МИНОБРНАУКИ РОССИИ
Санкт-Петербургский государственный
электротехнический университет «ЛЭТИ»
им. В.И. Ульянова (Ленина)
Кафедра САПР

# ЛАБОРАТОРНАЯ РАБОТА №2

## по дисциплине «Программирование»

## Тема: Функции, обеспечивающие обработку массива

Студенты гр. 9892 _____ Лескин К.А.

_____ Миллер В.В.

Преподаватель _____ Кузьмин С.А.

Санкт-Петербург
2020

## Цель работы

Получение практических навыков разработки алгоритмов для обработки массивов.

## Формулировка задания

Разработать программу, обрабатывающую элементы многомерного массива в соответствии с вариантом. Размерности массива должны вводиться пользователем с файла, или же с клавиатуры. На их основе массивы должны быть созданы в динамической памяти программы с помощью указателя. Доступ к элементам массива также осуществляется с помощью указетеля. Элементы массива могут вводиться пользователем как с файла, так и с клавиатуры. Исходный и преобразованный массивы должны быть выведены на экран и записаны в выходной файл после обработки.

Индивидуальный вариант: Разработать программу для сложения двух прямоугольных матриц (двумерных массивов).

# Форматы входных и выходных файлов

## Входной файл

Входной файл должен быть представлен в виде текстового файла, содержащего последовательность чисел. Разширение файла может быть любым. Последовательность должна быть записана в следующем виде: Первым членом последовательности должна быть ширина матрицы $w$ – количество столбцов, далее должна идти высота матрицы $h$ – количество строк матрицы. После должны идти элементы матрицы. Их количество определяется по формуле $w * h$. Между элементами могут быть любые разделительные символы (пробел, знак переноса строки, знак табуляции). Рекоментуется оформлять элементы матрицы в виде сетки $w * h$ для повышения читаемости файла.

## Выходной файл

Выходной файл записывается в следующем формате:

- Матрица 1
- Знак '+'
- Матрица 2
- Знак '='
- Матрица 3 (сумма матриц 1 и 2)

# Описание структур данных

Матрица в программе представлена в виде структуры. Структура состоит из:

- Ширина матрицы (по умолчанию 0)
- Высота матрицы (по умолчанию 0)
- Указатель массив указателей на число (по умолчанию *nullptr*)

Сама матрица является массивом указателей на числа. Доступ к массиву осуществляется через указатель на него (хранится в стуктуре). Память под числа и массив указателей на них выделяется динамически.

# Описание пользовательских функций

## GenElement

```
1 int GenElement(GenType gen_type, int width, int i, int j)
    ;
```

Функция для генерации элементов матрицы.

Принимает на вход тип генерации, ширину матрицы, строку и столбец элемента.

Возвращает число – элемент матрицы.

Алгоритм работы функции:

В зависимости от типа генерации

- Если GenType::Zero вернуть 0
- Если GenType::One вернуть 1
- Если GenType::Random вернуть случайное число от 0 до 9
- Если GenType::Fill0 вернуть $i * width + j$
- Если GenType::Fill1 вернуть $i * width + j + 1$
- Если GenType::Input вернуть InputElement(i, j)
- В остальных случаях вернуть -1

## Input

```
1 int Input(const std::string& message = "Input: ", int l =
    _min_int, int r = _max_int);
```

Функция для ввода числа. Содержит в себе логику по обработке исключений и ограничения диапазона допустимых чисел.

Принимает на вход сообщение, приглашающее к вводу, левую и правую границы диапазона ввода.

Возвращает введённое число

Алгоритм работы функции:

- Задаём вводимый элемент
- Создаём флаг, сигнализирующий конец цикла
- Пока флаг поднят:

  Вывести сообщение

  Ввести элемент

Если ошибка ввода или введённый элемент не входит в допустимый диапазон

Выводим "Wrong input!"

Иначе

Опускаем флаг цикла

Чистим поток cin от мусора

- Возвращаем ведённый элемент

## InputElement

```
1| int InputElement(int i, int j);
```

Функция для ввода элемента матрицы. Является обёрткой над функцией Input().

