**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №1**

Выполнил: студент группы ПИ-3-21

Копжашаров Азамат

Проверил: Мусабаев Э.Б.

**Бишкек 2021**

# **Задание №1**

**Постановка задачи:**

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 ().

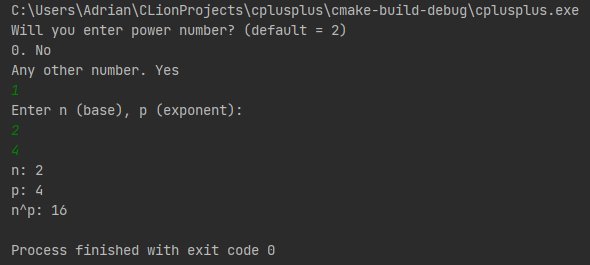
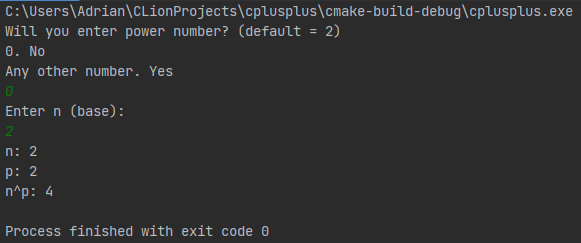
**Исходные данные:**

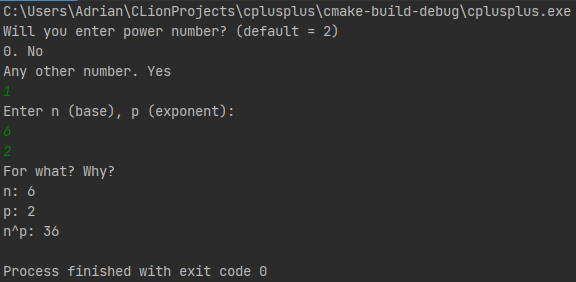
В качестве исходных данных нам даны n, p.

**Исходный код программы на C++:**

// 1.1  
#include <iostream>  
using namespace std;  
  
double power(double n, int p = 2);  
  
int main() {  
 int check;  
 int n, p, result;  
 cout << "Will you enter power number? (default = 2)" << endl  
 << "0. No \nAny other number. Yes" << endl;  
 cin >> check;  
 if(check){  
 cout << "Enter n (base), p (exponent): " << endl;  
 cin >> n >> p;  
 result = power(n, p);  
 if(p == 2) cout << "For what? Why?" << endl;  
 }else{  
 cout << "Enter n (base): " << endl;  
 cin >> n;  
 p = 2;  
 result = power(n);  
 }  
 cout << "n: " << n << endl  
 << "p: " << p << endl  
 << "n^p: " << result << endl;  
 return 0;  
}  
  
double power(double n, int p) {  
 double ans = n;  
 while(--p) ans \*= n;  
 return ans;  
}

**Тесты:**





# **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 ().

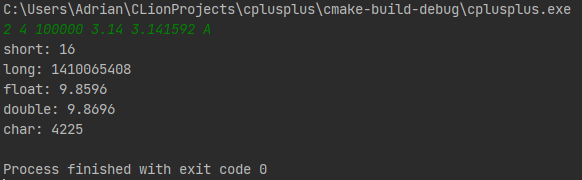
**Исходные данные:**

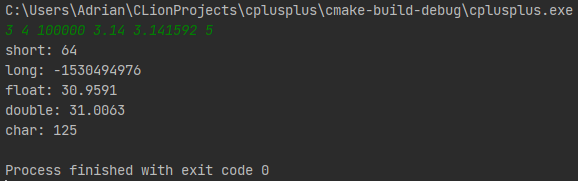
В качестве исходных данных нам ничего не дается, но исходя из условий задачи, требуется использовать аналогичные исходные данные, что и в задаче №1: p, n, только уже для каждого перечисленного типа данных.

