Лабораторные работы по программированию и основам алгоритмизации

Тутубалин П.И., Губайдуллин Ш.И.

ПМИ КНИТУ-КАИ

Материал подготовлен студентом группы 4112, Наумихиным Д.Д.

Содержание

1 Лабораторная работа

1.1 Задание

Дано натуральное число n. Вычислить значения функции

$$y = \frac{x^2 - 3x + 2}{\sqrt{2x^3 - 1}}$$

для $x = 1, 1.1, 1.2, \ldots, 1 + 0.1n$.

1.2 Требования

1. Дружественный интерфейс.
2. Возможность многократного ввода исходных данных необходимых для решения поставленной задачи. Программа должна спрашивать: «Хотите повторить ввод исходных данных? Да — 1, Нет — 0.» Также в некотором виде должен формироваться запрос(ы), определяющие откуда поступят исходные данные и куда будет осуществлён вывод результата.
3. Использование минимум одной функции помимо функции main.
4. Ввод исходных данных из консоли.
5. Вывод результатов в файл (путь к файлу задаётся в коде).
6. Вывод результатов в консоль.
7. В случае ввода данных из файла программа должна завершаться (не предлагать повторно ввести исходные данные).
8. Использование uniform инициализации c++.
9. Защиту от некорректного пользовательского ввода. При этом следует продумать возможные случаи такого некорректного ввода. Перечислить примеры возможных вариантов некорректного пользовательского ввода.
10. Размещение пользовательских констант в отдельном файле, например, ”constants.h”с «header guards».
11. Размещение прототипов пользовательских функций в отдельном файла, например, ”myfuncs.h”с «header guards». Допускается несколько таких файлов.
12. Размещение описаний пользовательских функций в отдельном файла, например, ”myfuncs.cpp”. Допускается несколько таких файлов.
13. Передачу статических массивов в функции(ю) в качестве параметров.

1.3 Листинги исходных файлов программы

Листинг 1: main.cpp

```cpp
#include <iostream>
#include <cmath>
#include <fstream>
#include "myfuncs.h"
#include <vector>
#include "constants.h"

int main() {
    bool DS = true;
    while (DS) {
        if (!LabsDialogs::step()) break;
        std::cout << "Repeat? 1 - Yes, 0 - No: " << std::endl;
        LabsInputFunctions::inputBool(DS, std::cin, std::cerr);
    }
    return 0;
}
```

Листинг 2: myfuncs.h

```cpp
#ifndef LABS_MYFUNCS_H
#define LABS_MYFUNCS_H

#include <iostream>
#include <cmath>
#include <fstream>

namespace LabsExceptions {
    class NotThatType : public std::exception {
    public:
        const char *what() const noexcept override {
            return "The type of input is not correct.";
        }
    };

    class ThereIsNothing : public std::exception {
    public:
        const char *what() const noexcept override {
            return "There is nothing";
        }
    };
}

namespace LabsInputFunctions {
    uint8_t inputULL(uint64_t &x, std::istream &in = std::cin,
       std::ostream &eout = std::cerr, bool printErr = true);
```

```
26      uint8_t inputNatural(uint64_t &x, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
27      uint8_t inputBool(bool &b, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
28  }
29
30  namespace LabsDialogs {
31      int step();
32  }
33  #endif //LABS_MYFUNCS_H
```

Листинг 3: myfuncs.cpp

```
1   #include "myfuncs.h"
2   #include "constants.h"
3   #include <iostream>
4   #include <string>
5
6   namespace {
7       double y(double x) {
8           return (x * x - 3 * x + 2) / (sqrt(2 * x * x * x - 1));
9       }
10
11      std::pair<double *, size_t> iXY(std::istream &in, std::ostream
           &out, std::ostream &eout, std::ostream &qout, bool
           printErr, bool printQst) {
12          size_t n = 0;
13          static double arr[arrsize];
14          if (printQst) qout << "n" << std::endl;
15          try {
16              if (LabsInputFunctions::inputNatural(n, in, eout,
                   printErr) != 0) throw
                   LabsExceptions::ThereIsNothing();
17              for (int i = 0; i <= n; ++i) {
18                  arr[i] = y(1 + 0.1 * i);
19              }
20          } catch (std::exception &err) {
21              if (printErr) eout << err.what() << std::endl;
22          }
23          return {arr, n};
24      }
25
26      void oXY(double inp[], size_t n, std::ostream &out) {
27          out << "x\ty" << std::endl;
28          for (uint64_t i = 0; i <= n; ++i) {
29              out << 1 + 0.1 * (double)i << "\t" << inp[i] <<
                   std::endl;
30          }
```

```
31      }
32
33      void ioXY(std::istream &in, std::ostream &out, std::ostream
           &eout, std::ostream &qout, bool printErr, bool printQst) {
34          auto [arr, n] = iXY(in, out, eout, qout, printErr,
               printQst);
35          if (n != 0) oXY(arr, n, out);
36      }
37
38      void consoleI(std::ostream &out) {
39          ioXY(std::cin, out, std::cerr, std::cout, true, true);
40      }
41
42      void fileI(std::ostream &out) {
43          std::ifstream fin("in.txt");
44          if (fin.fail()) throw LabsExceptions::ThereIsNothing{};
45          while (!fin.eof())
46              ioXY(fin, out, std::cerr, out, false, false);
47      }
48  }
49
50  uint8_t LabsInputFunctions::inputULL(uint64_t &x, std::istream
       &in, std::ostream &eout, bool printErr) {
51      bool f = true; std::string s;
52      do {
53          try {
54              if (in.eof()) return 1;
55              std::getline(in, s);
56              if (s.find('-') != std::string::npos) throw
                   LabsExceptions::NotThatType();
57              x = std::stoul(s);
58              f = false;
59          } catch (std::exception &err) {
60              if (printErr) eout << err.what() << "\n" << "Write an
                   unsigned integer, please" << std::endl;
61          }
62      } while (f);
63      return 0;
64  }
65
66  uint8_t LabsInputFunctions::inputNatural(uint64_t &x, std::istream
       &in, std::ostream &eout, bool printErr) {
67      bool f = true; std::string s;
68      do {
69          try {
70              uint64_t tmp;
71              if (inputULL(tmp, in, eout, printErr)) return 1;
```

```
72              if (tmp == 0) throw LabsExceptions::NotThatType();
73              x = tmp;
74              f = false;
75          } catch (std::exception &err) {
76              if (printErr) eout << err.what() << "\n" << "Write a
                    natural number, please" << std::endl;
77          }
78      } while (f);
79      return 0;
80  }
81
82  uint8_t LabsInputFunctions::inputBool(bool &b, std::istream &in,
        std::ostream &eout, bool printErr) {
83      bool f = true; std::string s;
84      do {
85          try {
86              if (in.eof()) return 1;
87              getline(in, s);
88              s.erase(remove_if(s.begin(), s.end(), ::isspace),
                    s.end());
89              if (!(s == "1"  s == "0")) throw
                    LabsExceptions::NotThatType();
90              b = (s == "1");
91              f = false;
92          } catch(std::exception &err) {
93              if (printErr) eout << err.what() << "\n" << "Write a
                    boolean value (0, 1), please" << std::endl;
94          }
95      } while(f);
96      return 0;
97  }
98
99  int LabsDialogs::step() {
100     bool cI, cO;
101     std::cout << "What would you use for input? 1 - Console, 0 -
            file: " << std::endl;
102     LabsInputFunctions::inputBool(cI,std::cin, std::cerr, true);
103     std::cout << "What would you use for output? 1 - Console, 0 -
            file: " << std::endl;
104     LabsInputFunctions::inputBool(cO, std::cin, std::cerr, true);
105
106     std::ofstream file("out.txt");
107     auto &out = (cO ? std::cout : file);
108
109     if (cI) {
110         consoleI(out);
111         return 1;
```

```
112     } else fileI(out);
113     return 0;
114 }
```

Листинг 4: constants.h

```
1 #ifndef LABS_CONSTANTS_H
2 #define LABS_CONSTANTS_H
3
4 #include <cstdint>
5
6 constexpr size_t arrsize = 100500;
7
8 #endif //LABS_CONSTANTS_H
```

1.4 Примеры выполнения программы

1. Консоль ⟶ консоль

консоль

```
1  What would you use for input? 1 - Console, 0 - file:
2  1
3  What would you use for output? 1 - Console, 0 - file:
4  1
5  n
6  8
7  x        y
8  1        0
9  1.1      -0.0698115
10 1.2      -0.102095
11 1.3      -0.113989
12 1.4      -0.113288
13 1.5      -0.104257
14 1.6      -0.0894925
15 1.7      -0.0706866
16 1.8      -0.0489959
17 Repeat? 1 - Yes, 0 - No:
18 1
19 What would you use for input? 1 - Console, 0 - file:
20 1
21 What would you use for output? 1 - Console, 0 - file:
22 1
23 n
24 2
25 x        y
26 1        0
27 1.1      -0.0698115
```

```
28 1.2      -0.102095
29 Repeat? 1 - Yes, 0 - No:
30 0
```

2. Консоль ⟶ файл

консоль

```
1 What would you use for input? 1 - Console, 0 - file:
2 1
3 What would you use for output? 1 - Console, 0 - file:
4 0
5 n
6 10
7 Repeat? 1 - Yes, 0 - No:
8 0
```

out.txt

```
1  x    y
2  1    0
3  1.1  -0.0698115
4  1.2  -0.102095
5  1.3  -0.113989
6  1.4  -0.113288
7  1.5  -0.104257
8  1.6  -0.0894925
9  1.7  -0.0706866
10 1.8  -0.0489959
11 1.9  -0.0252367
12 2    0
```