Принимает на вход строку и столбец вводимого элемента.

Возвращает вводимый пользователем элемент

Алгоритм работы функции:

- Создаём сообщение "Input element [i][j]: подставляя вместо i и j входные параметры
- Возвращаем Input()

## InputDimensions

```
1| Matrix2 InputDimensions();
```

Функция для ввода размеров матрицы. Является обёрткой над функцией Input().

Не принимает на вход параметров

Возвращает матрицу с определёнными размерами.

Алгоритм работы функции:

- Создаём результирующую матрицу
- Записываем в ширину результат ввода
- Записываем в высоту результат ввода
- Возвращаем результирующую матрицу

## CreateMatrix

```
1 Matrix2 CreateMatrix(int w, int h, GenType gen_type =
    GenType::Zero);
```

Функция для создания матрицы.
Принимает на вход ширину, высоту и тип генерации матрицы.
Возвращает матрицу.
Алгоритм работы функции:

- Создаём результирующую матрицу
- Записываем в ширину параметр ширины
- Записываем в высоту параметр высоты
- Выделяем память под строки (размером *height*)
- Цикл по i от 0 до *height*
  - Выделяем память под элементы строки (размером *width*)
  - Цикл по j от 0 до *width*
    - Генерируем элемент с помощью GenElement
    - Записываем элемент в матрицу
- Возвращаем результирующую матрицу

## LoadMatrix

```
1 Matrix2 LoadMatrix(const std::string& file_name);
```

Функция для загрузки матрицы из файла.
Принимает на вход имя файла (путь к файлу).
Возвращает матрицу.
Алгоритм работы функции:

- Создаём результирующую матрицу
- Пытаемся открыть файл
- Если файл не открыт, возвращаем пустую матрицу
- Считываем из файла параметры ширины и высоты
- Выделяем память под строки (размером *height*)
- Цикл по i от 0 до *height*

  - Выделяем память под элементы строки (размером *width*)
  - Цикл по j от 0 до *width*
    * Считываем элемент из файла.
    * Записываем элемент в матрицу

- Возвращаем результирующую матрицу

### String

```
1| std::string String(const Matrix2 &m);
```

Функция для получения форматированного строкового представления матрицы.

Принимает на вход константную ссылку на матрицу

Возвращает форматированное строковое представление матрицы

Алгоритм работы функции:

- Создаём строковый поток
- Цикл по i от 0 до *height* матрицы
  - Цикл по j от 0 до *width* матрицы
    * Записываем в поток смещённый вправо на 5 символов элемент матрицы
  - Записываем в поток символ конца строки
- Возвращаем строковое представление потока

### Print

```
1| void Print(const Matrix2 &m);
```

Функция для печати матрицы в стандартный поток вывода.

Принимает на вход константную ссылку на матрицу

Не возвращает ничего.

Алгоритм работы функции:

- Вывести в stdout результат String()

### Sum

```
1| Matrix2 Sum(Matrix2 a, Matrix2 b);
```

Функция для суммирования двух матриц.
Принимает на вход две константные ссылки на матрицы.
Возвращает матрицу – сумму двух переданных матриц.
Алгоритм работы функции:

- Если размеры матриц не совпадают, выкидываем исключение.
- Создаём результирующую матрицу
- Цикл по i от 0 до *height*
  - Цикл по j от 0 до *width*
    - Записываем в результирующую матрицу сумму элементов переданных матриц с соответствующими координатами.
- Возвращаем результирующую матрицу

## Delete

```
1| void Delete(Matrix2 m);
```

Функция для удаления матрицы из динамической памяти.
Принимает на вход ссылку на матрицу.
Не возвращает ничего.
Алгоритм работы функции:

- Цикл по i от 0 до *height*
  - Освобождаем память $i$ -ой строки.
- Освобождаем массив указателей