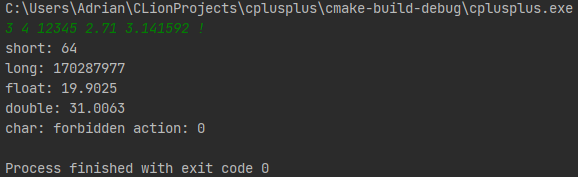
**Исходный код программы на C++:**

// 1.2  
#include <iostream>  
using namespace std;  
  
double power(double n, int p = 2) {  
 double ans = n;  
 while (--p) ans \*= n;  
 return ans;  
}  
  
short power(short n, int p = 2) {  
 short ans = n;  
 while (--p) ans \*= n;  
 return ans;  
}  
  
long power(long n, int p = 2) {  
 long ans = n;  
 while (--p) ans \*= n;  
 return ans;  
}  
  
float power(float n, int p = 2) {  
 float ans = n;  
 while (--p) ans \*= n;  
 return ans;  
}  
  
int power(char n, int p = 2) {  
 if((int)n >= 48 && (int)n <= 57){  
 int ans = (int)(n-48), exp = (int)(n-48);  
 while(--p) ans \*= exp;  
 return ans;  
 }else if(((int)n >= 65 && (int)n <= 90) || ((int)n >= 97 && (int)n <= 122)){  
 int ans = (int)n, exp = (int)n;  
 while(--p) ans \*= exp;  
 return ans;  
 }else{  
 cout << "forbidden action: ";  
 return 0;  
 }  
}  
  
int main() {  
 int exp;  
 short short1;  
 long long1;  
 float float1;  
 double double1;  
 char char1;  
  
 cin >> exp >> short1 >> long1 >> float1 >> double1 >> char1 ;  
 cout << "short: " << power(short1 ,exp) << endl;  
 cout << "long: " << power(long1 ,exp) << endl;  
 cout << "float: " << power(float1,exp) << endl;  
 cout << "double: " << power(double1 ,exp) << endl;  
 cout << "char: " << power(char1 ,exp) << endl;  
  
 return 0;  
}  
  
/\*  
 2 4 100000 3.14 3.141592 A  
 2 4 100000 3.14 3.141592 z  
 3 4 100000 3.14 3.141592 5  
 3 4 100000 3.14 3.141592 z  
 3 4 12345 2.71 3.141592 !  
 \*/

**Тесты:**







# **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.

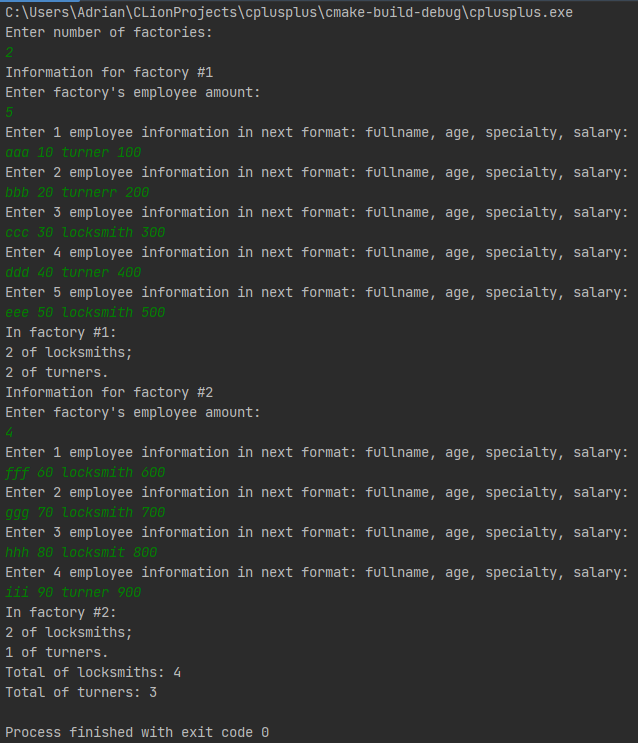
**Исходные данные:**

N – количество заводов.

**Исходный код программы на C++:**

// 1.3  
#include <iostream>  
#include <vector>  
using namespace std;  
  
struct Employee{  
 string fullname;  
 int age;  
 string specialty;  
 float salary;  
};  
  
int main() {  
 int n;  
 cout << "Enter number of factories: " << endl;  
 cin >> n;  
 vector<vector<Employee>> factories;  
 int turnersAmount = 0, locksmithsAmount = 0;  
 for(int i = 0; i < n; i++) {  
  
 cout << "Information for factory #" << i+1 << endl  
 << "Enter factory's employee amount: " << endl;  
 int m;  
 cin >> m;  
 vector<Employee> employees;  
 for(int j = 0; j < m; j++){  
 Employee emp;  
 cout << "Enter " << j+1 << " employee information in next format: fullname, age, specialty, salary: " << endl;  
 cin >> emp.fullname >> emp.age >> emp.specialty >> emp.salary;  
 employees.push\_back(emp);  
 }  
  