3. Файл ⟶ консоль

in.txt

```
1 8
2 5
3 1
```

консоль

```
1 What would you use for input? 1 - Console, 0 - file:
2 0
3 What would you use for output? 1 - Console, 0 - file:
4 1
5 x        y
6 1        0
7 1.1      -0.0698115
8 1.2      -0.102095
```

```
 9  1.3      -0.113989
10  1.4      -0.113288
11  1.5      -0.104257
12  1.6      -0.0894925
13  1.7      -0.0706866
14  1.8      -0.0489959
15  x        y
16  1        0
17  1.1      -0.0698115
18  1.2      -0.102095
19  1.3      -0.113989
20  1.4      -0.113288
21  1.5      -0.104257
22  x        y
23  1        0
24  1.1      -0.0698115
```

4. Файл ⟶ файл

in.txt

```
1  8
2  5
3  1
```

консоль

```
1  What would you use for input? 1 - Console, 0 - file:
2  0
3  What would you use for output? 1 - Console, 0 - file:
4  0
```

out.txt

```
1  x   y
2  1   0
3  1.1 -0.0698115
4  1.2 -0.102095
5  1.3 -0.113989
6  1.4 -0.113288
7  1.5 -0.104257
8  1.6 -0.0894925
9  1.7 -0.0706866
10 1.8 -0.0489959
11 x   y
12 1   0
13 1.1 -0.0698115
14 1.2 -0.102095
15 1.3 -0.113989
```

```
1.4 -0.113288
1.5 -0.104257
x   y
1   0
1.1 -0.0698115
```

1.5 Неккорктный ввод

1. Здесь приведены примеры обработки неправильного ввода данных со стороны пользователя (консоль):

консоль

```
What would you use for input? 1 - Console, 0 - file:
NotABoolean
The type of input is not correct.
Write a boolean value (0, 1), please
1.0
The type of input is not correct.
Write a boolean value (0, 1), please
-1
The type of input is not correct.
Write a boolean value (0, 1), please
1
What would you use for output? 1 - Console, 0 - file:
1
n
NotAnUInt
stoul
Write an unsigned integer, please
-1
The type of input is not correct.
Write an unsigned integer, please
0
The type of input is not correct.
Write a natural number, please
1.0
x       y
1       0
1.1     -0.0698115
Repeat? 1 - Yes, 0 - No:
0
```

2. Неправильный ввод (файл):

in.txt

```
NotAnUInt
-1
```

```
3 0
4 1.0
```

консоль

```
1 What would you use for input? 1 - Console, 0 - file:
2 0
3 What would you use for output? 1 - Console, 0 - file:
4 1
5 x        y
6 1        0
7 1.1      -0.0698115
```

3. Неправлиьный ввод (пустой файл):

in.txt

консоль

```
1 What would you use for input? 1 - Console, 0 - file:
2 0
3 What would you use for output? 1 - Console, 0 - file:
4 1
5 There is nothing
```

2 Лабораторная работа

2.1 Задание

Дано натуральное число n. Получить все его натуральные делители.

2.2 Требования

1. Требования 1-12 из лабораторной работы 1.
2. Передачу статических массивов в функции(ю) в качестве параметров.

2.3 Листинги исходных файлов программы

Листинг 5: main.cpp

```
#include <iostream>
#include <cmath>
#include <fstream>
#include "myfuncs.h"
#include <vector>


int main() {
    bool DS = true;
    while (DS) {
        if (!LabsDialogs::step()) break;
        std::cout << "Repeat? 1 - Yes, 0 - No: " << std::endl;
        LabsInputFunctions::inputBool(DS, std::cin, std::cerr);
    }
    return 0;
}
```

Листинг 6: myfuncs.h

```
#ifndef LABS_MYFUNCS_H
#define LABS_MYFUNCS_H

#include <iostream>
#include <cmath>
#include <fstream>

namespace LabsExceptions {
    class NotThatType : public std::exception {
    public:
        const char *what() const noexcept override {
            return "The type of input is not correct.";
        }
    };
```

```
15
16      class ThereIsNothing : public std::exception {
17      public:
18          const char *what() const noexcept override {
19              return "There is nothing";
20          }
21      };
22  }
23
24  namespace LabsInputFunctions {
25      uint8_t inputULL(uint64_t &x, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
26      uint8_t inputNatural(uint64_t &x, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
27      uint8_t inputBool(bool &b, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
28  }
29
30  namespace LabsDialogs {
31      int step();
32  }
33  #endif //LABS_MYFUNCS_H
```

Листинг 7: myfuncs.cpp

```
1   #include "myfuncs.h"
2   #include <string>
3   #include "constants.h"
4
5   namespace {
6       std::pair<uint64_t *, uint64_t> iXY(std::istream &in,
           std::ostream &out, std::ostream &eout, std::ostream &qout,
           bool printErr, bool printQst) {
7           uint64_t n = -1, kd = 0;
8           static uint64_t arr[arrsize];
9           if (printQst) qout << "n" << std::endl;
10          if (LabsInputFunctions::inputNatural(n, in, eout,
               printErr) == 0) {
11              for (int i = 1; i <= n; ++i) {
12                  if (n % i == 0) arr[kd++] = i;
13              }
14          } else {
15              eout << "There is nothing" << std::endl;
16          }
17          return {arr, kd};
18      }
19
20      void oXY(uint64_t *inp, uint64_t n, std::ostream &out) {
```

```
21          out << "Natural divisors of a number " << inp[n - 1] << "
               are:\n";
22          for (int i = 0; i < n; ++i) {
23              out << inp[i] << " ";
24          }
25          out << std::endl;
26      }
27
28      void ioXY(std::istream &in, std::ostream &out, std::ostream
           &eout, std::ostream &qout, bool printErr, bool printQst) {
29          auto [arr, kd] = iXY(in, out, eout, qout, printErr,
               printQst);
30          if (kd != 0) oXY(arr, kd, out);
31      }
32
33      void consoleI(std::ostream &out) {
34          ioXY(std::cin, out, std::cerr, std::cout, true, true);
35      }
36
37      void fileI(std::ostream &out) {
38          std::ifstream fin("in.txt");
39          if (fin.fail()) throw LabsExceptions::ThereIsNothing();
40          while (!fin.eof())
41              ioXY(fin, out, std::cerr, out, false, false);
42      }
43  }
44
45  namespace LabsInputFunctions {
46      uint8_t inputULL(uint64_t &x, std::istream &in, std::ostream
           &eout, bool printErr) {
47          bool f = true; std::string s;
48          do {
49              try {
50                  if (in.eof()) return 1;
51                  std::getline(in, s);
52                  if (s.find('-') != std::string::npos) throw
                       LabsExceptions::NotThatType();
53                  x = std::stoul(s);
54                  f = false;
55              } catch (std::exception &err) {
56                  if (printErr) eout << err.what() << "\n" << "Write
                       an unsigned integer, please" << std::endl;
57              }
58          } while (f);
59          return 0;
60      }
61
```

```
62      uint8_t inputNatural(uint64_t &x, std::istream &in,
           std::ostream &eout, bool printErr) {
63          bool f = true; std::string s;
64          do {
65              try {
66                  uint64_t tmp;
67                  if (LabsInputFunctions::inputULL(tmp, in, eout,
                       printErr)) return 1;
68                  if (tmp == 0) throw LabsExceptions::NotThatType();
69                  x = tmp;
70                  f = false;
71              } catch (std::exception &err) {
72                  if (printErr) eout << err.what() << "\n" << "Write
                       a natural number, please" << std::endl;
73              }
74          } while (f);
75          return 0;
76      }
77
78      uint8_t inputBool(bool &b, std::istream &in, std::ostream
           &eout, bool printErr) {
79          bool f = true; std::string s;
80          do {
81              try {
82                  if (in.eof()) return 1;
83                  std::getline(in, s);
84                  s.erase(std::remove_if(s.begin(), s.end(),
                       ::isspace), s.end());
85                  if (!(s == "1"  s == "0")) throw
                       LabsExceptions::NotThatType();
86                  b = (s == "1");
87                  f = false;
88              } catch(std::exception &err) {
89                  if (printErr) eout << err.what() << "\n" << "Write
                       a boolean value (0, 1), please" << std::endl;
90              }
91          } while(f);
92          return 0;
93      }
94  }
95
96
97  int LabsDialogs::step() {
98      bool cI, cO;
99      std::cout << "What would you use for input? 1 - Console, 0 -
           file: " << std::endl;
100     LabsInputFunctions::inputBool(cI, std::cin, std::cerr, true);
```

```
101     std::cout << "What would you use for output? 1 - Console, 0 -
            file: " << std::endl;
102     LabsInputFunctions::inputBool(c0, std::cin, std::cerr, true);
103
104     std::ofstream file{"out.txt"};
105     auto &out = (c0 ? std::cout : file);
106
107     if (cI) {
108         consoleI(out);
109         return 1;
110     } else fileI(out);
111     return 0;
112 }
```

Листинг 8: constants.h

```
1 #ifndef LABS_CONSTANTS_H
2 #define LABS_CONSTANTS_H
3
4 #include <cstdint>
5
6 constexpr size_t arrsize = 100500;
7
8 #endif //LABS_CONSTANTS_H
```

2.4 Примеры выполнения программы

1. Консоль ⟶ консоль

консоль

```
1  What would you use for input? 1 - Console, 0 - file:
2  1
3  What would you use for output? 1 - Console, 0 - file:
4  1
5  n
6  2596
7  Natural divisors of a number 2596 are:
8  1 2 4 11 22 44 59 118 236 649 1298 2596
9  Repeat? 1 - Yes, 0 - No:
10 0
```

3 Лабораторная работа

3.1 Задание

Дана целочисленная квадратная матрица порядка n. Найти номера строк:

a) все элементы которых - нули;

b) элементы в каждой из которых одинаковы;

c) все элементы которых четны;

d) элементы каждой из которых образуют монотонную последовательность (монотонно убывающую или монотонно возрастающую);

e) элементы которых образуют симметричные последовательности (палиндромы).

