

# Лабораторные работы по программированию и основам алгоритмизации

Тутубалин П.И., Губайдуллин Ш.И.

---

ПМИ КНИТУ-КАИ

# Содержание

<b>1</b>	<b>Собственные статические библиотеки</b>	<b>2</b>
1.1	MIO . . . . .	2
1.2	MDialogs . . . . .	6
<b>2</b>	<b>version.h</b>	<b>10</b>
<b>3</b>	<b>Лабораторная работа 1</b>	<b>11</b>
3.1	Задание . . . . .	11
3.2	Требования . . . . .	11
3.3	Листинги исходных файлов программы . . . . .	12
3.4	Примеры выполнения программы . . . . .	13
3.5	Некорктный ввод . . . . .	16
<b>4</b>	<b>Лабораторная работа 2</b>	<b>19</b>
4.1	Задание . . . . .	19
4.2	Требования . . . . .	19
4.3	Листинги исходных файлов программы . . . . .	19
4.4	Примеры выполнения программы . . . . .	21
<b>5</b>	<b>Лабораторная работа 3</b>	<b>22</b>
5.1	Задание . . . . .	22
5.2	Требования . . . . .	22
5.3	Листинги исходных файлов программы . . . . .	22
5.4	Примеры выполнения программы . . . . .	32
<b>6</b>	<b>Лабораторная работа 4</b>	<b>34</b>
6.1	Задание . . . . .	34
6.2	Требования . . . . .	34
6.3	Листинги исходных файлов программы . . . . .	34
6.4	Примеры выполнения программы . . . . .	39
<b>7</b>	<b>Лабораторная работа 5</b>	<b>40</b>
7.1	Задание . . . . .	40
7.2	Требования . . . . .	40
7.3	Листинги исходных файлов программы . . . . .	40
7.4	Примеры выполнения программы . . . . .	42
7.5	Некорктный ввод . . . . .	42
<b>8</b>	<b>Лабораторная работа 6</b>	<b>43</b>
8.1	Задание . . . . .	43
8.2	Требования . . . . .	43
8.3	Листинги исходных файлов программы . . . . .	43
8.4	Примеры выполнения программы . . . . .	48

# 1 Собственные статические библиотеки

В лабораторных работах используются следующие библиотеки: MIO и MDialogs

## 1.1 MIO

Эта библиотека предназначена для обозначения собственных исключений (MExc) и обработки пользовательского ввода (MIO):

Листинг 1: Исключения, MExc.h

```
1 #ifndef LABS_MEXC_H
2 #define LABS_MEXC_H
3
4 #include <exception>
5
6 namespace MExc {
7     class NotThatType : public std::exception {
8     public:
9         [[nodiscard]] const char *what() const noexcept override;
10    };
11
12    class ThereIsNothing : public std::exception {
13    public:
14        [[nodiscard]] const char *what() const noexcept override;
15    };
16 }
17
18 #endif //LABS_MEXC_H
```

Листинг 2: Исключения, MExc.cpp

```
1 #include "MExc.h"
2 #include <vector>
3
4 const char *MExc::NotThatType::what() const noexcept {
5     return "The type of input is not correct.";
6 }
7
8 const char *MExc::ThereIsNothing::what() const noexcept {
9     return "There is nothing";
10 }
```

Листинг 3: Пользовательский ввод, 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 inputStreamLL(int64_t &x, std::istream &in = std::cin,
        std::ostream &out = std::cerr, bool printErr = true);
10     uint8_t inputBool(bool &b, std::istream &in = std::cin,
        std::ostream &out = std::cerr, bool printErr = true);
11     uint8_t inputULL(uint64_t &x, std::istream &in = std::cin,
        std::ostream &out = std::cerr, bool printErr = true);
12     uint8_t inputNatural(uint64_t &x, std::istream &in = std::cin,
        std::ostream &out = std::cerr, bool printErr = true);
13     uint8_t chooseVariant(uint64_t &x, uint64_t border,
        std::istream &in = std::cin, std::ostream &out =
        std::cerr, bool printErr = true);
14
15     template<size_t T> uint8_t inputBS(std::bitset<T> &x,
        std::istream &in, std::ostream &out, 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) out << 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 #endif //LABS_MIO_H

```

Листинг 4: Пользовательский ввод, MIO.cpp

```

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

```

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

```

49         x = tmp;
50         f = false;
51     } catch (std::exception &err) {
52         if (printErr) eout << err.what() << "\n" << "Write a
            natural number, please" << std::endl;
53     }
54 } while (f);
55 return 0;
56 }
57
58 uint8_t MIO::inputBool(bool &b, std::istream &in, std::ostream
    &eout, bool printErr) {
59     bool f = true;
60     std::string s;
61     do {
62         try {
63             if (in.eof()) return 1;
64             getline(in, s);
65             s.erase(remove_if(s.begin(), s.end(), ::isspace),
                s.end());
66             if (!(s == "1" || s == "0")) throw MExc::NotThatType();
67             b = (s == "1");
68             f = false;
69         } catch (std::exception &err) {
70             if (printErr && !s.empty())
71                 eout << err.what() << "\n" << "Write a boolean
                    value (0, 1), please" << std::endl;
72         }
73     } while (f);
74     return 0;
75 }
76
77 uint8_t MIO::chooseVariant(uint64_t &x, uint64_t border,
    std::istream &in, std::ostream &eout, bool printErr) {
78     bool f = true;
79     std::string s;
80     do {
81         try {
82             uint64_t tmp;
83             if (MIO::inputNatural(tmp, in, eout, printErr)) return
                1;
84             if (tmp > border) throw MExc::NotThatType();
85             x = tmp;
86             f = false;
87         } catch (std::exception &err) {
88             if (printErr)
89                 eout << err.what() << "\n" << "Write a natural

```

```

        number not greater than " << border <<
        std::endl;
90     }
91     } while (f);
92     return 0;
93 }

```

## 1.2 MDialogs

В этой библиотеке представлены средства для ведения диалога с пользователем. Эта библиотека публично зависит от МЮ (это значит, что файлы из МЮ нужны как для этой библиотеки, так и для пользователей этой библиотеки):