## PrintMainMenu

```
1| void PrintMainMenu();
```

Функция для вывода пользовательского меню.
Не принимает на вход параметров
Не возвращает ничего.
Алгоритм работы функции:

- Вывести меню

### ChooseGenType

```
1| GenType ChooseGenType();
```

Функция для пользовательского выбора типа генерации матрицы.
Не принимает на вход параметров.
Возвращает выбранный пользователем тип генерации.
Алгоритм работы функции:

- Выводим меню
- Считываем пользовательский ввод (в диапазоне 1–7)
- Возвращаем считанное число, приведённое к типу GenType

### TryLoad

```
1| bool TryLoad(Matrix2 &m);
```

Функция для обработки загрузки матрицы. Включает в себя обработку исключений.
Принимает на вход ссылку на матрицу.
Возвращает bool – успешность загрузки.
Алгоритм работы функции:

- Считываем имя файла
- Загружаем матрицу
- Если размеры матриц не совпадают или матрица пустая
  - Выводим сообщение об ошибке
  - Возвращаем false
- Присваиваем переданной матрице загруженную
- Возвращаем true

### Menu

```
1| void Menu(Matrix2 &m);
```

Функция-меню для взаимодействия с пользователем и создания матрицы.
Принимает на вход ссылку на матрицу.
Не возвращает ничего.
Алгоритм работы функции:

- Создаём флаг, сигнализирующий конец цикла

- Пока флаг поднят:
  - Считываем тип генерации
  - Если тип генерации НЕ файл

    Создаём матрицу на основе типа генерации

    Опускаем флаг цикла
  - Иначе

    Флаг цикла = Успешности загрузки матрицы (TryLoad)
- Выводим полученную матрицу на экран

## Алгоритм программы

- Выводим общую инф-ю о программе
- Считываем размеры ссумируемых матриц и создаём две матрицы с соответствующими размерами
- Вызываем меню для первой матрицы
- Вызываем меню для второй матрицы
- Суммируем матрицы и записываем результат в третью матрицу
- Создаём форматированную строку в виде сложения матриц "столбиком"
- Выводим форматированную строку в консоль
- Выводим форматированную строку в файл output.txt
- Удаляем созданные матрицы

## Тестирование программы

Тесты с описанием действий. Обязательно картинки с результатами

## Выводы

Вывод

# Приложение А. Листинг программного кода

## main.cpp

```
#include <iostream>

#include "matrix.h"
#include "menu.h"

int main()
{
    std::cout << "Matrix addition program. (c) 2020\n"
                 "Enter the dimensions of the summed
                    matrices: \n";

    Matrix2 a = InputDimensions(), b = a;

    std::cout << "\nMatrix 1: \n\n";
    Menu(a);
    std::cout << "\nMatrix 2: \n\n";
    Menu(b);

    Matrix2 c = Sum(a, b);
    std::cout << "
       ==========================================\n"
                 "Sum:\n";

    std::stringstream file_content;
    std::string plus = "+\n", equals = "=\n";

    file_content << String(a)
                 << std::right << std::setw(4) << plus
                 << String(b)
                 << std::right << std::setw(4) << equals
                 << String(c);

    std::cout << file_content.str();

    std::ofstream fout("output.txt");
    fout << file_content.str();

    Delete(a);
```

```
37|     Delete(b);
38|     Delete(c);
39|     return 0;
40| }
```

## input.h

```
1  #ifndef LAB2_INPUT_H
2  #define LAB2_INPUT_H
3
4  #define _min_int std::numeric_limits<int>::min()
5  #define _max_int std::numeric_limits<int>::max()
6
7  #include <iostream>
8  #include <sstream>
9
10 int Input(const std::string& message = "Input: ", int l =
       _min_int, int r = _max_int);
11 int InputElement(int i, int j);
12
13 #endif //LAB2_INPUT_H
```