 int lCount = 0, tCount = 0;  
 for(auto emp : employees){  
 if(emp.specialty == "turner" || emp.specialty == "Turner") tCount++;  
 else if(emp.specialty == "locksmith" || emp.specialty == "Locksmith") lCount++;  
 }  
 cout << "In factory #" << i+1 << ":" << endl << lCount << " of locksmiths;" << endl << tCount << " of turners." << endl;  
 turnersAmount += tCount;  
 locksmithsAmount += lCount;  
 factories.push\_back(employees);  
 }  
 cout << "Total of locksmiths: " << locksmithsAmount << endl << "Total of turners: " << turnersAmount << endl;  
  
 return 0;  
}  
  
/\*  
2  
5  
aaa 10 turner 100  
bbb 20 turnerr 200  
ccc 30 locksmith 300  
ddd 40 turner 400  
eee 50 locksmith 500  
4  
fff 60 locksmith 600  
ggg 70 locksmith 700  
hhh 80 locksmit 800  
iii 90 turner 900  
 \*/

**Тесты:**



# **Ответы на вопросы:**

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

* Bool – тип данных для логических значений, резервируется 1 байт памяти;
* Char – тип данных для символьных значений, резервирует 1 байт памяти;
* Int – тип данных для целочисленных значений, резервируется 4 байта памяти;
* Short – тип данных для целочисленных значений, резервируется 2 байта памяти;
* Long – тип данных для целочисленных значений, резервируется 4 байта памяти;
* Float – тип данных для числовых значений с плавающей точкой, занимает 4 байта памяти;
* Double – тип данных для числовых значений с плавающей точкой двойной точности, резервируется 8 байт памяти;

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

а) 12;  
б) ‘a’;  
в) 4.28915;  
г) JungleJim;  
д) JungleJim().

а) 12 – числовая константа, значение типа int;  
б) ‘a’ – символьная константа, значение типа char;  
в) 4.28915 – числовая константа с плавающей запятой, значение типа float;  
г) JungleJim – строчный литерал, может быть как переменной типа string, так и названием класса, функции, объекта и т.п.;  
д) JungleJim() – объявление функции/конструктора JungleJim, может, даже обычный вызов функции.

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

Функция – это фрагмент кода, который выполняет определенные задачи внутри своего определения. Функция представляет собой отдельно выделенный алгоритм, который пошагово преобразует исходные данные в выходные, и может быть многократно использован.

**4. Напишите синтаксис функции.**

#include <iostream>  
using namespace std;  
 // Объявление функции (не обязательно, но желательно)  
void printHello(); // Считается хорошим тоном  
  
int main() {  
 printHello(); // Вызов функции  
 return 0;  
}  
  
// Определение функции  
void printHello(){ // Тип, название и аргументы функции  
 cout << "Hello, World!" << endl; // Тело функции  
} // Конец функции

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

Предварительное объявление функции дает компилятору понять, что где-то в проекте есть функция с идентичным названием и входными параметрами.

void printHello();  
  
int main() {  
 printHello();  
 return 0;  
}  
  
void printHello() {  
 cout << "Hello, World!" << endl;  
}

Без объявления функции компилятор не сможет понять, что существует функция, пока не дойдет до нее, поэтому вызов функции, которая определена после функции main приведет к ошибке.

int main() {  
 printHello();  
 return 0;  
}  
  
void printHello() {  
 cout << "Hello, World!" << endl;  
}

Для корректной работы без объявления функции требуется определить функцию до ее вызова.

void printHello() {  
 cout << "Hello, World!" << endl;  
}  
  
int main() {  
 printHello();  
 return 0;  
}

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

Объясните результаты работы программ:

1.

#include <iostream>  
using namespace std;  
int incr(int m){  
 m = m + 1;  
 return m;  
}  
int main() {  
 int n = 5;  
 cout << "n = " << incr(n) << endl;  
 cout << "n = " << n << endl;  
 return 0;  
}

В начале объявляется функция incr с входным параметром m типа int, она возвращает значение m+1. Т.к. функция принимает входные данные по значению, то это не отразится на реальной переменной, в этом случае, m не изменится на +1.

Программа начинается с инициализации переменной n типа int равной 5. Далее идет операция вывода с вызовом функции incr(n), которая выведет число 6. Далее идет вывод числа n без использования функции, поэтому будет выведено число 5.

2.

