|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 №2**

**«Решение задач оптимизации при принятии решений»**

**ДИСЦИПЛИНА: «Типы и структуры данных»**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-31Б | |  |  | ( | Суриков Н. С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Былинка М. И. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

**Цель:** формирование практических навыков создания алгоритмов решения оптимизационных задач.

**Задачи:**

1. Изучить виды задач оптимизации при принятии решений.
2. Изучить основные алгоритмы для решения данных задач.
3. Реализовать алгоритм согласно варианту.

**Вариант 4**

**Формулировка задания:**

*-*

**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.