```

project(MDialogs)

set(SRCS
    dialog.cpp
)
add_library(MDialogs STATIC ${SRCS})
target_link_libraries(MDialogs PUBLIC MIO)

```

Листинг 5: dialog.h

```

1  #ifndef LABS_DIALOG_H
2  #define LABS_DIALOG_H
3
4  #include <iostream>
5  #include <fstream>
6  #include <functional>
7  #include "MIO.h"
8  #include "MExc.h"
9  #include <string>
10
11
12 struct Dialog {
13     struct Ioeqpp {
14         std::istream &in;
15         std::ostream &out, &eout, &qout;
16         bool printErr, printQst;
17     };
18
19     virtual void ioXY(Ioeqpp &iqp) = 0;
20
21     void consoleI(std::ostream &out);
22

```

```
23     void fileI(std::ostream &out);
24
25     int step();
26
27     template<typename T=std::ifstream>
28     static T fileD(Ioeqpp &iqp) {
29         bool f = true;
30         T fio;
31         std::string name = ((typeid(T) == typeid(std::ifstream)) ?
32             "in.txt" : "out.txt");
33         if (typeid(T) != typeid(std::ifstream) && typeid(T) !=
34             typeid(std::ofstream)) throw MExc::NotThatType();
35         if (iqp.printQst)
36             iqp.qout << "Write name of file " << "(" << name << "
37                 as default" << ")" << ":" << std::endl;
38         do {
39             try {
40                 std::getline(iqp.in, name);
41                 if (name.empty())
42                     name = ((typeid(T) == typeid(std::ifstream)) ?
43                         "in.txt" : "out.txt");
44                 fio.open(name);
45                 if (fio.fail()) throw MExc::ThereIsNothing();
46                 f = false;
47             } catch (std::exception &err) {
48                 if (iqp.printErr) iqp.eout << err.what() << "\n"
49                     << "Write name of file, please" << std::endl;
50             }
51         } while (f);
52         return fio;
53     }
54
55     template<class T>
56     struct FuncWithDesc {
57         std::string desc;
58         std::function<T> func;
59     };
60
61     template<class T>
62     static uint64_t optionD(std::vector<FuncWithDesc<T>> &options,
63         Ioeqpp &iqp) {
64         if (iqp.printQst) {
65             iqp.qout << "Choose an option:\n";
66             for (int i = 0; i < options.size(); ++i) {
67                 iqp.qout << i + 1 << " - " << options[i].desc <<
68                     ";\n";
69             }
70         }
71     }
```



```

63         }
64     }
65     uint64_t option;
66     MIO::chooseVariant(option, options.size(), iqp.in,
67         iqp.eout, iqp.printErr);
68     return option - 1;
69 }
70 virtual ~Dialog() = default;
71 };
72
73 #endif //LABS_DIALOG_H

```

---

Листинг 6: dialog.cpp

---

```

1  #include "dialog.h"
2
3  void Dialog::consoleI(std::ostream &out) {
4      Ioeqpp iqp = {std::cin, out, std::cerr, std::cout, true, true};
5      ioXY(iqp);
6  }
7
8
9  void Dialog::fileI(std::ostream &out) {
10     Ioeqpp iqp1 = {std::cin, out, std::cerr, std::cout, true,
11         true};
12     std::ifstream fin = fileD(iqp1);
13     Ioeqpp iqp2 = {fin, out, std::cerr, out, false, false};
14     while (!fin.eof())
15         ioXY(iqp2);
16 }
17
18 int Dialog::step() {
19     Ioeqpp iqp1 = {std::cin, std::cout, std::cerr, std::cout,
20         true, true};
21     bool cI, cO;
22     iqp1.qout << "What would you use for input? 1 - Console, 0 -
23         file: " << std::endl;
24     MIO::inputBool(cI, iqp1.in, iqp1.eout, true);
25     iqp1.qout << "What would you use for output? 1 - Console, 0 -
26         file: " << std::endl;
27     MIO::inputBool(cO, iqp1.in, iqp1.eout, true);
28
29     std::ofstream file;
30     if (!cO) file = fileD<std::ofstream>(iqp1);
31     std::ostream &out = (cO ? std::cout : file);

```

```
30     if (cI) {  
31         consoleI(out);  
32         return 1;  
33     } else fileI(out);  
34     return 0;  
35 }
```

---

## 2 version.h

В файле version.h после сборки находится информация насчёт версии сборки и хеша коммита. В дереве можно найти шаблон для этого файла «version.h.in», который будет заполняться при помощи cmake.

Листинг 7: version.h.in

---

```
1 #ifndef VERSION_H
2 #define VERSION_H
3
4 #define GIT_COMMIT_HASH "@GIT_COMMIT_HASH@"
5 #define LABS_VERSION "@CMAKE_PROJECT_VERSION@"
6 #define LABS_VERSION_MAJOR @CMAKE_PROJECT_VERSION_MAJOR@
7 #define LABS_VERSION_MINOR @CMAKE_PROJECT_VERSION_MINOR@
8
9 #endif
```

---

Вот пример готового хедера:

Листинг 8: version.h

---

```
1 #ifndef VERSION_H
2 #define VERSION_H
3
4 #define GIT_COMMIT_HASH "76b83e7"
5 #define LABS_VERSION "1.1"
6 #define LABS_VERSION_MAJOR 1
7 #define LABS_VERSION_MINOR 1
8
9 #endif
```

---

## 3 Лабораторная работа 1

### 3.1 Задание

Дано натуральное число  $n$ . Вычислить значения функции

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

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

### 3.2 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 файле, например, `"myfuncs.h"` с «header guards». Допускается несколько таких файлов.
12. Размещение описаний пользовательских функций в отдельном файле, например, `"myfuncs.cpp"`. Допускается несколько таких файлов.
13. Передачу статических массивов в функции(ю) в качестве параметров.

### 3.3 Листинги исходных файлов программы

Листинг 9: main.cpp

---

```

1  #include "myfuncs.h"
2  #include "version.h"
3
4  int main() {
5      std::cout << "Currently you use the " << LABS_VERSION << "
        version of the Labs.\n" << "Git commit hash: " <<
        GIT_COMMIT_HASH << "." << std::endl;
6      MDialog dialog{};
7      bool DS = true;
8      while (DS) {
9          if (!dialog.step()) break;
10         std::cout << "Repeat? 1 - Yes, 0 - No: " << std::endl;
11         MIO::inputBool(DS, std::cin, std::cerr);
12     }
13     return 0;
14 }
```

---

Листинг 10: myfuncs.h

---