#include <iostream>  
using namespace std;  
int incr(int &m){  
 m = m + 1;  
 return m;  
}  
int main() {  
 int n = 5;  
 cout << "n = " << incr(n) << endl;  
 cout << "n = " << n << endl;  
 return 0;  
}

В начале объявляется функция incr с входным параметром m типа int, она возвращает значение m+1. Т.к. функция принимает аргументы по ссылке, то это напрямую перезапишет значение переменной n из функции main.

Программа начинается с инициализации переменной n типа int равной 5. Далее идет операция вывода с вызовом функции incr(n), которая выведет число 6. Далее идет вывод числа n без использования функции, т.к. в прошлом вызове функции incr(n) значение n изменилось на 6, будет выведено 6.

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

Передача по значению. В функцию передается копия объекта, с которой она работает. После отработки функции значение оригинальной переменной остается неизменной.

void inc(int x) {  
 x = x + 1;  
}  
int main() {  
 int x = 5;  
 inc(x);  
 cout << x;  
 return 0;  
}

Передача по указателю. Вызов функции вместо копии переменной создаст переменную ссылочного типа x, в нее будет скопирован адрес ячейки x операцией взятия адреса. Далее, в функции, с помощью операции косвенной адресации в ячейку x запишется новый результат, который поменяет переменную, использующуюся в main.

void inc(int \*x) {  
 \*x = \*x + 1;  
}  
int main() {  
 int x = 5;  
 inc(&x);  
 cout << x;  
 return 0;  
}

Передача по ссылке. В случае вызова по ссылке оператор вызова дает вызываемой функции возможность прямого доступа к передаваемым данным, а также возможность изменения этих данных. Вызов по ссылке хорош в смысле производительности, потому что он исключает накладные расходы на копирование больших объемов данных.

void inc(int &x) {  
 x = x + 1;  
}  
int main() {  
 int x = 5;  
 inc(x);  
 cout << x;  
 return 0;  
}

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

Аргументы по умолчанию – это способ способность работы функции в случае, если в функцию не были переданы аргументы при вызове функции.

void printX(int x = 10) {  
 cout << x << endl;  
}  
int main() {  
 int x = 5;  
 printX(x);  
 printX();  
 return 0;  
}

Объясните результат работы программы:

#include <iostream>  
using namespace std;  
// Аргумент функции имеет значение по умолчанию  
void showX(int x = 0){  
 cout << "x = " << x << endl;  
}  
void showXYZ(int x, int y = 1, int z = 2);  
// Два аргумента функции в прототипе имеют значения  
// по умолчанию  
// сама функция описана в конце программы  
int main() {  
 showX(3);  
 showX();  
 showXYZ(4, 5, 6);  
 showXYZ(7, 8);  
 showXYZ(9);  
 return 0;  
}  
// При описании функции значения по умолчанию не указываются  
void showXYZ(int x, int y, int z){  
 cout << "x = " << x << " ";  
 cout << "y = " << y << " ";  
 cout << "z = " << z << endl;  
}

В начале инициализируется функция showX, которая принимает переменную x типа int и выводит ее. Она имеет значение 0 по умолчанию.

После идет определение функции showXYZ с 3 входными переменными: x типа int, y типа int со значением по умолчанию 1, z типа int со значением по умолчанию 2.

В конце после функции main определяется функция showXYZ, в которой идет вывод переменных x, y, z.

В функции main идет последовательный вызов функций:  
вызов функции showX(3), которая выведет 3;  
вызов функции showX() без входного аргумента, которая выведет 0 в консоль;  
вызов функции showXYZ(4, 5, 6), которая выведет “x = 4, y = 5, y = 6”$  
вызов функции showXYZ(7, 8), которая выведет “x = 7, y = 8, z = 2”, т.к. для z не передавалось значение и оно было установлено по умолчанию;  
вызов функции showXYZ(9), которая выведет «x = 9, y = 1, z = 2», т.к. для y, z не передавались значения и они были установлены по умолчанию.

**9. Что такое перегрузка функций?**

Перегрузка функции – возможность создания и использования функций с одним названием, но разными параметрами. При вызове функции компилятор автоматически выбирает правильную для использования функцию на основе количества входных параметров и их типах.

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

Массив (Array) – это упорядоченный набор последовательно расположенных однотипных элементов. В массиве все элементы имеют один и тот же тип данных и обращение к ним реализуется при помощи индексов.

Структура (Struct) – пользовательский тип данных, задаваемый программистом на свое усмотрение. В отличие от массива, структура может содержать различные типы данных, которые называются полями.