### input.cpp

```
1  #include "input.h"
2
3  int Input(const std::string& message, int l, int r)
4  {
5      int element;
6      bool input = true;
7      while (input)
8      {
9          std::cout << message;
10         std::cin >> element;
11         if (std::cin.fail() || (element < l || element > r)
               )
12             std::cout << "Wrong input!\n";
13         else
14             input = false;
15         std::cin.clear();
16         std::cin.ignore(std::numeric_limits<std::streamsize
               >::max(), '\n');
17     }
18     return element;
19 }
20
21 int InputElement(int i, int j)
22 {
23     std::stringstream ss;
24     ss << "Input element ["<<i<<"]["<<j<<"]: ";
25     return Input(ss.str());
26 }
```

### gen.h

```
1  #ifndef LAB2_GEN_H
2  #define LAB2_GEN_H
3
4  #include <random>
5
6  #include "input.h"
7
8  enum class GenType
9  {
10     Zero = 1,
11     One = 2,
12     Random = 3,
13     Fill0 = 4,
14     Fill1 = 5,
15     Input = 6,
16     File = 7,
17     None,
18 };
19
20 int GenElement(GenType gen_type, int width, int i, int j);
21
22 #endif //LAB2_GEN_H
```

### gen.cpp

```cpp
#include "gen.h"

int GenElement(GenType gen_type, int width, int i, int j)
{
    switch (gen_type)
    {
        case GenType::Zero:
            return 0;
        case GenType::One:
            return 1;
        case GenType::Random:
        {
            std::random_device rd;
            std::mt19937 Rand(rd());
            return std::abs(static_cast<int>(Rand())) % 10;
        }
        case GenType::Fill0:
            return i*width+j;
        case GenType::Fill1:
            return i*width+j+1;
        case GenType::Input:
            return InputElement(i, j);
        case GenType::File:
        case GenType::None:
        default:
            return -1;
    }
}
```

### matrix.h

```
1  #ifndef LAB2_MATRIX_H
2  #define LAB2_MATRIX_H
3
4  #include <iostream>
5  #include <malloc.h>
6  #include <iomanip>
7  #include <fstream>
8  #include <random>
9
10 #include "gen.h"
11 #include "input.h"
12
13 struct Matrix2
14 {
15     int width = 0;
16     int height = 0;
17     int **matrix = nullptr;
18 };
19
20 Matrix2 InputDimensions();
21
22 Matrix2 CreateMatrix(int w, int h, GenType gen_type =
       GenType::Zero);
23
24 Matrix2 LoadMatrix(const std::string& file_name);
25
26 std::string String(const Matrix2 &m);
27
28 void Print(const Matrix2 &m);
29
30 Matrix2 Sum(const Matrix2 &a, const Matrix2 &b);
31
32 void Delete(Matrix2 &m);
33
34 #endif //LAB2_MATRIX_H
```