3.2 Требования

1. Требования 1-12 из лабораторной работы 1.
2. Динамическое выделение памяти.
3. Передачу динамических массивов в качестве параметров функции.

3.3 Листинги исходных файлов программы

Листинг 9: main.cpp

```
#include <iostream>
#include "myfuncs.h"

int main() {
    bool DS = true;
    while (DS) {
        if (!LabsDialogs::step()) break;
        std::cout << "Repeat? 1 - Yes, 0 - No: " << std::endl;
        LabsInputFunctions::inputBool(DS, std::cin, std::cerr);
    }
    return 0;
}
```

Листинг 10: myfuncs.h

```
#ifndef LABS_MYFUNCS_H
#define LABS_MYFUNCS_H

#include <iostream>
#include <cmath>
#include <fstream>

```

```
8  namespace LabsExceptions {
9      class NotThatType : public std::exception {
10     public:
11         const char *what() const noexcept override {
12             return "The type of input is not correct.";
13         }
14     };
15
16     class ThereIsNothing : public std::exception {
17     public:
18         const char *what() const noexcept override {
19             return "There is nothing";
20         }
21     };
22 }
23
24 namespace LabsInputFunctions {
25     uint8_t inputStreamLL(int64_t &x, std::istream &in,
          std::ostream &eout, bool printErr);
26     uint8_t inputULL(uint64_t &x, std::istream &in = std::cin,
          std::ostream &eout = std::cerr, bool printErr = true);
27     uint8_t inputNatural(uint64_t &x, std::istream &in = std::cin,
          std::ostream &eout = std::cerr, bool printErr = true);
28     uint8_t inputBool(bool &b, std::istream &in = std::cin,
          std::ostream &eout = std::cerr, bool printErr = true);
29 }
30
31 namespace LabsDialogs {
32     void ioXY(std::istream &in, std::ostream &out, std::ostream
          &eout, std::ostream &qout, bool printErr, bool printQst);
33     void consoleI(std::ostream &out);
34     void fileI(std::ostream &out);
35     int step();
36 }
37
38 #endif //LABS_MYFUNCS_H
```

Листинг 11: myfuncs.cpp

```
1  #include "myfuncs.h"
2  #include <typeinfo>
3  #include <string>
4
5  namespace LabsInputFunctions {
6      uint8_t inputStreamLL(int64_t &x, std::istream &in,
           std::ostream &eout, bool printErr) {
7          bool f = true;
8          do {
```

```
9               try {
10                  if (in.eof()) return 1;
11                  int64_t tmp;
12                  if (!(in >> tmp)) throw
                        LabsExceptions::NotThatType();
13                  x = tmp;
14                  f = false;
15              } catch (std::exception &err) {
16                  if (printErr) eout << err.what() << "\n" << "Write
                        an integer, please\n" << std::endl;
17                  in.clear();
18                  if (in.eof()) return 1;
19                  std::string tmp;
20                  in >> tmp;
21              }
22          } while (f);
23          return 0;
24      }
25
26      uint8_t inputULL(uint64_t &x, std::istream &in, std::ostream
            &eout, bool printErr) {
27          bool f = true;
28          std::string s;
29          do {
30              try {
31                  if (in.eof()) return 1;
32                  getline(in, s);
33                  if (s.find('-') != std::string::npos) throw
                        LabsExceptions::NotThatType();
34                  x = stoul(s);
35                  f = false;
36              } catch (std::exception &err) {
37                  if (printErr) eout << err.what() << "\n" << "Write
                        an unsigned integer, please" << std::endl;
38              }
39          } while (f);
40          return 0;
41      }
42
43      uint8_t inputNatural(uint64_t &x, std::istream &in,
            std::ostream &eout, bool printErr) {
44          bool f = true;
45          std::string s;
46          do {
47              try {
48                  uint64_t tmp;
49                  if (inputULL(tmp, in, eout, printErr)) return 1;
```

```
50                  if (tmp == 0) throw LabsExceptions::NotThatType();
51                  x = tmp;
52                  f = false;
53              } catch (std::exception &err) {
54                  if (printErr) eout << err.what() << "\n" << "Write
                        a natural number, please" << std::endl;
55              }
56          } while (f);
57          return 0;
58      }
59
60      uint8_t inputBool(bool &b, std::istream &in, std::ostream
            &eout, bool printErr) {
61          bool f = true;
62          std::string s;
63          do {
64              try {
65                  if (in.eof()) return 1;
66                  getline(in, s);
67                  s.erase(remove_if(s.begin(), s.end(), ::isspace),
                        s.end());
68                  if (!(s == "1"  s == "0")) throw
                        LabsExceptions::NotThatType();
69                  b = (s == "1");
70                  f = false;
71              } catch (std::exception &err) {
72                  if (printErr && !s.empty()) eout << err.what() <<
                        "\n" << "Write a boolean value (0, 1), please"
                        << std::endl;
73              }
74          } while (f);
75          return 0;
76      }
77  }
78  namespace {
79      std::pair<int64_t *, size_t> iXY(std::istream &in,
            std::ostream &eout, std::ostream &qout, bool printErr, bool
            printQst) {
80          size_t n = 0;
81          int64_t *mrx = nullptr;
82          if (printQst) qout << "n" << std::endl;
83
84          try {
85              if (LabsInputFunctions::inputULL(n, in, eout,
                    printErr) != 0) throw
                    LabsExceptions::ThereIsNothing();
86              mrx = new int64_t[n * n];
```

```
87              if (printQst) qout << "Matrix nxn:" << std::endl;
88              for (size_t i = 0; i < n * n; ++i)
89                  if (LabsInputFunctions::inputStreamLL(mrx[i], in,
                        eout, printErr) != 0)
90                      throw LabsExceptions::ThereIsNothing();
91          } catch (std::exception &err) {
92              eout << err.what() << std::endl;
93              delete[] mrx;
94              return {nullptr, 0};
95          }
96          return {mrx, n};
97      }
98
99      void oXY(const uint32_t *inp, size_t n, const std::string
            &exs, std::ostream &out) {
100         out << exs << ") ";
101         for (int i = 1; i <= n; ++i) {
102             bool f = false;
103             for (int j = 0; j < n; ++j) {
104                 if (inp[j] == i) {
105                     out << j + 1 << " ";
106                     f = true;
107                 }
108             }
109             if (f) out << "\t";
110         }
111         out << std::endl;
112     }
113 }
114
115 namespace LabsDialogs {
116     void ioXY(std::istream &in, std::ostream &out, std::ostream
            &eout, std::ostream &qout, bool printErr, bool printQst) {
117         auto [mrx, n] = iXY(in, eout, qout, printErr, printQst);
118         if (mrx == nullptr) return;
119         auto *ans = new uint32_t[n];
120         for (size_t i = 0; i < n; ++i) {
121             ans[i] = 1;
122             for (size_t j = 0; j < n; ++j) {
123                 ans[i] = ans[i] && (mrx[i * n + j] == 0);
124             }
125         }
126         oXY(ans, n, "a", out);
127         delete[] ans;
128
129         ans = new uint32_t[n];
130         for (size_t i = 0; i < n; ++i) {
```

```
131             ans[i] = 0;
132             for (size_t j = 0; j < n; ++j) {
133                 bool f = true;
134                 for (size_t k = 0; k < n; ++k) {
135                     f = f && (mrx[i * n + k] == mrx[j * n + k]);
136                 }
137                 if (f) ans[j] = i + 1;
138             }
139         }
140         oXY(ans, n, "b", out);
141         delete[] ans;
142 
143         ans = new uint32_t[n];
144         for (size_t i = 0; i < n; ++i) {
145             ans[i] = 1;
146             for (size_t j = 0; j < n; ++j) {
147                 ans[i] = ans[i] && (mrx[i * n + j] % 2 == 0);
148             }
149         }
150         oXY(ans, n, "c", out);
151         delete[] ans;
152 
153         ans = new uint32_t[n];
154         for (size_t i = 0; i < n; ++i) {
155             ans[i] = 1;
156             for (size_t j = 1; j < n; ++j) {
157                 ans[i] = ans[i] && ((mrx[i * n] < mrx[(i + 1) * n
                        - 1] && mrx[i * n + j - 1] < mrx[i * n + j])
158                                     (mrx[i * n] > mrx[(i + 1) * n
                                        - 1] && mrx[i * n + j - 1]
                                        > mrx[i * n + j]));
159             }
160         }
161         oXY(ans, n, "d", out);
162         delete[] ans;
163 
164         ans = new uint32_t[n];
165         for (size_t i = 0; i < n; ++i) {
166             ans[i] = 1;
167             for (size_t j = 0; j < n; ++j) {
168                 ans[i] = ans[i] && (mrx[i * n + j] == mrx[(i + 1)
                        * n - 1 - j]);
169             }
170         }
171         oXY(ans, n, "e", out);
172         delete[] ans;
173         delete[] mrx;
```

```
174     }
175
176     void consoleI(std::ostream &out) {
177         ioXY(std::cin, out, std::cerr, std::cout, true, true);
178     }
179
180     void fileI(std::ostream &out) {
181         std::ifstream fin("in.txt");
182         while (!fin.eof())
183             ioXY(fin, out, std::cerr, out, false, false);
184     }
185
186     int step() {
187         bool cI, cO;
188         std::cout << "What would you use for input? 1 - Console, 0
              - file: " << std::endl;
189         LabsInputFunctions::inputBool(cI, std::cin, std::cerr,
              true);
190         std::cout << "What would you use for output? 1 - Console,
              0 - file: " << std::endl;
191         LabsInputFunctions::inputBool(cO, std::cin, std::cerr,
              true);
192
193         std::ofstream file("out.txt");
194         auto &out = (cO ? std::cout : file);
195
196         if (cI) {
197             consoleI(out);
198             return 1;
199         } else fileI(out);
200         return 0;
201     }
202 }
```