```

1  #ifndef LAB1_MYFUNCS_H
2  #define LAB1_MYFUNCS_H
3
4  #include <iostream>
5  #include <cmath>
6  #include <fstream>
7  #include "dialog.h"
8
9  struct MDialog : public Dialog {
10     void ioXY(Ioeqpp &iqp) override;
11 };
12
13 #endif//LAB1_MYFUNCS_H
```

---

Листинг 11: myfuncs.cpp

---

```

1  #include "myfuncs.h"
2  #include "constants.h"
3  #include "MExc.h"
4  #include "MIO.h"
5  #include <iostream>
6  #include <string>
7  #include "dialog.h"
8
9  namespace {
10     double y(double x) {
```

---

```

11         return (x * x - 3 * x + 2) / (sqrt(2 * x * x * x - 1));
12     }
13
14     std::pair<double *, size_t> iXY(Dialog::Ioeqpp &iqp) {
15         size_t n = 0;
16         static double arr[arrsize];
17         if (iqp.printQst) iqp.qout << "n" << std::endl;
18         try {
19             if (MIO::inputNatural(n, iqp.in, iqp.eout,
20                 iqp.printErr) != 0) throw MExc::ThereIsNothing();
21             for (int i = 0; i <= n; ++i) {
22                 arr[i] = y(1 + 0.1 * i);
23             }
24         } catch (std::exception &err) {
25             if (iqp.printErr) iqp.eout << err.what() << std::endl;
26         }
27         return {arr, n};
28     }
29
30     void oXY(double inp[], size_t n, std::ostream &out) {
31         out << "x\ty" << std::endl;
32         for (uint64_t i = 0; i <= n; ++i) {
33             out << 1 + 0.1 * (double)i << "\t" << inp[i] <<
34                 std::endl;
35         }
36     }
37
38     void MDialog::ioXY(Ioeqpp &iqp) {
39         auto [arr, n] = iXY(iqp);
40         if (n != 0) oXY(arr, n, iqp.out);
41     }

```

Листинг 12: constants.h

```

1  #ifndef LAB1_CONSTANTS_H
2  #define LAB1_CONSTANTS_H
3
4  #include <cstdint>
5
6  constexpr size_t arrsize = 100500;
7
8  #endif //LAB1_CONSTANTS_H

```

### 3.4 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
16 1.4 -0.113288
17 1.5 -0.104257
18 x    y
19 1    0
20 1.1 -0.0698115
```

---

### 3.5 Неккорктный ввод

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

#### КОНСОЛЬ

---

```
1 What would you use for input? 1 - Console, 0 - file:
2 NotABoolean
3 The type of input is not correct.
4 Write a boolean value (0, 1), please
5 1.0
6 The type of input is not correct.
7 Write a boolean value (0, 1), please
```

---

```

8 -1
9 The type of input is not correct.
10 Write a boolean value (0, 1), please
11 1
12 What would you use for output? 1 - Console, 0 - file:
13 1
14 n
15 NotAnUInt
16 stoul
17 Write an unsigned integer, please
18 -1
19 The type of input is not correct.
20 Write an unsigned integer, please
21 0
22 The type of input is not correct.
23 Write a natural number, please
24 1.0
25 x      y
26 1      0
27 1.1    -0.0698115
28 Repeat? 1 - Yes, 0 - No:
29 0

```

## 2. Неправильный ввод (файл):

in.txt

```

1 NotAnUInt
2 -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. Неправильный 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
```

---

## 4 Лабораторная работа 2

### 4.1 Задание

Дано натуральное число  $n$ . Получить все его натуральные делители.

### 4.2 Требования

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

### 4.3 Листинги исходных файлов программы

Листинг 13: main.cpp

---

```

1  #include "myfuncs.h"
2  #include "version.h"
3
4  int main() {
5      std::cout << "Currently you use the " << LABS_VERSION << "
6          version of the Labs.\n" << "Git commit hash: " <<
7          GIT_COMMIT_HASH << "." << std::endl;
8      MDialog dialog{};
9      bool DS = true;
10     while (DS) {
11         if (!dialog.step()) break;
12         std::cout << "Repeat? 1 - Yes, 0 - No: " << std::endl;
13         MID::inputBool(DS, std::cin, std::cerr);
14     }
15     return 0;
16 }
```

---

Листинг 14: myfuncs.h

---

```

1  #ifndef LAB2_MYFUNCS_H
2  #define LAB2_MYFUNCS_H
3
4  #include <iostream>
5  #include <cmath>
6  #include <fstream>
7  #include "dialog.h"
8
9  struct MDialog : public Dialog {
10     void ioXY(Ioeqpp &iqp) override;
11 };
12
13 #endif//LAB2_MYFUNCS_H
```

---

Листинг 15: myfuncs.cpp

```

1  #include "myfuncs.h"
2  #include "constants.h"
3  #include "MExc.h"
4  #include "MIO.h"
5  #include <string>
6  #include "dialog.h"
7
8
9  namespace {
10     std::pair<uint64_t *, uint64_t> iXY(Dialog::Ioeqpp &iqp) {
11         uint64_t n = -1, kd = 0;
12         static uint64_t arr[arrsize];
13         if (iqp.printQst) iqp.qout << "n" << std::endl;
14         if (MIO::inputNatural(n, iqp.in, iqp.eout, iqp.printErr)
15             == 0) {
16             for (int i = 1; i <= n; ++i) {
17                 if (n % i == 0) arr[kd++] = i;
18             }
19         } else {
20             iqp.eout << "There is nothing" << std::endl;
21         }
22         return {arr, kd};
23     }
24
25     void oXY(uint64_t *inp, uint64_t n, std::ostream &out) {
26         out << "Natural divisors of a number " << inp[n - 1] << "
27             are:\n";
28         for (int i = 0; i < n; ++i) {
29             out << inp[i] << " ";
30         }
31         out << std::endl;
32     }
33
34     void MDialog::ioXY(Ioeqpp &iqp) {
35         auto [arr, n] = iXY(iqp);
36         if (n != 0) oXY(arr, n, iqp.out);
37     }

