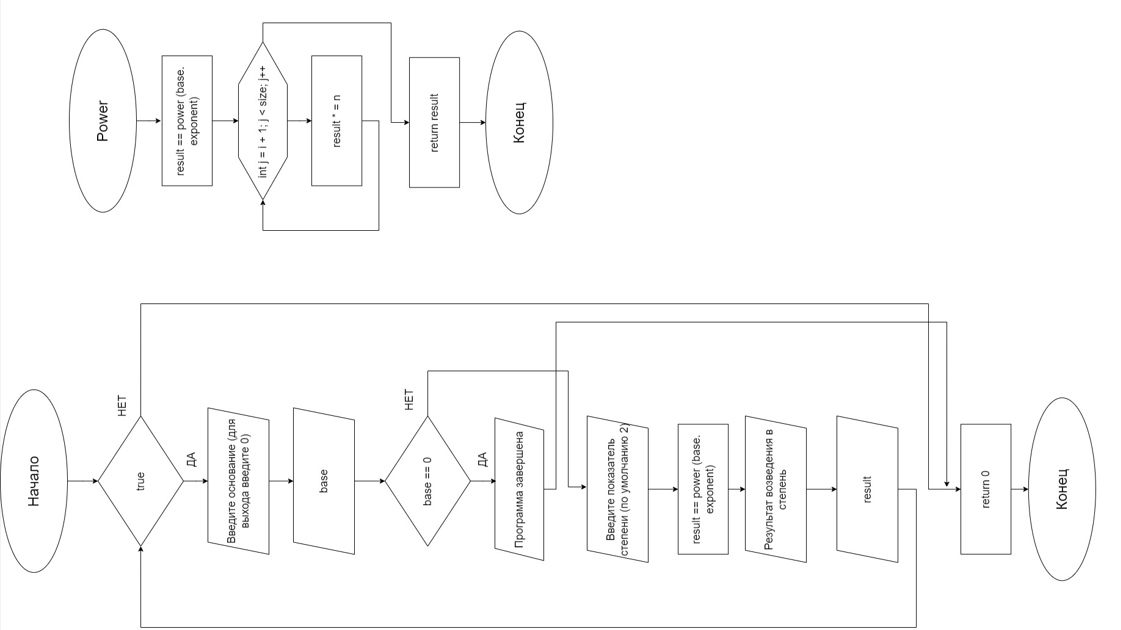
Функция power()

* **Параметры:**
  + **double n**: значение, которое нужно возвести в степень.
  + **int p** (со значением по умолчанию 2): показатель степени.
* **Возвращаемое значение:**
  + **double**: результат возведения числа **n** в степень **p**.

**Исходные данные:**

* **double base**: число, которое нужно возвести в степень.
* **int exponent**: показатель степени (значение по умолчанию: 2).

Блок-схема:



**Код программы:**

#include <iostream>

#include <sstream>

using namespace std;

// Прототип функции power

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

int main() {

setlocale(LC\_ALL, "Russian");

while (true) {

double base;

int exponent;

// Ввод числа

cout << "Введите основание (для выхода введите 0): ";

cin >> base;

if (base == 0) {

cout << "Программа завершена." << endl;

break; // Выход из цикла

}

// Ввод строки для показателя степени

string exponentInput;

cout << "Введите показатель степени (по умолчанию 2): ";

cin.ignore(); // Игнорируем оставшийся символ новой строки в буфере

getline(cin, exponentInput);

// Проверка, что строка не пуста

if (!exponentInput.empty()) {

// Преобразование строки в число

stringstream ss(exponentInput);

ss >> exponent;

// Проверка на корректность ввода

if (ss.fail()) {

cerr << "Ошибка: Некорректный ввод показателя степени." << endl;

return 1;

}

}

else {

// Если строка пуста, использовать значение по умолчанию

exponent = 2;

}

// Вызов функции power с введенными значениями

double result = power(base, exponent);

// Вывод результата

cout << "Результат возведения в степень: " << result << endl << endl;

}

return 0;

}

// Определение функции power

double power(double n, int p) {

double result = 1.0; // Инициализируем результат единицей

// Цикл для умножения числа n само на себя p раз

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

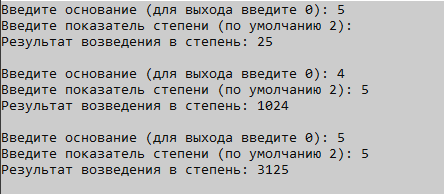
result \*= n;

}

return result;

}

Результат:



Задание №2

Постановка задачи:

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 ().