Листинг 12: constants.h

```
1 #ifndef LABS_CONSTANTS_H
2 #define LABS_CONSTANTS_H
3
4 #include <cstdint>
5
6 constexpr size_t arrsize = 100500;
7
8 #endif //LABS_CONSTANTS_H
```

3.4 Примеры выполнения программы

1. Консоль ⟶ консоль

консоль

```
What would you use for input? 1 - Console, 0 - file:
1
What would you use for output? 1 - Console, 0 - file:
1
n
6
Matrix nxn:
1 2 3 3 2 1
2 4 6 6 4 2
1 2 3 4 5 6
6 5 4 3 2 1
0 0 0 0 0 0
1 2 3 3 2 1
a) 5
b) 2    3       4       5       1 6
c) 2 5
d) 3 4
e) 1 2 5 6
Repeat? 1 - Yes, 0 - No:
0
```

4 Лабораторная работа

4.1 Задание

Даны натуральное число n, символы $s_1, \ldots, s_n$. Преобразовать последовательность $s_1, \ldots, s_n$, удалив каждый символ * и повторив каждый символ, отличный от *.

4.2 Требования

1. Требования 1-3 из лабораторной работы 3.
2. работа со строками исходной длины.
3. Ввод исходных данных из файла.
4. Вывод результатов в файл (Путь к файлу задаётся с консоли, если он не указывается, то используется путь к файлу по умолчанию. То есть, например программа выдаёт такой запрос: «Укажите файл для вывода результатов работы программы [‘rez‘out.txt]:». Если просто нажать Enter, то файлом вывода будет файл « ‘rez‘out.txt»).

4.3 Листинги исходных файлов программы

Листинг 13: MExc.h

```
#ifndef LABS_MEXC_H
#define LABS_MEXC_H

#include <exception>

namespace MExc {
    class NotThatType : public std::exception {
    public:
        [[nodiscard]] const char *what() const noexcept override;
    };

    class ThereIsNothing : public std::exception {
    public:
        [[nodiscard]] const char *what() const noexcept override;
    };
}

#endif //LABS_MEXC_H
```

Листинг 14: MExc.cpp

```
#include "MExc.h"

const char *MExc::NotThatType::what() const noexcept {
    return "The type of input is not correct.";
```

```
5 }
6
7 const char *MExc::ThereIsNothing::what() const noexcept {
8     return "There is nothing";
9 }
```

Листинг 15: MyString.h

```
1  #ifndef LABS_MYSTRING_H
2  #define LABS_MYSTRING_H
3
4  #include <iostream>
5
6  class MyString {
7      size_t size_;
8      size_t capacity_;
9      char*  buffer_;
10
11     void strExt();
12     void reset();
13
14 public:
15     MyString();
16     MyString(MyString &x);
17     explicit MyString(const char *x);
18     [[nodiscard]] size_t size() const;
19     [[nodiscard]] const char* c_str() const;
20     void readStr(std::istream &in);
21     ~MyString();
22     void f(MyString &x);
23     MyString& operator=(const char *x);
24     char operator[](size_t i);
25     friend std::istream &operator>>(std::istream &in, MyString
           &string);
26     friend std::ostream &operator<<(std::ostream &os, const
           MyString &string);
27     friend MyString operator+(const MyString& a, const MyString&
           b);
28     const char *begin();
29     const char *end();
30 };
31
32
33 #endif //LABS_MYSTRING_H
```

Листинг 16: MyString.cpp

```
1 #include "MyString.h"
2 #include <sstream>
```

```
3
4  MyString::MyString() : size_(0), capacity_(64) {
5      buffer_ = new char[capacity_];
6  }
7
8  size_t MyString::size() const {
9      return size_;
10 }
11
12 const char *MyString::c_str() const {
13     return buffer_;
14 }
15
16 void MyString::readStr(std::istream &in) {
17     char buffer[4096];
18     while (in.get(buffer, 4096)) {
19         size_t size = in.gcount();
20         if (size_ + size >= capacity_) strExt();
21         memcpy(buffer_ + size_, buffer, size);
22         size_ += size;
23         if (size != 4095  in.peek() == '\n') {
24             buffer_[size_] = '\0';
25             in.ignore();
26             break;
27         }
28     }
29 }
30
31 MyString::~MyString() {
32     delete[] buffer_;
33 }
34
35 std::ostream &operator<<(std::ostream &os, const MyString &string)
      {
36     os << "size_: " << string.size_ << " capacity_: " <<
          string.capacity_ << " buffer_: " << string.buffer_;
37     return os;
38 }
39
40 void MyString::f(MyString &x) {
41     reset();
42     const char *str = x.c_str();
43     for (size_t i = 0; i < x.size(); ++i) {
44         if (str[i] != '*') {
45             if (size_ + 2 >= capacity_) strExt();
46             buffer_[size_++] = str[i];
47             buffer_[size_++] = str[i];
```

```
48          }
49      }
50      buffer_[size_] = '\0';
51  }
52
53  void MyString::strExt() {
54      capacity_ *= 2;
55      char *tmp = new char[capacity_];
56      memcpy(tmp, buffer_, size_);
57      delete[] buffer_;
58      buffer_ = tmp;
59  }
60
61  MyString::MyString(MyString &x) : size_(0), capacity_(64),
        buffer_(nullptr)  {
62      this->f(x);
63  }
64
65  std::istream &operator>>(std::istream &in, MyString &string) {
66      string.readStr(in);
67      return in;
68  }
69
70  MyString operator+(const MyString& a, const MyString& b) {
71      MyString x;
72      x.size_ = a.size_ + b.size_;
73      x.capacity_ = std::max(a.capacity_, b.capacity_);
74      if (x.size_ >= x.capacity_) x.strExt();
75      memcpy(x.buffer_, a.buffer_, a.size_);
76      memcpy(x.buffer_ + a.size_, b.buffer_, b.size_);
77      x.buffer_[x.size_] = '\0';
78      return x;
79  }
80
81  MyString::MyString(const char *x) : size_(0), capacity_(64),
        buffer_(nullptr) {
82      (*this) = x;
83  }
84
85
86  MyString& MyString::operator=(const char *x) {
87      reset();
88      std::stringstream ss;
89      ss << x;
90      ss >> (*this);
91      return *this;
92  }
```

```
93
94  void MyString::reset() {
95      size_ = 0;
96      capacity_ = 64;
97      delete []buffer_;
98      buffer_ = new char[capacity_];
99  }
100
101 const char *MyString::begin() {
102     return buffer_;
103 }
104
105 const char *MyString::end() {
106     return buffer_ + size_;
107 }
108
109 char MyString::operator[](size_t i) {
110     if (i >= size_) throw std::out_of_range("MyStr");
111     return buffer_[i];
112 }
```

Листинг 17: MIO.h

```
1  #ifndef LABS_MIO_H
2  #define LABS_MIO_H
3
4  #include <iostream>
5
6  namespace MIO {
7      uint8_t inputStreamLL(int64_t &x, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
8      uint8_t inputBool(bool &b, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
9      uint8_t inputULL(uint64_t &x, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
10     uint8_t inputNatural(uint64_t &x, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
11 }
12 #endif //LABS_MIO_H
```