```

Листинг 16: constants.h

```

1  #ifndef LAB2_CONSTANTS_H
2  #define LAB2_CONSTANTS_H
3
4  #include <cstdint>
5
6  constexpr size_t arrsize = 100500;

```

```
7  
8 #endif //LAB2_CONSTANTS_H
```

---

## 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	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

---

## 5 Лабораторная работа 3

### 5.1 Задание

Дана целочисленная квадратная матрица порядка  $n$ . Найти номера строк:

- все элементы которых - нули;
- элементы в каждой из которых одинаковы;
- все элементы которых четны;
- элементы каждой из которых образуют монотонную последовательность (монотонно убывающую или монотонно возрастающую);
- элементы которых образуют симметричные последовательности (палиндромы).

### 5.2 Требования

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

### 5.3 Листинги исходных файлов программы

Листинг 17: main.cpp

```

1  #include "myfuncs.h"
2  #include "version.h"
3
4  int t();
5
6  int main() {
7      std::cout << "Currently you use the " << LABS_VERSION << "
          version of the Labs.\n" << "Git commit hash: " <<
          GIT_COMMIT_HASH << "." << std::endl;
8      MDialog dialog{};
9      bool DS = true;
10     while (DS) {
11         if (!dialog.step()) break;
12         std::cout << "Repeat? 1 - Yes, 0 - No: " << std::endl;
13         MID::inputBool(DS, std::cin, std::cerr);
14     }
15     return 0;
16 }
```

Листинг 18: Matrix.h

```

1  #ifndef LAB3_MATRIX_H
2  #define LAB3_MATRIX_H
3
4  #include <cstdint>
5
6  class Matrix final {
7      private:
8          size_t rows_, cols_;
9          int64_t* data_;
10
11     class ProxyRow {
12     private:
13         int64_t* rowStart_;
14         size_t cols_;
15
16     public:
17         ProxyRow(int64_t* rowStart, size_t cols);
18
19         int64_t &operator[](size_t col);
20         const int64_t &operator[](size_t col) const;
21     };
22     public:
23         size_t rows() const;
24         size_t cols() const;
25
26         Matrix(size_t _rows, size_t _cols);
27         ~Matrix();
28
29         ProxyRow operator[](size_t row);
30         const ProxyRow operator[](size_t row) const;
31
32         Matrix(const Matrix& other);
33
34         Matrix& operator=(const Matrix& other);
35
36         Matrix(Matrix&& other) noexcept;
37
38         Matrix& operator=(Matrix&& other) noexcept;
39     };
40
41 #endif //LAB3_MATRIX_H

```

Листинг 19: Matrix.cpp

```

1  #include "Matrix.h"
2  #include <memory>
3

```



```
4 Matrix::ProxyRow::ProxyRow(int64_t* rowStart, size_t cols) :  
    rowStart_(rowStart), cols_(cols) {}  
5  
6 int64_t &Matrix::ProxyRow::operator[](size_t col) {  
7     return rowStart_[col];  
8 }  
9  
10 const int64_t &Matrix::ProxyRow::operator[](size_t col) const {  
11     return rowStart_[col];  
12 }  
13  
14 size_t Matrix::rows() const {  
15     return rows_;  
16 }  
17  
18 size_t Matrix::cols() const {  
19     return cols_;  
20 }  
21  
22 Matrix::Matrix(size_t _rows, size_t _cols) : rows_(rows_),  
    cols_(cols_) {  
23     data_ = new int64_t[rows_ * cols_];  
24 }  
25  
26 Matrix::~~Matrix() {  
27     delete[] data_;  
28 }  
29  
30 Matrix::ProxyRow Matrix::operator[](size_t row) {  
31     return ProxyRow(data_ + row * cols_, cols_);  
32 }  
33  
34  
35 const Matrix::ProxyRow Matrix::operator[](size_t row) const {  
36     return ProxyRow(data_ + row * cols_, cols_);  
37 }  
38  
39 Matrix::Matrix(const Matrix& other) : rows_(other.rows_),  
    cols_(other.cols_) {  
40     data_ = new int64_t[rows_ * cols_];  
41     std::copy(other.data_, other.data_ + (rows_ * cols_), data_);  
42 }  
43  
44 Matrix& Matrix::operator=(const Matrix& other) {  
45     if (this != &other) {  
46         delete[] data_;  
47         rows_ = other.rows_;
```

```

48         cols_ = other.cols_;
49         data_ = new int64_t[rows_ * cols_];
50         std::copy(other.data_, other.data_ + (rows_ * cols_),
                    data_);
51     }
52     return *this;
53 }
54
55 Matrix::Matrix(Matrix&& other) noexcept : rows_(other.rows_),
        cols_(other.cols_), data_(other.data_) {
56     other.rows_ = 0;
57     other.cols_ = 0;
58     other.data_ = nullptr;
59 }
60
61 Matrix& Matrix::operator=(Matrix&& other) noexcept {
62     if (this != &other) {
63         std::swap(rows_, other.rows_);
64         std::swap(cols_, other.cols_);
65         std::swap(data_, other.data_);
66     }
67     return *this;
68 }

```

#### Листинг 20: myfuncs.h

```

1  #ifndef LAB3_MYFUNCS_H
2  #define LAB3_MYFUNCS_H
3
4  #include <iostream>
5  #include <cmath>
6  #include <fstream>
7  #include "dialog.h"
8
9  struct MDialog : public Dialog {
10     void ioXY(Ioeqpp &iqp) override;
11 };
12
13 #endif // LAB3_MYFUNCS_H

```

#### Листинг 21: myfuncs.cpp, общая часть

```

1  #include "myfuncs.h"
2  #include "constants.h"
3  #include "MExc.h"
4  #include "MIO.h"
5  #include <stdint.h>
6  #include <string>
7  #include <vcruntime.h>

```

```

8 #include "dialog.h"
9
10 #if defined USE_VECTOR
11 #include "Matrix.h"

```

Различные реализации:

Листинг 22: USE VECTOR, myfuncs.cpp

```

1 namespace {
2     Matrix iXY(Dialog::Ioeqpp &iqp) {
3         try {
4             if (iqp.printQst) iqp.qout << "n" << std::endl;
5             size_t n;
6             if (MIO::inputULL(n, iqp.in, iqp.eout, iqp.printErr)
7                 != 0) throw MExc::ThereIsNothing();
8             Matrix mrx{n, n};
9             if (iqp.printQst) iqp.qout << "Matrix nxn" <<
10                 std::endl;
11             for (size_t i = 0; i < n; ++i) {
12                 for (size_t j = 0; j < n; ++j) {
13                     if (MIO::inputStreamLL(mrx[i][j], iqp.in,
14                         iqp.eout, iqp.printErr) != 0)
15                         throw MExc::ThereIsNothing();
16                 }
17             }
18             return mrx;
19         } catch (std::exception &err) {
20             iqp.eout << err.what() << std::endl;
21         }
22         return {0, 0};
23     }
24
25     void oXY(const std::vector<int64_t> &vr, const std::string
26         &exc, std::ostream &out) {
27         out << exc << " ";
28         for (size_t i = 1, endi = vr.size(); i <= endi; ++i) {
29             bool f = false;
30             for (size_t j = 0, endj = vr.size(); j < endj; ++j) {
31                 if (vr[j] == i) {
32                     out << j + 1 << " ";
33                     f = true;
34                 }
35             }
36             if (f) out << "\t";
37         }
38         out << std::endl;
39     }
40 }

```

```
37 void MDialog::ioXY(Dialog::Ioeqpp &iqp) {
38     Matrix mrx = iXY(iqp);
39
40     auto ans = std::vector<int64_t>(mrx.cols());
41     for (size_t i = 0, n = ans.size(); i < n; ++i) {
42         ans[i] = 1;
43         for (size_t j = 0; j < n; ++j) {
44             ans[i] = ans[i] && (mrx[i][j] == 0);
45         }
46     }
47     oXY(ans, "a", iqp.out);
48
49     ans = std::vector<int64_t>(mrx.cols());
50     for (size_t i = 0, n = ans.size(); i < n; ++i) {
51         ans[i] = 0;
52         for (size_t j = 0; j < n; ++j) {
53             bool f = true;
54             for (size_t k = 0; k < n; ++k) {
55                 f = f && (mrx[i][k] == mrx[j][k]);
56             }
57             if (f) ans[j] = i + 1;
58         }
59     }
60     oXY(ans, "b", iqp.out);
61
62     ans = std::vector<int64_t>(mrx.cols());
63     for (size_t i = 0, n = ans.size(); i < n; ++i) {
64         ans[i] = 1;
65         for (size_t j = 0; j < n; ++j) {
66             ans[i] = ans[i] && (mrx[i][j] % 2 == 0);
67         }
68     }
69     oXY(ans, "c", iqp.out);
70
71     ans = std::vector<int64_t>(mrx.cols());
72     for (size_t i = 0, n = ans.size(); i < n; ++i) {
73         ans[i] = 1;
74         for (size_t j = 1; j < n; ++j) {
75             ans[i] = ans[i] && ((mrx[i][0] < mrx[i][n - 1] &&
76                 mrx[i][j - 1] < mrx[i][j])
77                 (mrx[i][0] > mrx[i][n - 1] &&
78                 mrx[i][j - 1] > mrx[i][j]));
79         }
80     }
81     oXY(ans, "d", iqp.out);
82
83     ans = std::vector<int64_t>(mrx.cols());
```

```

82     for (size_t i = 0, n = ans.size(); i < n; ++i) {
83         ans[i] = 1;
84         for (size_t j = 0; j < n; ++j) {
85             ans[i] = ans[i] && (mrx[i][j] == mrx[i][n - j - 1]);
86         }
87     }
88     oXY(ans, "e", iqp.out);
89 }

```

Листинг 23: USE ONLY MATRIX, myfuncs.cpp

```

1  #include "Matrix.h"
2  namespace {
3      Matrix iXY(Dialog::Ioeqpp &iqp) {
4          try {
5              if (iqp.printQst) iqp.qout << "n" << std::endl;
6              size_t n;
7              if (MIO::inputULL(n, iqp.in, iqp.eout, iqp.printErr)
                  != 0) throw MExc::ThereIsNothing();
8              Matrix mrx{n, n};
9              if (iqp.printQst) iqp.qout << "Matrix nxn" <<
                  std::endl;
10             for (size_t i = 0; i < n; ++i) {
11                 for (size_t j = 0; j < n; ++j) {
12                     if (MIO::inputStreamLL(mrx[i][j], iqp.in,
                                                iqp.eout, iqp.printErr) != 0)
13                         throw MExc::ThereIsNothing();
14                 }
15             }
16             return mrx;
17         } catch (std::exception &err) {
18             iqp.eout << err.what() << std::endl;
19         }
20         return {0, 0};
21     }
22
23     void oXY(const Matrix &mrx, const std::string &exc,
              std::ostream &out) {
24         out << exc << ") ";
25         for (size_t i = 1, endi = mrx.cols(); i <= endi; ++i) {
26             bool f = false;
27             for (size_t j = 0, endj = mrx.cols(); j < endj; ++j) {
28                 if (mrx[0][j] == i) {
29                     out << j + 1 << " ";
30                     f = true;
31                 }
32             }
33             if (f) out << "\t";

