



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ,
информационные технологии»

ДОМАШНЯЯ РАБОТА №1

«Моделирование операций над длинными числами»


ДИСЦИПЛИНА: «Типы и структуры данных»

Выполнил: студент гр. ИУК4-31Б


(подпись)

(Суриков Н. С.)
(Ф.И.О.)

Проверил:


(подпись)

(Былинка М. И.)
(Ф.И.О.)

Дата сдачи (защиты): 05.11.24

Результаты сдачи (защиты):

- Балльная оценка: 145

- Оценка:

Цель: формирование практических навыков моделирования операций над длинными числами.

Задачи:

1. Познакомиться с представлением чисел в памяти компьютера.
2. Создать собственную модель для представления длинного числа в памяти компьютера.
3. Научиться составлять и реализовывать алгоритмы для арифметических операций над длинными числами.
4. Смоделировать математическую операцию с длинными числами согласно варианту.

Вариант 25

Формулировка задания:

Смоделировать операцию вычисления квадрата действительного числа в форме $(zn)0.m \text{ E } N$. Результат выдать в форме $(zn)0.m1 \text{ E } N1$.

Листинг программы:

```
1  #include <cmath>
2  #include <iomanip>
3  #include <iostream>
4  #include <stdexcept>
5  #include <string>
6  #include <vector>
7
8  #define MAX_DIGITS 30
9  #define BASE 10
10
11 struct LongNumber
12 {
13     bool is_negative;           // Знак числа
14     std::vector<int> digits;    // Цифры числа
15     int exponent;              // Экспонента
16
17     LongNumber() : is_negative(false), exponent(0) {}
18
19     void readNumber(const std::string &input)
20     {
21         size_t e_pos = input.find('E');
22         if (e_pos == std::string::npos)
23         {
```

```

24         throw std::invalid_argument("Invalid format: missing 'E'");
25     }
26
27     std::string mantissa = input.substr(0, e_pos);
28     std::string exponent_str = input.substr(e_pos + 1);
29
30     // Определение знака экспоненты
31     bool exp_negative = false;
32     if (exponent_str[0] == '-')
33     {
34         exp_negative = true;
35         exponent_str = exponent_str.substr(1);
36     }
37     else if (exponent_str[0] == '+')
38     {
39         exponent_str = exponent_str.substr(1);
40     }
41
42     exponent = std::stoi(exponent_str);
43     if (exp_negative)
44     {
45         exponent = -exponent;
46     }
47
48     is_negative = (mantissa[0] == '-');
49     if (is_negative || mantissa[0] == '+')
50         mantissa = mantissa.substr(1);
51
52     size_t dot_pos = mantissa.find('.');
53     if (dot_pos != std::string::npos)
54     {
55         exponent -= (mantissa.size() - dot_pos - 1);
56         mantissa.erase(dot_pos, 1);
57     }
58
59     digits.clear();
60     for (char c : mantissa)
61     {
62         if (isdigit(c))
63         {
64             digits.push_back(c - '0');
65         }
66         else
67         {
68             throw std::invalid_argument("Invalid character in number");
69         }
70     }
71
72     if (digits.size() > MAX_DIGITS)
73     {
74         digits.resize(MAX_DIGITS);
75     }
76 }
77
78 LongNumber square() const

```

```

79     {
80         LongNumber result;
81         result.is_negative = false;
82
83         std::vector<int> temp_digits(2 * digits.size(), 0);
84
85         for (size_t i = 0; i < digits.size(); ++i)
86         {
87             for (size_t j = 0; j < digits.size(); ++j)
88             {
89                 temp_digits[i + j] += digits[i] * digits[j];
90                 if (temp_digits[i + j] >= BASE)
91                 {
92                     temp_digits[i + j + 1] += temp_digits[i + j] / BASE;
93                     temp_digits[i + j] %= BASE;
94                 }
95             }
96         }
97
98         // Удаляем лишние нули в конце
99         while (temp_digits.size() > 1 && temp_digits.back() == 0)
100         {
101             temp_digits.pop_back();
102         }
103
104         // Сохраняем результат
105         result.digits.assign(temp_digits.begin(), temp_digits.begin() +
std::min(MAX_DIGITS, int(temp_digits.size())));
106         result.exponent = 2 * exponent;
107
108         return result;
109     }
110
111     void printNumber() const
112     {
113         if (is_negative)
114             std::cout << "-";
115         std::cout << "0.";
116         for (size_t i = 0; i < digits.size(); ++i)
117         {
118             std::cout << digits[i];
119         }
120         std::cout << " E " << exponent << std::endl;
121     }
122 };
123
124 int main()
125 {
126     LongNumber num;
127     std::string input;
128
129     std::cout << "Введите число в формате (zn)0.m E N: ";
130     std::cin >> input;
131
132     try

```

```

133     {
134         num.readNumber(input);
135         LongNumber result = num.square();
136         result.printNumber();
137     }
138     catch (const std::invalid_argument &e)
139     {
140         std::cerr << "Ошибка: " << e.what() << std::endl;
141     }
142
143     return 0;
144 }

```

Результат работы:

Введите число в формате (zn)0.m E N: 1.2345E2
0.1401770592 E -4

Вывод: в ходе работы были сформированы практические навыки моделирования операций над длинными числами.