Листинг 18: MIO.cpp

```
1  #include "MIO.h"
2  #include "MExc.h"
3  #include <string>
4
5  uint8_t MIO::inputStreamLL(int64_t &x, std::istream &in,
       std::ostream &eout, bool printErr) {
6      bool f = true;
```

```
7       do {
8           try {
9               if (in.eof()) return 1;
10              int64_t tmp;
11              if ((in >> tmp).fail()) throw MExc::NotThatType();
12              x = tmp;
13              f = false;
14          } catch (std::exception &err) {
15              if (printErr) eout << err.what() << "\n" << "Write an
                    integer, please\n" << std::endl;
16              in.clear();
17              if (in.eof()) return 1;
18              std::string tmp;
19              in >> tmp;
20          }
21      } while (f);
22      return 0;
23  }
24
25  uint8_t MIO::inputULL(uint64_t &x, std::istream &in, std::ostream
        &eout, bool printErr) {
26      bool f = true;
27      std::string s;
28      do {
29          try {
30              if (in.eof()) return 1;
31              getline(in, s);
32              if (s.find('-') != std::string::npos) throw
                    MExc::NotThatType();
33              x = stoul(s);
34              f = false;
35          } catch (std::exception &err) {
36              if (printErr) eout << err.what() << "\n" << "Write an
                    unsigned integer, please" << std::endl;
37          }
38      } while (f);
39      return 0;
40  }
41
42  uint8_t MIO::inputNatural(uint64_t &x, std::istream &in,
        std::ostream &eout, bool printErr) {
43      bool f = true;
44      std::string s;
45      do {
46          try {
47              uint64_t tmp;
48              if (MIO::inputULL(tmp, in, eout, printErr)) return 1;
```

```
49              if (tmp == 0) throw MExc::NotThatType();
50              x = tmp;
51              f = false;
52          } catch (std::exception &err) {
53              if (printErr) eout << err.what() << "\n" << "Write a
                    natural number, please" << std::endl;
54          }
55      } while (f);
56      return 0;
57  }
58
59  uint8_t MIO::inputBool(bool &b, std::istream &in, std::ostream
        &eout, bool printErr) {
60      bool f = true;
61      std::string s;
62      do {
63          try {
64              if (in.eof()) return 1;
65              getline(in, s);
66              s.erase(remove_if(s.begin(), s.end(), ::isspace),
                    s.end());
67              if (!(s == "1"  s == "0")) throw MExc::NotThatType();
68              b = (s == "1");
69              f = false;
70          } catch (std::exception &err) {
71              if (printErr && !s.empty()) eout << err.what() << "\n"
                    << "Write a boolean value (0, 1), please" <<
                    std::endl;
72          }
73      } while (f);
74      return 0;
75  }
```

Листинг 19: dialog.h

```
1  #ifndef LABS_MYFUNCS_H
2  #define LABS_MYFUNCS_H
3
4  #include <iostream>
5  #include <cmath>
6  #include <fstream>
7  #include "MExc.h"
8  #include <string>
9
10 using namespace std;
11
12 namespace LabsDialogs {
13     void ioXY(istream &in, ostream &out, ostream &eout, ostream
```

```
        &qout, bool printErr, bool printQst);
14      void consoleI(ostream &out);
15      void fileI(ostream &out);
16      int step();
17
18      template<typename T>
19      T fileD(istream &in, ostream &out, ostream &eout, ostream
            &qout, bool printErr, bool printQst) {
20          bool f = true;
21          T fio;
22          string name = ((typeid(T) == typeid(ifstream)) ? "in.txt"
                : "out.txt");
23          if (typeid(T) != typeid(ifstream) && typeid(T) !=
                typeid(ofstream)) throw MExc::NotThatType();
24          if (printQst)
25              qout << "Write name of file " << "(" << name << " as
                    default" << ")" << ":" << endl;
26          do {
27              try {
28                  getline(in, name);
29                  if (name.empty())
30                      name = ((typeid(T) == typeid(ifstream)) ?
                            "in.txt" : "out.txt");
31                  fio.open(name);
32                  if (fio.fail()) throw MExc::ThereIsNothing();
33                  f = false;
34              } catch (exception &err) {
35                  if (printErr) eout << err.what() << "\n" << "Write
                        name of file, please" << endl;
36
37              }
38          } while (f);
39          return fio;
40      }
41  }
42  #endif //LABS_MYFUNCS_H
```

Листинг 20: dialog.cpp

```
1   #include "dialog.h"
2   #include "MyString.h"
3   #include "MIO.h"
4
5   namespace LabsDialogs {
6       MyString iXY(istream &in, ostream &out, ostream &eout, ostream
            &qout, bool printErr, bool printQst) {
7           if (printQst) qout << "Write your string, please" << endl;
8           MyString str;
```

```
 9          in >> str;
10          return str;
11      }
12
13      void oXY(MyString &str, ostream &out) {
14          out << str.c_str() << endl;
15      }
16
17      void ioXY(istream &in, ostream &out, ostream &eout, ostream
           &qout, bool printErr, bool printQst) {
18          MyString x = iXY(in, out, eout, qout, printErr, printQst);
19          MyString y(x);
20          oXY(y, out);
21      }
22
23      void consoleI(ostream &out) {
24          ioXY(cin, out, cerr, cout, true, true);
25      }
26
27
28      void fileI(ostream &out) {
29          auto fin = fileD<ifstream>(cin, out, cerr, cout, true,
               true);
30          while (!fin.eof())
31              ioXY(fin, out, cerr, out, false, false);
32      }
33
34      int step() {
35          bool cI, cO;
36          cout << "What would you use for input? 1 - Console, 0 -
               file: " << endl;
37          MIO::inputBool(cI, cin, cerr, true);
38          cout << "What would you use for output? 1 - Console, 0 -
               file: " << endl;
39          MIO::inputBool(cO, cin, cerr, true);
40
41          ofstream file;
42          if (!cO) file = fileD<std::ofstream>(cin, cout, cerr,
               cout, true, true);
43          std::ostream &out = (cO ? cout : file);
44
45          if (cI) {
46              consoleI(out);
47              return 1;
48          } else fileI(out);
49          return 0;
50      }
```

```
51 }
```

Листинг 21: main.cpp

```
1  #include <iostream>
2  #include "dialog.h"
3  #include "MIO.h"
4
5  using namespace std;
6
7  int main() {
8      bool DS = true;
9      while (DS) {
10         if (!LabsDialogs::step()) break;
11         cout << "Repeat? 1 - Yes, 0 - No: " << endl;
12         MIO::inputBool(DS, cin, cerr);
13     }
14     return 0;
15 }
```

4.4 Примеры выполнения программы

1. Консоль ⟶ консоль

консоль

```
1 What would you use for input? 1 - Console, 0 - file:
2 1
3 What would you use for output? 1 - Console, 0 - file:
4 1
5 Write your string, please
6 Hello ** world *t t
7 HHeellllloo    wwoorrlldd  tt  tt
8 Repeat? 1 - Yes, 0 - No:
9 0
```

2. Консоль ⟶ консоль

in.txt

```
1 123456
2 101
3 Helllo *** world!
```

консоль

```
1 What would you use for input? 1 - Console, 0 - file:
2 0
3 What would you use for output? 1 - Console, 0 - file:
4 1
```

```
5 Write name of file (in.txt as default):
6
7 112233445566
8 110011
9 HHeellllllloo    wwoorrlldd!!
```

5 Лабораторная работа

5.1 Задание

Определение наличия в числе указанной битовой последовательности.

5.2 Требования

1. Дружественный интерфейс (удобство ввода данных, наглядность получаемых результатов).
2. Ввод исходных данных через параметры командной строки.
3. Реализацию задания в виде отдельной функции.
4. Корректность работы реализованной функции с целыми числами различных размерностей (от 1-го до 8 байт, при необходимости уточнять нюансы у преподавателя).

5.3 Листинги исходных файлов программы

Листинг 22: MExc.h

```h
#ifndef LABS_MEXC_H
#define LABS_MEXC_H

#include <exception>

namespace MExc {
    class NotThatType : public std::exception {
    public:
        [[nodiscard]] const char *what() const noexcept override;
    };

    class ThereIsNothing : public std::exception {
    public:
        [[nodiscard]] const char *what() const noexcept override;
    };
}

#endif //LABS_MEXC_H
```

Листинг 23: MExc.cpp

```cpp
#include "MExc.h"

const char *MExc::NotThatType::what() const noexcept {
    return "The type of input is not correct.";
}

```

```
7  const char *MExc::ThereIsNothing::what() const noexcept {
8      return "There is nothing";
9  }
```

Листинг 24: MIO.h

```
1  #ifndef LABS_MIO_H
2  #define LABS_MIO_H
3
4  #include <iostream>
5  #include <bitset>
6  #include "MExc.h"
7
8  namespace MIO {
9      uint8_t
10     inputStreamLL(int64_t &x, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
11     uint8_t inputBool(bool &b, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
12     uint8_t inputULL(uint64_t &x, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
13     uint8_t inputNatural(uint64_t &x, std::istream &in = std::cin,
           std::ostream &eout = std::cerr, bool printErr = true);
14
15     template<size_t T> uint8_t inputBS(std::bitset<T> &x,
           std::istream &in, std::ostream &eout, bool printErr) {
16         bool f = true;
17         std::bitset<T> y;
18         do {
19             try {
20                 if (in.eof()) return 1;
21                 if ((in >> y).fail()) throw MExc::NotThatType();
22                 x = y;
23                 f = false;
24             } catch (std::exception &err) {
25                 if (printErr) eout << err.what() << "\n" << "Write
                       bit sequence, please" << std::endl;
26                 in.clear();
27                 if (in.eof()) return 1;
28                 std::string tmp;
29                 in >> tmp;
30             }
31         } while (f);
32         return 0;
33     }
34 }
35
36 #endif //LABS_MIO_H
```