```

```

34         }
35         out << std::endl;
36     }
37 }
38 void MDialog::ioXY(Dialog::Ioeqpp &iqp) {
39     Matrix mrx = iXY(iqp);
40     auto ans = Matrix(1, mrx.cols());
41     for (size_t i = 0, n = mrx.cols(); i < n; ++i) {
42         ans[0][i] = 1;
43         for (size_t j = 0; j < n; ++j) {
44             ans[0][i] = ans[0][i] && (mrx[i][j] == 0);
45         }
46     }
47     oXY(ans, "a", iq.p.out);
48
49     ans = {1, mrx.cols()};
50     for (size_t i = 0, n = ans.cols(); i < n; ++i) {
51         ans[0][i] = 0;
52         for (size_t j = 0; j < n; ++j) {
53             bool f = true;
54             for (size_t k = 0; k < n; ++k) {
55                 f = f && (mrx[i][k] == mrx[j][k]);
56             }
57             if (f) ans[0][j] = i + 1;
58         }
59     }
60     oXY(ans, "b", iq.p.out);
61
62     ans = {1, mrx.cols()};
63     for (size_t i = 0, n = ans.cols(); i < n; ++i) {
64         ans[0][i] = 1;
65         for (size_t j = 0; j < n; ++j) {
66             ans[0][i] = ans[0][i] && (mrx[i][j] % 2 == 0);
67         }
68     }
69     oXY(ans, "c", iq.p.out);
70
71     ans = {1, mrx.cols()};
72     for (size_t i = 0, n = ans.cols(); i < n; ++i) {
73         ans[0][i] = 1;
74         for (size_t j = 1; j < n; ++j) {
75             ans[0][i] = ans[0][i] && ((mrx[i][0] < mrx[i][n - 1]
76                                     && mrx[i][j - 1] < mrx[i][j])
77                                     (mrx[i][0] > mrx[i][n - 1] &&
78                                     mrx[i][j - 1] > mrx[i][j]));
79         }
80     }
81 }

```

```

79     oXY(ans, "d", iqp.out);
80
81     ans = {1, mrx.cols()};
82     for (size_t i = 0, n = ans.cols(); i < n; ++i) {
83         ans[0][i] = 1;
84         for (size_t j = 0; j < n; ++j) {
85             ans[0][i] = ans[0][i] && (mrx[i][j] == mrx[i][n - j -
                1]);
86         }
87     }
88     oXY(ans, "e", iqp.out);

```

Листинг 24: Первая реализация, myfuncs.cpp

```

1  #else
2  namespace {
3      std::pair<int64_t *, size_t> iXY(Dialog::Ioeqpp &iqp) {
4          size_t n = 0;
5          int64_t *mrx = nullptr;
6
7          try {
8              if (iqp.printQst) iqp.qout << "n" << std::endl;
9              if (MIO::inputULL(n, iqp.in, iqp.eout, iqp.printErr)
                  != 0) throw MExc::ThereIsNothing();
10             mrx = new int64_t[n * n];
11             if (iqp.printQst) iqp.qout << "Matrix nxn:" <<
                std::endl;
12             for (size_t i = 0; i < n * n; ++i)
13                 if (MIO::inputStreamLL(mrx[i], iqp.in, iqp.eout,
                    iqp.printErr) != 0)
14                     throw MExc::ThereIsNothing();
15             } catch (std::exception &err) {
16                 iqp.eout << err.what() << std::endl;
17                 delete[] mrx;
18                 return {nullptr, 0};
19             }
20             return {mrx, n};
21         }
22
23     void oXY(const uint32_t *inp, size_t n, const std::string
        &exs, std::ostream &out) {
24         out << exs << ") ";
25         for (int i = 1; i <= n; ++i) {
26             bool f = false;
27             for (int j = 0; j < n; ++j) {
28                 if (inp[j] == i) {
29                     out << j + 1 << " ";
30                     f = true;

```

```

31         }
32     }
33     if (f) out << "\t";
34 }
35 out << std::endl;
36 }
37 }
38
39 void MDialog::ioXY(Ioeqpp &iqp) {
40     auto [mrX, n] = iXY(iqp);
41     if (mrX == nullptr) return;
42     auto *ans = new uint32_t[n];
43     for (size_t i = 0; i < n; ++i) {
44         ans[i] = 1;
45         for (size_t j = 0; j < n; ++j) {
46             ans[i] = ans[i] && (mrX[i * n + j] == 0);
47         }
48     }
49     oXY(ans, n, "a", iqp.out);
50     delete[] ans;
51
52     ans = new uint32_t[n];
53     for (size_t i = 0; i < n; ++i) {
54         ans[i] = 0;
55         for (size_t j = 0; j < n; ++j) {
56             bool f = true;
57             for (size_t k = 0; k < n; ++k) {
58                 f = f && (mrX[i * n + k] == mrX[j * n + k]);
59             }
60             if (f) ans[j] = i + 1;
61         }
62     }
63     oXY(ans, n, "b", iqp.out);
64     delete[] ans;
65
66     ans = new uint32_t[n];
67     for (size_t i = 0; i < n; ++i) {
68         ans[i] = 1;
69         for (size_t j = 0; j < n; ++j) {
70             ans[i] = ans[i] && (mrX[i * n + j] % 2 == 0);
71         }
72     }
73     oXY(ans, n, "c", iqp.out);
74     delete[] ans;
75
76     ans = new uint32_t[n];
77     for (size_t i = 0; i < n; ++i) {

```



```

78     ans[i] = 1;
79     for (size_t j = 1; j < n; ++j) {
80         ans[i] = ans[i] && ((mrx[i * n] < mrx[(i + 1) * n - 1]
81             && mrx[i * n + j - 1] < mrx[i * n + j])
82             (mrx[i * n] > mrx[(i + 1) * n - 1]
83             && mrx[i * n + j - 1] > mrx[i *
84                 n + j]));
85     }
86 }
87 oXY(ans, n, "d", iqp.out);
88 delete[] ans;
89
90 ans = new uint32_t[n];
91 for (size_t i = 0; i < n; ++i) {
92     ans[i] = 1;
93     for (size_t j = 0; j < n; ++j) {
94         ans[i] = ans[i] && (mrx[i * n + j] == mrx[(i + 1) * n
95             - 1 - j]);
96     }
97 }
98 oXY(ans, n, "e", iqp.out);
99 delete[] ans;
100 delete[] mrx;

```

Листинг 25: constants.h

```

1 #ifndef LAB3_CONSTANTS_H
2 #define LAB3_CONSTANTS_H
3
4 #include <cstdint>
5
6 constexpr size_t arrsize = 100500;
7
8 #endif //LAB3_CONSTANTS_H