### matrix.cpp

```
1  #include "matrix.h"
2
3  Matrix2 InputDimensions()
4  {
5      Matrix2 result{};
6      result.width = Input("Width: ", 1);
7      result.height = Input("Height: ", 1);
8      return result;
9  }
10
11 Matrix2 CreateMatrix(int w, int h, GenType gen_type)
12 {
13     Matrix2 result = Matrix2();
14     result.width = w;
15     result.height = h;
16     result.matrix = (int**)calloc(result.height, sizeof(int
           *));
17     for(int i = 0; i < result.height; ++i)
18     {
19         result.matrix[i] = (int*)calloc(result.width,
               sizeof(int));
20         for(int j = 0; j < result.width; ++j)
21         {
22             int new_element = GenElement(gen_type, result.
                   width, i, j);
23             result.matrix[i][j] = new_element;
24         }
25     }
26     return result;
27 }
28
29 Matrix2 LoadMatrix(const std::string& file_name)
30 {
31     Matrix2 result = Matrix2();
32
33     std::ifstream fin(file_name);
34     if(!fin)
35     {
36         return result;
37     }
38     fin >> result.width >> result.height;
39
```

```
40      result.matrix = (int**)calloc(result.height, sizeof(int
            *));
41      for(int i = 0; i < result.height; ++i)
42      {
43          result.matrix[i] = (int*)calloc(result.width,
                sizeof(int));
44          for(int j = 0; j < result.width; ++j)
45          {
46              int new_element;
47              fin >> new_element;
48              result.matrix[i][j] = new_element;
49          }
50      }
51
52      return result;
53  }
54
55  std::string String(const Matrix2 &m)
56  {
57      std::stringstream ss;
58      for(int i = 0; i < m.height; ++i)
59      {
60          for(int j = 0; j < m.width; ++j)
61          {
62              ss << std::right << std::setw(5) << m.matrix[i
                    ][j] << " ";
63          }
64          ss << "\n";
65      }
66      return ss.str();
67  }
68
69  void Print(const Matrix2 &m)
70  {
71      std::cout << String(m);
72  }
73
74  Matrix2 Sum(const Matrix2 &a, const Matrix2 &b)
75  {
76      if(a.width != b.width || a.height != b.height)
77      {
78          throw std::exception();
79      }
```

```
80
81      Matrix2 result = CreateMatrix(a.width, a.height,
            GenType::Zero);
82      for(int i = 0; i < a.height; ++i)
83      {
84          for(int j = 0; j < a.width; ++j)
85          {
86              result.matrix[i][j] = a.matrix[i][j] + b.matrix
                    [i][j];
87          }
88      }
89
90      return result;
91  }
92
93  void Delete(Matrix2 &m)
94  {
95      for(int i = 0; i < m.height; ++i)
96      {
97          free(m.matrix[i]);
98      }
99      free(m.matrix);
100 }
```

### menu.h

```
1  #ifndef LAB2_MENU_H
2  #define LAB2_MENU_H
3
4  #include <iostream>
5  #include <fstream>
6
7  #include "matrix.h"
8
9  void PrintMainMenu();
10
11 GenType ChooseGenType();
12
13 bool TryLoad(Matrix2 &m);
14
15 void Menu(Matrix2 &m);
16
17 #endif //LAB2_MENU_H
```

### menu.cpp

```
1  #include "menu.h"
2
3  void PrintMainMenu()
4  {
5      std::cout << "1 - Fill matrix with 0" << "\n";
6      std::cout << "2 - Fill matrix with 1" << "\n";
7      std::cout << "3 - Fill matrix with random numbers (0-9)
           " << "\n";
8      std::cout << "4 - Fill matrix with ordinal numbers (
           begin with 0)" << "\n";
9      std::cout << "5 - Fill matrix with ordinal numbers (
           begin with 1)" << "\n";
10     std::cout << "6 - Fill matrix manually" << "\n";
11     std::cout << "7 - Load matrix form file" << "\n";
12 }
13
14 GenType ChooseGenType()
15 {
16     PrintMainMenu();
17     int option = Input("Input: ", 1, 7);
18     return static_cast<GenType>(option);
19 }
20
21 bool TryLoad(Matrix2 &m)
22 {
23     std::string file_name;
24     std::cout << "Input name of file: ";
25     std::cin >> file_name;
26
27     Matrix2 lm = LoadMatrix(file_name);
28
29     if (lm.width != m.width || lm.height != m.height || !lm
           .matrix)
30     {
31         std::cout << "Could not load matrix\n"
32                      "File does not exist or\n"
33                      "Loaded matrix has different
                           dimensions with defined.\n"
34                      "Choose other file or generate matrix
                           with the proposed options.\n\n";
35         return false;
36     }
```

```
37
38      m = lm;
39      return true;
40  }
41
42  void Menu(Matrix2 &m)
43  {
44      bool continue_ = true;
45      while(continue_)
46      {
47          GenType task = ChooseGenType();
48          if (task != GenType::File)
49          {
50              m = CreateMatrix(m.width, m.height, task);
51              continue_ = false;
52          }
53          else
54          {
55              continue_ = !TryLoad(m);
56          }
57      }
58
59      std::cout << "Resulting matrix:\n";
60      Print(m);
61  }
```