Листинг 25: MIO.cpp

```cpp
#include "MIO.h"
#include <string>

uint8_t MIO::inputStreamLL(int64_t &x, std::istream &in, std::ostream &eout, bool printErr) {
    bool f = true;
    do {
        try {
            if (in.eof()) return 1;
            int64_t tmp;
            if ((in >> tmp).fail()) throw MExc::NotThatType();
            x = tmp;
            f = false;
        } catch (std::exception &err) {
            if (printErr) eout << err.what() << "\n" << "Write an integer, please\n" << std::endl;
            in.clear();
            if (in.eof()) return 1;
            std::string tmp;
            in >> tmp;
        }
    } while (f);
    return 0;
}

uint8_t MIO::inputULL(uint64_t &x, std::istream &in, std::ostream &eout, bool printErr) {
    bool f = true;
    std::string s;
    do {
        try {
            if (in.eof()) return 1;
            getline(in, s);
            if (s.find('-') != std::string::npos) throw MExc::NotThatType();
            x = stoull(s);
            f = false;
        } catch (std::exception &err) {
            if (printErr) eout << err.what() << "\n" << "Write an unsigned integer, please" << std::endl;
        }
    } while (f);
    return 0;
}


```

```
42 uint8_t MIO::inputNatural(uint64_t &x, std::istream &in,
      std::ostream &eout, bool printErr) {
43     bool f = true;
44     std::string s;
45     do {
46         try {
47             uint64_t tmp;
48             if (MIO::inputULL(tmp, in, eout, printErr)) return 1;
49             if (tmp == 0) throw MExc::NotThatType();
50             x = tmp;
51             f = false;
52         } catch (std::exception &err) {
53             if (printErr) eout << err.what() << "\n" << "Write a
                  natural number, please" << std::endl;
54         }
55     } while (f);
56     return 0;
57 }
58
59 uint8_t MIO::inputBool(bool &b, std::istream &in, std::ostream
      &eout, bool printErr) {
60     bool f = true;
61     std::string s;
62     do {
63         try {
64             if (in.eof()) return 1;
65             getline(in, s);
66             s.erase(remove_if(s.begin(), s.end(), ::isspace),
                  s.end());
67             if (!(s == "1"  s == "0")) throw MExc::NotThatType();
68             b = (s == "1");
69             f = false;
70         } catch (std::exception &err) {
71             if (printErr && !s.empty()) eout << err.what() << "\n"
                  << "Write a boolean value (0, 1), please" <<
                  std::endl;
72         }
73     } while (f);
74     return 0;
75 }
```

Листинг 26: dialog.h

```
1 #ifndef LABS_MYFUNCS_H
2 #define LABS_MYFUNCS_H
3
4 #include <iostream>
5 #include <cmath>
```

```
6  #include <fstream>
7
8  using namespace std;
9
10 namespace LabsDialogs {
11     void ioXY(istream &in, ostream &out, ostream &eout, ostream
          &qout, bool printErr, bool printQst);
12     void consoleI(ostream &out);
13     void fileI(ostream &out);
14     int step();
15
16     template<typename T>
17     T fileD(istream &in, ostream &out, ostream &eout, ostream
          &qout, bool printErr, bool printQst) {
18         bool f = true;
19         T fio;
20         string name = ((typeid(T) == typeid(ifstream)) ? "in.txt"
              : "out.txt");
21         if (typeid(T) != typeid(ifstream) && typeid(T) !=
              typeid(ofstream)) throw MExc::NotThatType();
22         if (printQst)
23             qout << "Write name of file " << "(" << name << " as
                  default" << ")" << ":" << endl;
24         do {
25             try {
26                 getline(in, name);
27                 if (name.empty())
28                     name = ((typeid(T) == typeid(ifstream)) ?
                          "in.txt" : "out.txt");
29                 fio.open(name);
30                 if (fio.fail()) throw MExc::ThereIsNothing();
31                 f = false;
32             } catch (exception &err) {
33                 if (printErr) eout << err.what() << "\n" << "Write
                      name of file, please" << endl;
34
35             }
36         } while (f);
37         return fio;
38     }
39 }
40 #endif //LABS_MYFUNCS_H
```

Листинг 27: dialog.cpp

```
1 #include "dialog.h"
2 #include "MIO.h"
3 #include <bitset>
```

```
4  #include <string>
5
6  namespace LabsDialogs {
7      pair<uint64_t, uint64_t> iXY(istream &in, ostream &out,
           ostream &eout, ostream &qout, bool printErr, bool printQst)
           {
8          if (printQst) qout << "Unsigned integer: " << endl;
9          uint64_t n;
10         MIO::inputULL(n, in, eout, printErr);
11         bitset<sizeof(n)*8> bits;
12         if (printQst) qout << "Binary subsequence: " << endl;
13         MIO::inputBS<sizeof(n)*8>(bits, in, eout, printErr);
14         return {n, bits.to_ullong()};
15     }
16
17     void ioXY(istream &in, ostream &out, ostream &eout, ostream
           &qout, bool printErr, bool printQst) {
18         auto [x, y] = iXY(in, out, eout, qout, printErr, printQst);
19         bitset<sizeof(x)*8> bb(x);
20         out << bb << "\n";
21         bb = y;
22         out << bb << endl;
23         size_t msb, sz = sizeof(x)*8;
24         for (msb = 0; (y >> msb) != 0; ++msb);
25
26         bool f = false;
27         for (size_t i = 0; i < sz - msb; ++i) {
28             f = f  (((x << (sz - msb - i)) >> (sz - msb)) == y);
29         }
30         out << (f ? "There is the subsequence" : "There isn't the
               subsequence") << endl;
31     }
32
33     void consoleI(ostream &out) {
34         ioXY(cin, out, cerr, cout, true, true);
35     }
36
37     ifstream fileD(istream &in, ostream &out, ostream &eout,
           ostream &qout, bool printErr, bool printQst) {
38         bool f = true;
39         ifstream fin;
40         if (printQst) qout << "Write name of file (in.txt as
               default):" << endl;
41         do {
42             try {
43                 string name;
44                 getline(in, name);
```

```
45                  if (name.empty())
46                      name = "in.txt";
47                  fin.open(name);
48                  if (fin.fail()) throw MExc::ThereIsNothing();
49                  f = false;
50              } catch (exception &err) {
51                  if (printErr) eout << err.what() << "\n" << "Write
                       name of file, please" << endl;
52              }
53          } while (f);
54          return fin;
55      }
56
57      void fileI(ostream &out) {
58          ifstream fin = fileD(cin, out, cerr, cout, true, true);
59          while (!fin.eof())
60              ioXY(fin, out, cerr, out, false, false);
61      }
62
63      int step() {
64          bool cI, cO;
65          cout << "What would you use for input? 1 - Console, 0 -
               file: " << endl;
66          MIO::inputBool(cI, cin, cerr, true);
67          cout << "What would you use for output? 1 - Console, 0 -
               file: " << endl;
68          MIO::inputBool(cO, cin, cerr, true);
69
70          ofstream file;
71          if (!cO) file = fileD<std::ofstream>(cin, cout, cerr,
               cout, true, true);
72          std::ostream &out = (cO ? cout : file);
73
74          if (cI) {
75              consoleI(out);
76              return 1;
77          } else fileI(out);
78          return 0;
79      }
80  }
```

Листинг 28: main.cpp

```
1  #include <iostream>
2  #include <sstream>
3  #include "dialog.h"
4  #include "MIO.h"
5
```

```
6  using namespace std;
7
8  int main(int argc, char **argv) {
9      try {
10         if (argc < 3) throw MExc::ThereIsNothing();
11         stringstream sst;
12         for (int i = 1; i < argc; ++i) sst << argv[i] << "\n";
13         LabsDialogs::ioXY(sst, cout, cerr, cout, false, false);
14     } catch (exception &err) {
15         cerr << err.what() << "\n" << "Give me the number and some
               sequence, please" << endl;
16     }
17     return 0;
18 }
```

5.4 Примеры выполнения программы

1. Консоль ⟶ консоль

консоль (здесь есть сокращения)

```
1  >  .\Labs.exe 7 10
2  000...0000111
3  000...0000010
4  There isn't the subsequence
5  >  .\Labs.exe 7 11
6  000...0000111
7  000...0000011
8  There is the subsequence
```

5.5 Неккорктный ввод

1. Недостаточное количество аргументов:

консоль

```
1  >  .\Labs.exe
2  There is nothing
3  Give me the number and some sequence, please
4  >  .\Labs.exe 1
5  There is nothing
6  Give me the number and some sequence, please
```

6 Лабораторная работа

6.1 Задание

В лабораторной работе нужно реализовать в программе работу с некоторой структурой представления данных (однонаправленный список, содержит в себе {ФИО; Должность; Место проживания}), то есть обеспечить доступ к её данным в рамках заданного перечня функций:

1. Подсчитать: сколько имеется элементов с заданным содержимым одного из полей;
2. Печать всех элементов (вывод на консоль);
3. Сброс значений всех элементов (например обнуление);
4. Чтение из файла, запись в файл.

6.2 Требования

1. Дружественный интерфейс (удобство ввода данных, наглядность получаемых результатов);
2. Возможность проверки (использования) всех функций реализованных в программе без её перезапуска, то есть должно быть реализовано некоторое «консольное» меню (выводится список вариантов действия и предлагается сделать выбор);
3. Для сохранения списка в программе предусмотреть текстовый файл (для чтения из него и записи в него). Файл должен содержать все элементы списка со всеми их полями, представленные в табличном виде;

6.3 Листинги исходных файлов программы

Листинг 29: MExc.h

```
#ifndef LABS_MEXC_H
#define LABS_MEXC_H

#include <exception>

namespace MExc {
    class NotThatType : public std::exception {
    public:
        [[nodiscard]] const char *what() const noexcept override;
    };

    class ThereIsNothing : public std::exception {
    public:
        [[nodiscard]] const char *what() const noexcept override;
    };
}

#endif //LABS_MEXC_H
```

Листинг 30: MExc.cpp

```cpp
#include "MExc.h"
#include <vector>

const char *MExc::NotThatType::what() const noexcept {
    return "The type of input is not correct.";
}

const char *MExc::ThereIsNothing::what() const noexcept {
    return "There is nothing";
}
```

Листинг 31: MIO.h

```cpp
#ifndef LABS_MIO_H
#define LABS_MIO_H

#include <iostream>

namespace MIO {
    uint8_t inputStreamLL(int64_t &x, std::istream &in = std::cin,
       std::ostream &eout = std::cerr, bool printErr = true);
    uint8_t inputBool(bool &b, std::istream &in = std::cin,
       std::ostream &eout = std::cerr, bool printErr = true);
    uint8_t inputULL(uint64_t &x, std::istream &in = std::cin,
       std::ostream &eout = std::cerr, bool printErr = true);
    uint8_t inputNatural(uint64_t &x, std::istream &in = std::cin,
       std::ostream &eout = std::cerr, bool printErr = true);
    uint8_t chooseVariant(uint64_t &x, uint64_t border,
       std::istream &in = std::cin, std::ostream &eout =
       std::cerr, bool printErr = true);
}
#endif //LABS_MIO_H
```

Листинг 32: MIO.cpp

```cpp
#include "MIO.h"
#include "MExc.h"
#include <string>

uint8_t MIO::inputStreamLL(int64_t &x, std::istream &in,
   std::ostream &eout, bool printErr) {
    bool f = true;
    do {
        try {
            if (in.eof()) return 1;
            int64_t tmp;
            if ((in >> tmp).fail()) throw MExc::NotThatType();
```

```
12              x = tmp;
13              f = false;
14          } catch (std::exception &err) {
15              if (printErr) eout << err.what() << "\n" << "Write an
                    integer, please\n" << std::endl;
16              in.clear();
17              if (in.eof()) return 1;
18              std::string tmp;
19              in >> tmp;
20          }
21      } while (f);
22      return 0;
23  }
24
25  uint8_t MIO::inputULL(uint64_t &x, std::istream &in, std::ostream
        &eout, bool printErr) {
26      bool f = true;
27      std::string s;
28      do {
29          try {
30              if (in.eof()) return 1;
31              getline(in, s);
32              if (s.find('-') != std::string::npos) throw
                    MExc::NotThatType();
33              x = stoul(s);
34              f = false;
35          } catch (std::exception &err) {
36              if (printErr) eout << err.what() << "\n" << "Write an
                    unsigned integer, please" << std::endl;
37          }
38      } while (f);
39      return 0;
40  }
41
42  uint8_t MIO::inputNatural(uint64_t &x, std::istream &in,
        std::ostream &eout, bool printErr) {
43      bool f = true;
44      std::string s;
45      do {
46          try {
47              uint64_t tmp;
48              if (MIO::inputULL(tmp, in, eout, printErr)) return 1;
49              if (tmp == 0) throw MExc::NotThatType();
50              x = tmp;
51              f = false;
52          } catch (std::exception &err) {
53              if (printErr) eout << err.what() << "\n" << "Write a
```

```
                natural number, please" << std::endl;
54          }
55      } while (f);
56      return 0;
57  }
58
59  uint8_t MIO::inputBool(bool &b, std::istream &in, std::ostream
        &eout, bool printErr) {
60      bool f = true;
61      std::string s;
62      do {
63          try {
64              if (in.eof()) return 1;
65              getline(in, s);
66              s.erase(remove_if(s.begin(), s.end(), ::isspace),
                    s.end());
67              if (!(s == "1"  s == "0")) throw MExc::NotThatType();
68              b = (s == "1");
69              f = false;
70          } catch (std::exception &err) {
71              if (printErr && !s.empty())
72                  eout << err.what() << "\n" << "Write a boolean
                        value (0, 1), please" << std::endl;
73          }
74      } while (f);
75      return 0;
76  }
77
78  uint8_t MIO::chooseVariant(uint64_t &x, uint64_t border,
        std::istream &in, std::ostream &eout, bool printErr) {
79      bool f = true;
80      std::string s;
81      do {
82          try {
83              uint64_t tmp;
84              if (MIO::inputNatural(tmp, in, eout, printErr)) return
                    1;
85              if (tmp > border) throw MExc::NotThatType();
86              x = tmp;
87              f = false;
88          } catch (std::exception &err) {
89              if (printErr)
90                  eout << err.what() << "\n" << "Write a natural
                        number not greater than " << border <<
                        std::endl;
91          }
92      } while (f);
```

```
93      return 0;
94  }
```

Листинг 33: MLL.h

```
1   #ifndef LABS_MLL_H
2   #define LABS_MLL_H
3
4   #include <iostream>
5   #include <string>
6
7   struct Node {
8       Node *next = nullptr;
9       std::string NSF_, job_, place_;
10      friend std::istream &operator>>(std::istream &in, Node &n);
11      friend bool operator==(const Node &a, const Node &b);
12      friend std::ostream &operator<<(std::ostream &out, Node &n);
13  };
14
15  class MLL {
16      Node *first_ = nullptr, *last_ = nullptr;
17
18  public:
19      bool empty();
20      void r_f();
21      void clean();
22      void print(std::ostream &out = std::cout, std::ostream &eout =
            std::cerr, std::ostream &qout = std::cout);
23      void printT(std::ostream &out = std::cout, std::ostream &eout
            = std::cerr, std::ostream &qout = std::cout);
24      void push_back(Node *node);
25      uint64_t cnt(std::string &s, uint64_t type);
26      ~MLL();
27
28      friend std::istream &operator>>(std::istream &in, MLL &mll);
29      friend std::ostream &operator<<(std::ostream &out, MLL &mll);
30  };
31
32
33  #endif //LABS_MLL_H
```

Листинг 34: MLL.cpp

```
1   #include "MLL.h"
2   #include "MExc.h"
3   #include <iomanip>
4
5   std::istream &operator>>(std::istream &in, Node &n) {
6       try {
```

```
7           std::getline(in, n.NSF_, '$');
8           std::getline(in, n.job_, '$');
9           std::getline(in, n.place_, '$');
10      }
11      catch (std::exception &err) {}
12      return in;
13  }
14
15  bool operator==(const Node &a, const Node &b) {
16      return a.NSF_ == b.NSF_ && a.job_ == b.job_ && a.place_ ==
            b.place_;
17  }
18
19  std::ostream &operator<<(std::ostream &out, Node &n) {
20      out << "{"
21          << "\"Full name\": " << "\"" << n.NSF_ << "\"" << ", "
22          << "\"Job\": " << "\"" << n.job_ << "\"" << ", "
23          << "\"Place\": " << "\"" << n.place_ << "\"" << "}";
24      return out;
25  }
26
27  bool MLL::empty() {
28      return first_ == nullptr;
29  }
30
31  void MLL::r_f() {
32      if (empty()) return;
33      Node *p = first_;
34      first_ = p->next;
35      delete p;
36  }
37
38  void MLL::clean() {
39      while (!empty()) r_f();
40  }
41
42  void MLL::print(std::ostream &out, std::ostream &eout,
        std::ostream &qout) {
43      for (Node *p = first_; p != nullptr; p = p->next)
44          out << *p << ", ";
45  }
46
47  void MLL::printT(std::ostream &out, std::ostream &eout,
        std::ostream &qout) {
48      out << std::setw(32) << std::left << "Full name"
49          << std::setw(32) << std::left << "Job"
50          << std::setw(32) << std::left << "Place" << '\n';
```

```
51      for (Node *p = first_; p != nullptr; p = p->next)
52          out << std::left << std::setw(32) << p->NSF_
53              << std::left << std::setw(32) << p->job_
54              << std::left << std::setw(32) << p->place_ << '\n';
55      out << std::endl;
56  }
57
58  void MLL::push_back(Node *node) {
59      if (empty()) {
60          first_ = node;
61          last_ = node;
62          return;
63      }
64      last_->next = node;
65      last_ = node;
66  }
67
68  uint64_t MLL::cnt(std::string &s, uint64_t type) {
69      uint64_t res = 0;
70      if (type > 2) throw MExc::NotThatType();
71      for (Node *p = first_; p != nullptr; p = p->next) {
72          std::string *ss[] = {&p->NSF_, &p->job_, &p->place_};
73          if (s == *ss[type]) ++res;
74      }
75      return res;
76  }
77
78  MLL::~MLL() {
79      clean();
80  }
81
82  std::istream &operator>>(std::istream &in, MLL &mll) {
83      Node *n = new Node;
84      in >> *n;
85      mll.push_back(n);
86      return in;
87  }
88
89  std::ostream &operator<<(std::ostream &out, MLL &mll) {
90      for (const Node *p = mll.first_; p != nullptr; p = p->next)
91          out << p->NSF_ << "$" << p->job_ << "$" << p->place_ << (p
                != mll.last_ ? "$" : "");
92      return out;
93  }
```

