Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский государственный национальный**

**исследовательский университет»**

**Индивидуальная работа по дисциплине «Языки программирования»**

Выполнил студент группы ФИТ 1,2 – 2023

Лисуков Дмитрий Владимирович

Преподаватель: Иммис М.В.

Пермь 2023

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ «ДЛИННАЯ АРИФМЕТИКА»**

1. **Постановка задачи**

Составить программу вычисления числа a в степени n, если a > MaxInt, n > 10.

1. **Описание программы**

Данная программа подразумевает умножение длинных чисел, с помощью применения подхода длинной арифметики. С помощью данного подхода можно выполнять арифметические операции с числами, которые превышают максимальное значение в стандартных типах данных, таких как int или long.

В данной программе необходимо возводить число a в степень числа равному n. Подразумевается, что ‘a’ больше максимального числа, которое может хранить тип данных int (максимальное возможное число, которое может хранить int = 2147483647). Число n должно быть строго больше 10.

1. **Описание алгоритма решения**

Для начала необходимо ввести число ‘a’, которое будет храниться в строковом виде и произвести проверку на корректность, для этого необходимо, чтобы символ с индексом 0 не содержал ‘-‘, а при переводе в тип int, число не было меньшим 2147483647.

Далее создаем вектор, в котором будем хранить каждый символ нашей строки отдельно. В данный вектор необходимо преобразовать каждое число, путем вычитания из него символа '0'. При таком варианте, наш вектор будет хранить отдельно каждую цифру числа ‘a’ на разных разрядах отдельно.

Задаем еще один вектор, который будет равен предыдущему.

В программе прописываем цикл, который сработает n раз, в цикле будем вызывать функцию, которая умножает все числа вектора на сами себя и записывать результат в результирующем векторе.

Сама функция будет производить умножение числа на число, и сохранять результат в конечном векторе. При необходимости переноса разряда числа, данная функция переносит разряд на 1 вправо и оставляет одно число в текущем разряде. В конце удаляет все незначащие нули, если такие имеются.

Такие структуры данных были выбраны неслучайно, дело в том, что нам необходим подход, с помощью которого можно разбить число поразрядно и работать с каждым разрядом отдельно, и грамотно обращаться к элементам с помощью индекса числа, при этом сохраняя все значащие цифры числа.

1. **Тесты программы**

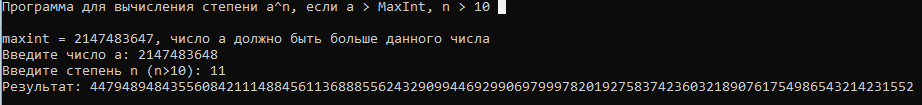
В качестве проверки результата работы программы, результат будем сверять с помощью самого популярного и точного калькулятора «Wolframalpha».

**Тест № 1**. Возьмем числа, которые являются минимальными в качестве возможных.

a = 2147483648

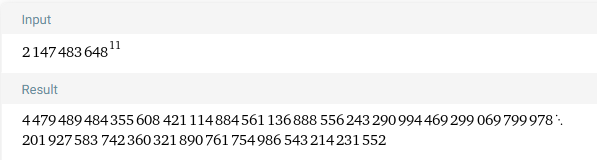
n = 11

Результат выполнения программы:



4479489484355608421114884561136888556243290994469299069799978201927583742360321890761754986543214231552

Результат выполнения в калькуляторе «wolframalpha»:



Сверяем результат с помощью простого скрипта на python.



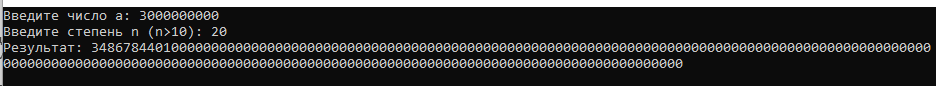
Результат выполнения выдал верный результат

**Тест № 2**. Возьмем числа, которые можно легко доказать

a = 3000000000

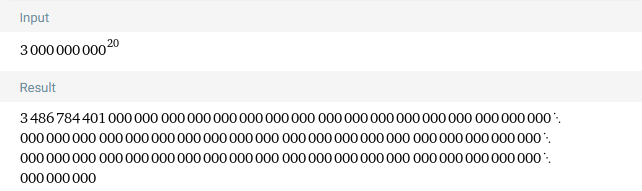
n = 20

Результат выполнения программы:

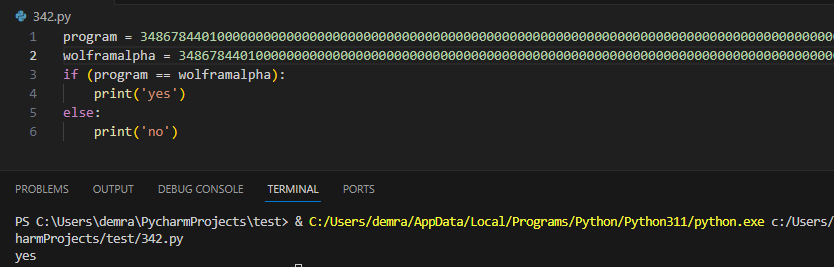


3486784401000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

Результат работы в калькуляторе «wolframalpha»:



Сверяем результат с помощью простого скрипта на python:



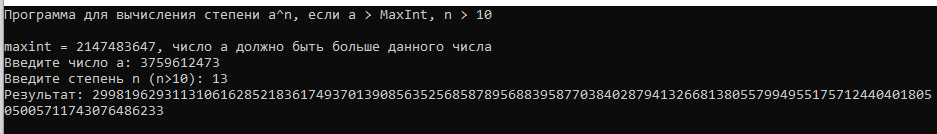
Результат совпал

**Тест № 3** Входные данные:

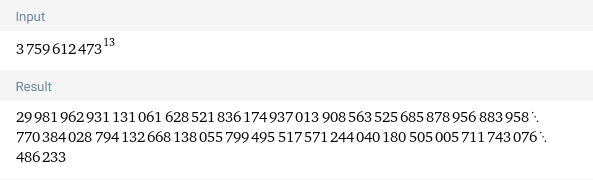
а = 3759612473

n = 13

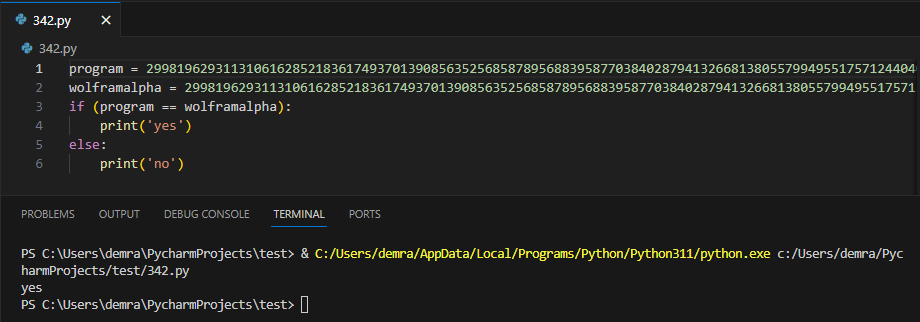
Результат работы программы:



29981962931131061628521836174937013908563525685878956883958770384028794132668138055799495517571244040180505005711743076486233



Сверяем результат с помощью простого скрипта на python.



Листинг программы:

#include <iostream>

#include <string>

#include <vector>

#include <windows.h>

#include <cstdlib>

using namespace std;

vector<int> multiply(vector<int>& a, vector<int>& b) {

int m = a.size();

int n = b.size();

vector<int> result(m + n, 0);

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

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

result[i + j] += a[i] \* b[j]; // Умножение цифр чисел и сохранение в промежуточном результате

result[i + j + 1] += result[i + j] / 10; // Перенос разряда при необходимости

result[i + j] %= 10; // Оставляем только одну цифру в текущем разряде

}

}

while (result.size() > 1 && result.back() == 0) {

result.pop\_back(); // Удаляем незначащие нули в конце числа

}

return result;

}

int main() {

setlocale(LC\_ALL, "RU");

string a\_str;

int n;

int num = 0;

cout << "Программа для вычисления степени a^n, если a > MaxInt, n > 10\n" << endl;

cout << "maxint = 2147483647, число a должно быть больше данного числа" << endl;

cout << "Введите число a: ";

cin >> a\_str;

// проверка на коректность данных

num = atoi(a\_str.c\_str());

if (a\_str[0] == '-') {

cout << "НЕВЕРНО, число не должно быть отрицательным" << endl;

return 0;

}

if (num < 2147483647) {

cout << "НЕВЕРНО, маленькое число, необходимо количество разрядов > 10 или большее число" << endl;

return 0;

}

cout << "Введите степень n (n>10): "; // проверка на коректность введенных данных для переменной n

cin >> n;

if (n <= 10) {

cout << "Степень должна быть больше 10" << endl;

return 0;

}

vector<int> a;

for (int i = a\_str.length() - 1; i >= 0; i--) {

a.push\_back(a\_str[i] - '0'); // Преобразование строки в вектор цифр

}

vector<int> result = a;

for (int i = 1; i < n; i++) {

result = multiply(result, a); // Последовательное умножение числа на само себя n раз

}

cout << "Результат: ";

for (int i = result.size() - 1; i >= 0; i--) {

cout << result[i]; // Вывод результата

}

cout << endl;

return 0;

}

Также исходный код можно найти на github.

URL: https://github.com/emfermenty/university.cpp/blob/main/long%20arithmetic/individylka.cpp