Описание исходных данных:

1. **Функции power() (перегруженные):**
   * **Аргументы:**
     + **double n**: Основа, которую нужно возвести в степень.
     + **int p** (со значением по умолчанию 2): Показатель степени.
   * **Возвращаемое значение:**
     + **double**: Результат возведения числа **n** в степень **p**.
2. **Простая функция power\_int():**
   * **Аргументы:**
     + **double n**: Основа, которую нужно возвести в степень.
     + **int p**: Показатель степени.
   * **Возвращаемое значение:**
     + **double**: Результат возведения числа **n** в степень **p**.

**Исходные данные:**

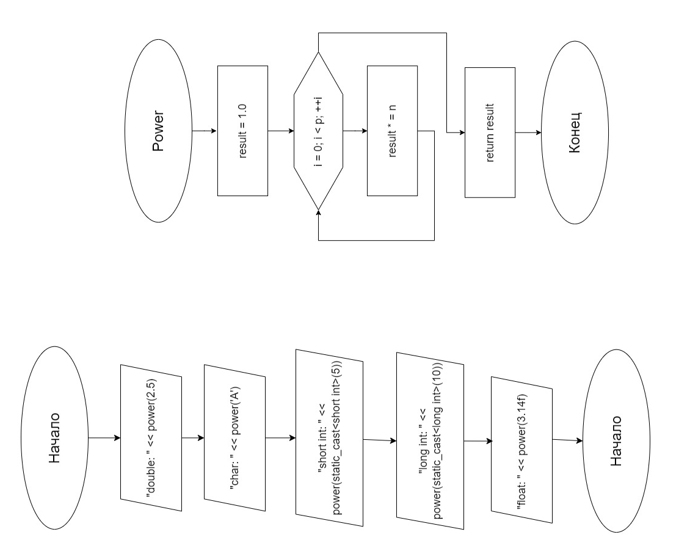
* **double n1 = 6.25: Пример для типа double.**
* **char n2 = '4225': Пример для типа char.**
* **short int n3 = 5: Пример для типа short int.**
* **long int n4 = 100: Пример для типа long int.**
* **float n5 = 9.8596: Пример для типа float.**

Описание перегруженных функций power():

Для каждого типа аргумента производится вызов основной функции power(double n, int p), преобразуя аргумент к типу double.

Такая перегрузка позволяет использовать функцию power() с различными типами данных.

Блок-схема:



Код программы:

#include <iostream>

using namespace std;

// Перегруженные функции power() для разных типов данных

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

double result = 1.0;

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

result \*= n;

}

return result;

}

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

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

}

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

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

}

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

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

}

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

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

}

// Простая функция возведения в степень для целых чисел

double power\_int(double n, int p) {

double result = 1.0;

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

result \*= n;

}

return result;

}

int main() {

// Вызов функции power() со всеми возможными типами аргументов

cout << "double: " << power(2.5) << endl;

cout << "char: " << power('A') << endl;

cout << "short int: " << power(static\_cast<short int>(5)) << endl;

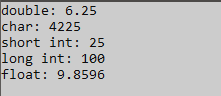
cout << "long int: " << power(static\_cast<long int>(10)) << endl;

cout << "float: " << power(3.14f) << endl;

return 0;

}

Результат:



Задание №3

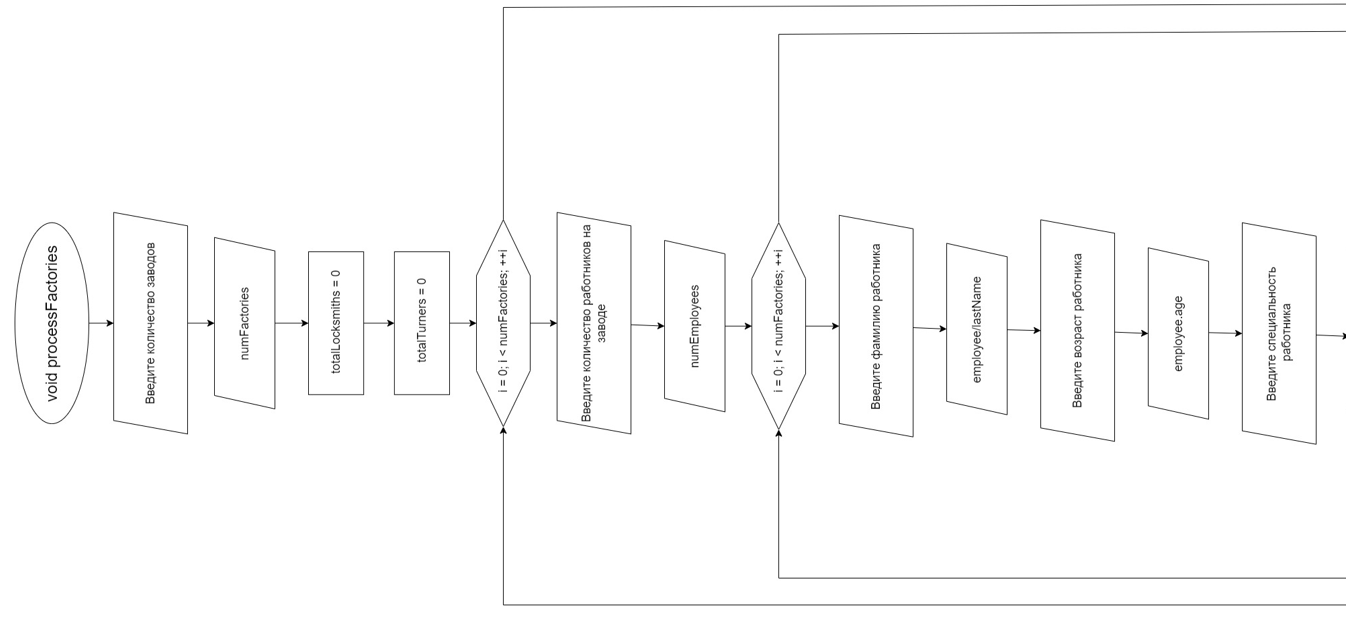
Постановка задачи:

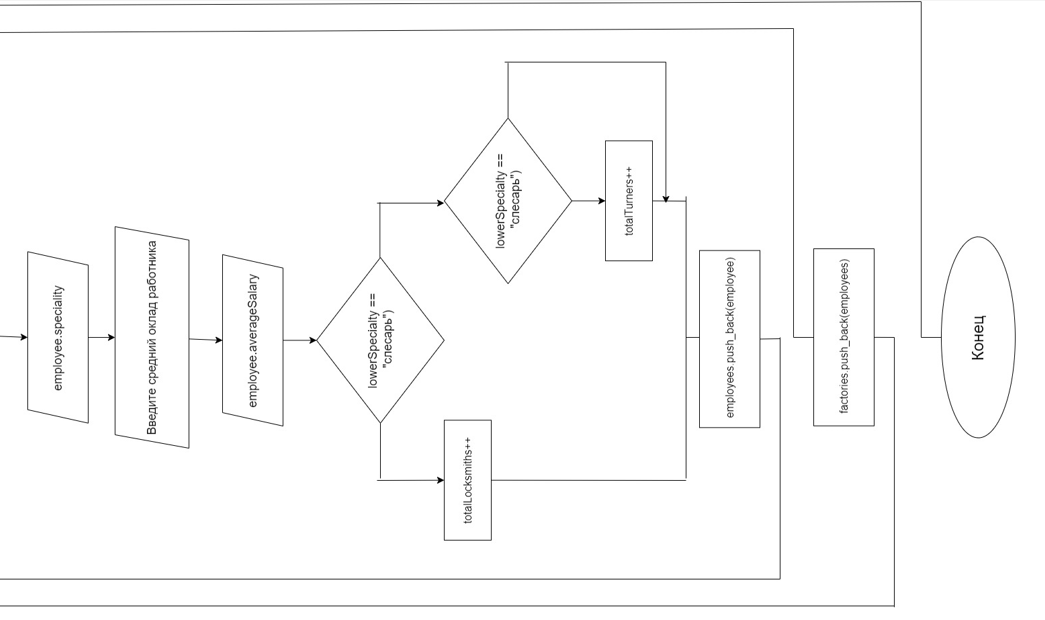
Информация по работникам N заводов задается структурой с полями: фамилия, возраст, специальность, средний оклад. Ввести информацию по заводам, посчитать количество слесарей и токарей. Вывести эти значения на консоль.