```

## 5.4 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 6
7 Matrix nxn:

```

```
8 1 2 3 3 2 1
9 2 4 6 6 4 2
10 1 2 3 4 5 6
11 6 5 4 3 2 1
12 0 0 0 0 0 0
13 1 2 3 3 2 1
14 a) 5
15 b) 2      3      4      5      1 6
16 c) 2 5
17 d) 3 4
18 e) 1 2 5 6
19 Repeat? 1 - Yes, 0 - No:
20 0
```

---

## 6 Лабораторная работа 4

### 6.1 Задание

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

### 6.2 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»).

### 6.3 Листинги исходных файлов программы

Листинг 26: main.cpp

```

1  #include "myfuncs.h"
2  #include "version.h"
3
4  int main() {
5      std::cout << "Currently you use the " << LABS_VERSION << "
        version of the Labs.\n" << "Git commit hash: " <<
        GIT_COMMIT_HASH << "." << std::endl;
6      MDialog dialog{};
7      bool DS = true;
8      while (DS) {
9          if (!dialog.step()) break;
10         std::cout << "Repeat? 1 - Yes, 0 - No: " << std::endl;
11         MID::inputBool(DS, std::cin, std::cerr);
12     }
13     return 0;
14 }
```

Листинг 27: myfuncs.h

```

1  #ifndef LAB4_MYFUNCS_H
2  #define LAB4_MYFUNCS_H
3
4  #include <iostream>
5  #include <cmath>
6  #include <fstream>
```

```

7  #include "dialog.h"
8
9  struct MDialog : public Dialog {
10      void ioXY(Ioeqpp &iqp) override;
11  };
12
13 #endif//LAB4_MYFUNCS_H

```

---

Листинг 28: myfuncs.cpp

---

```

1  #include "myfuncs.h"
2  #include "MyString.h"
3
4  namespace {
5      MyString iXY(Dialog::Ioeqpp &iqp) {
6          if (iqp.printQst) iqp.qout << "Write your string, please"
              << std::endl;
7          MyString str;
8          iqp.in >> str;
9          return str;
10     }
11
12     void oXY(MyString &str, std::ostream &out) {
13         out << str.c_str() << std::endl;
14     }
15 }
16
17 void MDialog::ioXY(Ioeqpp &iqp) {
18     MyString x = iXY(iqp);
19     MyString y(x);
20     oXY(y, iqp.out);
21 }

```

---

Листинг 29: myfuncs.h

---

```

1  #ifndef LAB4_MYSTRING_H
2  #define LAB4_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 //LAB4_MYSTRING_H

```

#### Листинг 30: myfuncs.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 {
37     os << "size_: " << string.size_ << " capacity_: " <<
38         string.capacity_ << " buffer_: " << string.buffer_;
39     return os;
40 }
41
42 void MyString::f(MyString &x) {
43     reset();
44     const char *str = x.c_str();
45     for (size_t i = 0; i < x.size(); ++i) {
46         if (str[i] != '\\*') {
47             if (size_ + 2 >= capacity_) strExt();
48             buffer_[size_++] = str[i];
49             buffer_[size_++] = str[i];
50         }
51     }
52     buffer_[size_] = '\\0';
53 }
54
55 void MyString::strExt() {
56     capacity_ *= 2;
57     char *tmp = new char[capacity_];
58     memcpy(tmp, buffer_, size_);
59     delete[] buffer_;
60     buffer_ = tmp;
61 }
62
63 MyString::MyString(MyString &x) : size_(0), capacity_(64),
64     buffer_(nullptr) {
65     this->f(x);
66 }
67
68 std::istream &operator>>(std::istream &in, MyString &string) {
69     string.readStr(in);
70     return in;
71 }
```

```
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 }
```

## 6.4 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 Hello *** 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!!

```

---



## 7 Лабораторная работа 5

### 7.1 Задание

Определение наличия в числе указанной битовой последовательности.

### 7.2 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).

### 7.3 Листинги исходных файлов программы

Листинг 31: main.cpp

```

1  #include <iostream>
2  #include <sstream>
3  #include "LAB4/myfuncs.h"
4  #include "dialog.h"
5  #include "MIO.h"
6  #include "version.h"
7
8
9  int main(int argc, char **argv) {
10     std::cout << "Currently you use the " << LABS_VERSION << "
        version of the Labs.\n" << "Git commit hash: " <<
        GIT_COMMIT_HASH << "." << std::endl;
11     MDialog dialog{};
12     try {
13         if (argc < 3) throw MExc::ThereIsNothing();
14         std::stringstream sst;
15         for (int i = 1; i < argc; ++i) sst << argv[i] << "\n";
16         Dialog::Ioeqpp iqp{sst, std::cout, std::cerr, std::cout,
            false, false};
17         dialog.ioXY(iqp);
18     } catch (std::exception &err) {
19         std::cerr << err.what() << "\n" << "Give me the number and
            some sequence, please" << std::endl;
20     }
21     return 0;
22 }
```

Листинг 32: myfuncs.h

```

1  #ifndef LAB5_MYFUNCS_H
2  #define LAB5_MYFUNCS_H
3
4  #include <iostream>
5  #include <cmath>
6  #include <fstream>
7  #include "dialog.h"
8
9  struct MDialog : public Dialog {
10     void ioXY(Ioeqpp &iqp) override;
11 };
12
13 #endif//LAB5_MYFUNCS_H

```

Листинг 33: myfuncs.cpp

```

1  #include "myfuncs.h"
2  #include "MIO.h"
3
4  namespace {
5  std::pair<uint64_t, uint64_t> iXY(Dialog::Ioeqpp &iqp) {
6      if (iqp.printQst) iqp.qout << "Unsigned integer: " <<
          std::endl;
7      uint64_t n;
8      MIO::inputULL(n, iqp.in, iqp.eout, iqp.printErr);
9      std::bitset<sizeof(n)*8> bits;
10     if (iqp.printQst) iqp.qout << "Binary subsequence: " <<
        std::endl;
11     MIO::inputBS<sizeof(n)*8>(bits, iqp.in, iqp.eout,
        iqp.printErr);
12     return {n, bits.to_ullong()};
13 }
14 }
15
16
17 void MDialog::ioXY(Ioeqpp &iqp) {
18     auto [x, y] = iXY(iqp);
19     std::bitset<sizeof(x)*8> bb(x);
20     iqp.out << bb << "\n";
21     bb = y;
22     iqp.out << bb << std::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     iqp.out << (f ? "There is the subsequence" : "There isn't the
        subsequence") << std::endl;
31 }

```

## 7.4 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

```

## 7.5 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

```

## 8 Лабораторная работа 6

### 8.1 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.

### 8.2 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;

### 8.3 Листинги исходных файлов программы

Листинг 34: main.cpp

---

```

1  #include "myfuncs.h"
2  #include "version.h"
3
4  int main() {
5      std::cout << "Currently you use the " << LABS_VERSION << "
6          version of the Labs.\n" << "Git commit hash: " <<
7          GIT_COMMIT_HASH << "." << std::endl;
8      MDialog dialog{};
9      bool DS = true;
10     while (DS) {
11         if (!dialog.step()) break;
12         std::cout << "Repeat? 1 - Yes, 0 - No: " << std::endl;
13         MIO::inputBool(DS, std::cin, std::cerr);
14     }
15     return 0;
16 }
```

---

Листинг 35: myfuncs.h

```

1 #ifndef LAB6_MYFUNCS_H
2 #define LAB6_MYFUNCS_H
3
4 #include <iostream>
5 #include <cmath>
6 #include <fstream>
7 #include "dialog.h"
8
9 struct MDialog : public Dialog {
10     void ioXY(Ioeqpp &iqp) override;
11 };
12
13 #endif//LAB6_MYFUNCS_H

```

Листинг 36: myfuncs.cpp

```

1 #include "myfuncs.h"
2 #include "MLL.h"
3
4 namespace {
5     void cntD(MLL *mll, Dialog::Ioeqpp &iqp) {
6         if (iqp.printQst) iqp.qout << "Write string, which ends by
7             \'$\' sign:" << std::endl;
8         std::string str;
9         getline(iqp.in, str, '$');
10        iqp.in.ignore();
11        std::vector<Dialog::FuncWithDesc<uint64_t(MLL *,
12            std::string &)>> options = {
13            {"same full name", [](MLL *mll, std::string &str)
14                { return mll->cnt(str, 0); }},
15            {"same job", [](MLL *mll, std::string &str)
16                { return mll->cnt(str, 1); }},
17            {"same place", [](MLL *mll, std::string &str)
18                { return mll->cnt(str, 2); }},
19        };
20        iqp.out << options[Dialog::optionD<uint64_t(MLL *,
21            std::string &)>(options, iqp)].func(mll, str) <<
22            std::endl;
23    }
24 }
25
26 void MDialog::ioXY(Ioeqpp &iqp) {
27     auto *mll = new MLL;
28     std::vector<Dialog::FuncWithDesc<bool(MLL *, Dialog::Ioeqpp
29         &)>> options = {
30         {"print elements", [](MLL *mll, Dialog::Ioeqpp
31             &iqp) {

```

```

23         mll->print(iqp.out);
24         iqp.qout << std::endl;
25         return true;
26     }},
27     {"print table",          [](MLL *mll, Dialog::Ioeqpp
        &iqp) {
28         mll->printT(iqp.out);
29         return true;
30     }},
31     {"count coincidences", [](MLL *mll, Dialog::Ioeqpp
        &iqp) {
32         cntD(mll, iqp);
33         return true;
34     }},
35     {"reset",                [](MLL *mll, Dialog::Ioeqpp
        &iqp) {
36         mll->clean();
37         return true;
38     }},
39     {"read file",            [](MLL *mll, Dialog::Ioeqpp
        &iqp) {
40         auto fin = Dialog::fileD<std::ifstream>(iqp);
41         while (!fin.eof()) fin >> *mll;
42         fin.close();
43         return true;
44     }},
45     {"write to file",        [](MLL *mll, Dialog::Ioeqpp
        &iqp) {
46         auto fout = Dialog::fileD<std::ofstream>(iqp);
47         fout << *mll;
48         fout.close();
49         return true;
50     }},
51     {"exit",                 [](MLL *mll, Dialog::Ioeqpp
        &iqp) { return false; }}
52 };
53
54 bool f = true;
55 while (f) f = options[Dialog::optionD<bool>(MLL *,
        Dialog::Ioeqpp &)>(options, iqp)].func(mll, iqp);
56
57 delete mll;
58 }

```

Листинг 37: myfuncs.h

```

1 #ifndef LAB6_MLL_H
2 #define LAB6_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 //LAB6_MLL_H

```

---

Листинг 38: myfuncs.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
92             != mll.last_ ? "$" : "");
93     return out;
94 }

```

## 8.4 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 Choose an option:
6 1 - print elements;
7 2 - print table;
8 3 - count coincidences;
9 4 - reset;
10 5 - read file;
11 6 - write to file;
12 7 - exit;
13 5
14 Write name of file (in.txt as default):
15
16 Choose an option:
17 1 - print elements;
18 2 - print table;
19 3 - count coincidences;
20 4 - reset;
21 5 - read file;
22 6 - write to file;
23 7 - exit;
24 1
25 {"Full name": "NDD", "Job": "Student", "Place": "Kazan"},
    {"Full name": "TPI", "Job": "Lecturer", "Place": "Kazan"},
    {"Full name": "GSI", "Job": "Tutor", "Place": "Kazan"},
26 Choose an option:
27 1 - print elements;
28 2 - print table;
29 3 - count coincidences;
30 4 - reset;
31 5 - read file;
32 6 - write to file;
33 7 - exit;
34 2
35 Full name          Job
    Place
36 NDD                Student
                        Kazan
37 TPI                Lecturer
                        Kazan
38 GSI                Tutor
                        Kazan
39
40 Choose an option:
41 1 - print elements;
42 2 - print table;
43 3 - count coincidences;
44 4 - reset;

```

```

45 5 - read file;
46 6 - write to file;
47 7 - exit;
48 7
49 Repeat? 1 - Yes, 0 - No:
50 0

```

---

in.txt

---

```

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

```

---

2. Файл → файл

---

in1.txt

---

```

1 5
2 in.txt
3 1
4 7

```

---



---

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
5 Write name of file (out.txt as default):
6
7 Write name of file (in.txt as default):
8 in1.txt

```

---



---

in.txt

---

```

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

```

---



---

out.txt

---

```

1 {"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"},

```

---