Листинг 35: dialog.h

```
1  #ifndef LABS_DIALOG_H
```

```
2  #define LABS_DIALOG_H
3
4  #include <iostream>
5  #include <fstream>
6  #include <functional>
7  #include "MExc.h"
8  #include "MIO.h"
9
10 using namespace std;
11
12 namespace LabsDialogs {
13     struct Ioeqpp {
14         istream &in;
15         ostream &out, &eout, &qout;
16         bool printErr, printQst;
17     };
18
19     void ioXY(Ioeqpp &iqp);
20
21     void consoleI(ostream &out);
22
23     void fileI(ostream &out);
24
25     int step();
26
27     template<typename T=ifstream>
28     T fileD(Ioeqpp &iqp);
29
30     template<typename T>
31     T fileD(Ioeqpp &iqp) {
32         bool f = true;
33         T fio;
34         string name = ((typeid(T) == typeid(ifstream)) ? "in.txt"
               : "out.txt");
35         if (typeid(T) != typeid(ifstream) && typeid(T) !=
               typeid(ofstream)) throw MExc::NotThatType();
36         if (iqp.printQst)
37             iqp.qout << "Write name of file " << "(" << name << "
                   as default" << ")" << ":" << endl;
38         do {
39             try {
40                 getline(iqp.in, name);
41                 if (name.empty())
42                     name = ((typeid(T) == typeid(ifstream)) ?
                           "in.txt" : "out.txt");
43                 fio.open(name);
44                 if (fio.fail()) throw MExc::ThereIsNothing();
```

```
45                  f = false;
46              } catch (exception &err) {
47                  if (iqp.printErr) iqp.eout << err.what() << "\n"
                       << "Write name of file, please" << endl;
48
49              }
50          } while (f);
51          return fio;
52      }
53
54      template<class T>
55      struct FuncWithDesc {
56          std::string desc;
57          std::function<T> func;
58      };
59
60      template<class T>
61      uint64_t optionD(vector<FuncWithDesc<T>> &options, Ioeqpp
           &iqp) {
62          if (iqp.printQst) {
63              iqp.qout << "Choose an option:\n";
64              for (int i = 0; i < options.size(); ++i) {
65                  iqp.qout << i + 1 << " - " << options[i].desc <<
                       ";\n";
66              }
67          }
68          uint64_t option;
69          MIO::chooseVariant(option, options.size(), iqp.in,
               iqp.eout, iqp.printErr);
70          return option - 1;
71      }
72  }
73
74  #endif //LABS_DIALOG_H
```

Листинг 36: dialog.cpp

```
1   #include "dialog.h"
2   #include "MIO.h"
3   #include "MLL.h"
4   #include <vector>
5
6   namespace LabsDialogs {
7       void cntD(MLL *mll, Ioeqpp &iqp) {
8           if (iqp.printQst) iqp.qout << "Write string, which ends by
               \'$\' sign:" << endl;
9           string str;
10          getline(iqp.in, str, '$');
```

```
11          iqp.in.ignore();
12          vector<FuncWithDesc<uint64_t(MLL *, string &)>> options = {
13                  {"same full name", [](MLL *mll, string &str) {
                       return mll->cnt(str, 0); }},
14                  {"same job",       [](MLL *mll, string &str) {
                       return mll->cnt(str, 1); }},
15                  {"same place",     [](MLL *mll, string &str) {
                       return mll->cnt(str, 2); }},
16          };
17          iqp.out << options[optionD<uint64_t(MLL *, string
              &)>(options, iqp)].func(mll, str) << endl;
18      }
19
20      void ioXY(Ioeqpp &iqp) {
21          auto *mll = new MLL;
22          vector<FuncWithDesc<bool(MLL *, Ioeqpp &)>> options = {
23                  {"print elements",     [](MLL *mll, Ioeqpp &iqp) {
24                      mll->print(iqp.out);
25                      iqp.qout << endl;
26                      return true;
27                  }},
28                  {"print table",        [](MLL *mll, Ioeqpp &iqp) {
29                      mll->printT(iqp.out);
30                      return true;
31                  }},
32                  {"count coincidences", [](MLL *mll, Ioeqpp &iqp) {
33                      cntD(mll, iqp);
34                      return true;
35                  }},
36                  {"reset",              [](MLL *mll, Ioeqpp &iqp) {
37                      mll->clean();
38                      return true;
39                  }},
40                  {"read file",          [](MLL *mll, Ioeqpp &iqp) {
41                      auto fin = fileD<ifstream>(iqp);
42                      while (!fin.eof()) fin >> *mll;
43                      fin.close();
44                      return true;
45                  }},
46                  {"write to file",      [](MLL *mll, Ioeqpp &iqp) {
47                      auto fout = fileD<ofstream>(iqp);
48                      fout << *mll;
49                      fout.close();
50                      return true;
51                  }},
52                  {"exit",               [](MLL *mll, Ioeqpp &iqp) {
                       return false; }}
```

```
53          };
54
55          bool f = true;
56          while (f) f = options[optionD<bool(MLL *, Ioeqpp
               &)>(options, iqp)].func(mll, iqp);
57
58          delete mll;
59      }
60
61      void consoleI(ostream &out) {
62          Ioeqpp iqp = {cin, out, cerr, cout, true, true};
63          ioXY(iqp);
64      }
65
66
67      void fileI(ostream &out) {
68          Ioeqpp iqp1 = {cin, out, cerr, cout, true, true};
69          ifstream fin = fileD(iqp1);
70          Ioeqpp iqp2 = {fin, out, cerr, out, false, false};
71          while (!fin.eof())
72              ioXY(iqp2);
73      }
74
75
76      int step() {
77          Ioeqpp iqp1 = {cin, cout, cerr, cout, true, true};
78          bool cI, cO;
79          iqp1.qout << "What would you use for input? 1 - Console, 0
               - file: " << endl;
80          MIO::inputBool(cI, iqp1.in, iqp1.eout, true);
81          iqp1.qout << "What would you use for output? 1 - Console,
               0 - file: " << endl;
82          MIO::inputBool(cO, iqp1.in, iqp1.eout, true);
83
84          ofstream file;
85          if (!cO) file = fileD<std::ofstream>(iqp1);
86          std::ostream &out = (cO ? cout : file);
87
88          if (cI) {
89              consoleI(out);
90              return 1;
91          } else fileI(out);
92          return 0;
93      }
94  }
```

Листинг 37: main.cpp

```cpp
#include <iostream>
#include "dialog.h"
#include "MIO.h"

using namespace std;

int main() {
    bool DS = true;
    while (DS) {
        if (!LabsDialogs::step()) break;
        cout << "Repeat? 1 - Yes, 0 - No: " << endl;
        MIO::inputBool(DS, cin, cerr);
    }
    return 0;
}
```

6.4 Примеры выполнения программы

1. Консоль ⟶ консоль

консоль

```
What would you use for input? 1 - Console, 0 - file:
1
What would you use for output? 1 - Console, 0 - file:
1
Choose an option:
1 - print elements;
2 - print table;
3 - count coincidences;
4 - reset;
5 - read file;
6 - write to file;
7 - exit;
5
Write name of file (in.txt as default):

Choose an option:
1 - print elements;
2 - print table;
3 - count coincidences;
4 - reset;
5 - read file;
6 - write to file;
7 - exit;
1
{"Full name": "NDD", "Job": "Student", "Place": "Kazan"},
    {"Full name": "TPI", "Job": "Lecturer", "Place": "Kazan"},
```

```
      {"Full name": "GSI", "Job": "Tutor", "Place": "Kazan"},
Choose an option:
1 - print elements;
2 - print table;
3 - count coincidences;
4 - reset;
5 - read file;
6 - write to file;
7 - exit;
2
Full name                       Job
   Place
NDD                              Student
                            Kazan
TPI                              Lecturer
                           Kazan
GSI                              Tutor
                              Kazan

Choose an option:
1 - print elements;
2 - print table;
3 - count coincidences;
4 - reset;
5 - read file;
6 - write to file;
7 - exit;
7
Repeat? 1 - Yes, 0 - No:
0
```

in.txt

```
NDD$Student$Kazan$TPI$Lecturer$Kazan$GSI
   $Tutor$Kazan$NDD$Student$Kazan$TPI$
   Lecturer$Kazan$GSI$Tutor$Kazan
```

2. Файл ⟶ файл

in1.txt

```
5
in.txt
1
7
```

консоль

```
What would you use for input? 1 - Console, 0 - file:
```

```
0
What would you use for output? 1 - Console, 0 - file:
0
Write name of file (out.txt as default):

Write name of file (in.txt as default):
in1.txt
```

in.txt

```
NDD$Student$Kazan$TPI$Lecturer$Kazan$GSI
   $Tutor$Kazan$NDD$Student$Kazan$TPI$
   Lecturer$Kazan$GSI$Tutor$Kazan
```

out.txt

```
{"Full name": "NDD", "Job": "Student", "Place": "Kazan"},
   {"Full name": "TPI", "Job": "Lecturer", "Place": "Kazan"},
   {"Full name": "GSI", "Job": "Tutor", "Place": "Kazan"},
   {"Full name": "NDD", "Job": "Student", "Place": "Kazan"},
   {"Full name": "TPI", "Job": "Lecturer", "Place": "Kazan"},
   {"Full name": "GSI", "Job": "Tutor", "Place": "Kazan"},
```