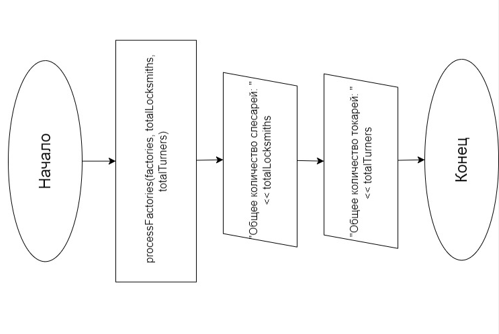
Описание входных данных:

1. **Структура Employee:**
   * **string lastName**: Фамилия работника.
   * **int age**: Возраст работника.
   * **string specialty**: Специальность работника.
   * **double averageSalary**: Средний оклад работника.
2. **Функция processFactories():**
   * **Параметры:**
     + **vector<vector<Employee>>& factories**: Ссылка на вектор векторов для хранения информации о работниках на каждом заводе.
     + **int& totalLocksmiths**: Ссылка на переменную для общего количества слесарей.
     + **int& totalTurners**: Ссылка на переменную для общего количества токарей.
   * **Описание:**
     + Вводит информацию по каждому заводу, считает количество слесарей и токарей, сохраняет данные в вектор **factories**.
   * **Исходные данные:**
     + **int numFactories**: Количество заводов, вводится с клавиатуры.
     + **int numEmployees**: Количество работников на каждом заводе, вводится с клавиатуры.
     + **vector<vector<Employee>> factories**: Вектор векторов для хранения информации о работниках на каждом заводе.
     + **int totalLocksmiths**: Переменная для общего количества слесарей.
     + **int totalTurners**: Переменная для общего количества токарей.
     + **string lowerSpecialty**: Строка, полученная после приведения специальности к нижнему регистру.
3. **Функция main():**
   * **Описание:**
     + Включает в себя ввод информации по заводам, вызов функции **processFactories()**, вывод результатов на консоль.
   * **Исходные данные:**
     + **vector<vector<Employee>> factories**: Вектор векторов для хранения информации о работниках на каждом заводе.
     + **int totalLocksmiths**: Переменная для общего количества слесарей.
     + **int totalTurners**: Переменная для общего количества токарей.

Блок-схема:







Код программы:

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

// Структура для хранения информации о работнике

struct Employee {

string lastName;

int age;

string specialty;

double averageSalary;

};

// Привести строку к нижнему регистру

string toLowerCase(string str) {

transform(str.begin(), str.end(), str.begin(), ::tolower);

return str;

}

// Функция для ввода информации по заводам и подсчета слесарей и токарей

void processFactories(vector<vector<Employee>>& factories, int& totalLocksmiths, int& totalTurners) {

// Количество заводов

int numFactories;

cout << "Введите количество заводов: ";

cin >> numFactories;

// Инициализация переменных с общим количеством слесарей и токарей

totalLocksmiths = 0;

totalTurners = 0;

// Ввод информации по каждому заводу

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

int numEmployees;

cout << "Введите количество работников на заводе " << i + 1 << ": ";

cin >> numEmployees;

// Вектор для хранения информации о работниках на текущем заводе

vector<Employee> employees;

// Ввод информации о каждом работнике

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

Employee employee;

cout << "Введите фамилию работника " << j + 1 << ": ";

cin >> employee.lastName;

cout << "Введите возраст работника " << j + 1 << ": ";

cin >> employee.age;

cout << "Введите специальность работника " << j + 1 << ": ";

cin >> employee.specialty;

cout << "Введите средний оклад работника " << j + 1 << ": ";

cin >> employee.averageSalary;

// Привести специальность к нижнему регистру перед сравнением

string lowerSpecialty = toLowerCase(employee.specialty);

// Подсчет слесарей и токарей

if (lowerSpecialty == "слесарь") {

totalLocksmiths++;

}

else if (lowerSpecialty == "токарь") {

totalTurners++;

}

// Добавление работника в вектор

employees.push\_back(employee);

}

// Добавление вектора с информацией о работниках на текущем заводе

factories.push\_back(employees);

}

}

int main() {

setlocale(LC\_ALL, "Russian");

// Вектор векторов для хранения информации о работниках на каждом заводе

vector<vector<Employee>> factories;

// Общее количество слесарей и токарей

int totalLocksmiths, totalTurners;

// Вызов функции для ввода информации по заводам и подсчета слесарей и токарей

processFactories(factories, totalLocksmiths, totalTurners);

// Вывод результатов

cout << "Общее количество слесарей: " << totalLocksmiths << endl;

cout << "Общее количество токарей: " << totalTurners << endl;

return 0;

}

Результат:

