**Отчёт по лабораторной работе №5**

**Постановка задачи вариант №9**

1. Сформировать динамический одномерный массив, заполнить его случайными числами и вывести на печать.

2. Выполнить указанное в варианте задание и вывести полученный массив на печать.

3. Сформировать динамический двумерный массив, заполнить его случайными числами и вывести на печать.

4. Выполнить указанное в варианте задание и вывести полученный массив на печать.

5. Сформировать динамический двумерный массив, заполнить его случайными числами и вывести на печать.

6. Выполнить указанное в варианте задание и вывести полученный массив на печать.



**Анализ первой задачи**

|  |  |  |
| --- | --- | --- |
| Входные данные | Классы входных данных | Выходные данные |
| command – целое число (выбор пункта меню) | 1<= command <=5 | Программа переходит на следующий этап работы |
| command – целое число(выбор пункта меню) | command < 1; command > 5; | Сообщение об ошибке |
| command – целое число(выбор пункта меню) | command = «abc» | Сообщение об ошибке |
| A[i] – целое число (элемент массива) | A[i] **∈** Z | Массив успешно заполнен |
| A[i] – целое число (элемент массива) | A[i] = «abc» | Сообщение об ошибке |

1. Ввести требуемые данные, выполнить расчёты и вывести результаты в консоль.
2. В случае ввода некорректных данных, выдать сообщение об ошибке и запросить повторный ввод.

**Анализ второй задачи**

|  |  |  |
| --- | --- | --- |
| Входные данные | Классы входных данных | Выходные данные |
| command – целое число (выбор пункта меню) | 1<= command <=5 | Программа переходит на следующий этап работы |
| command – целое число(выбор пункта меню) | command < 1; command > 5; | Сообщение об ошибке |
| command – целое число(выбор пункта меню) | command = «abc» | Сообщение об ошибке |
| A[i][j] – целое число (элемент массива) | A[i][j] **∈** Z | Массив успешно заполнен |
| A[i][j] – целое число (элемент массива) | A[i][j] = «abc» | Сообщение об ошибке |

1. Ввести требуемые данные, выполнить расчёты и вывести результаты в консоль.
2. В случае ввода некорректных данных, выдать сообщение об ошибке и запросить повторный ввод.

**Анализ третьей задачи**

|  |  |  |
| --- | --- | --- |
| Входные данные | Классы входных данных | Выходные данные |
| command – целое число (выбор пункта меню) | 1<= command <=5 | Программа переходит на следующий этап работы |
| command – целое число(выбор пункта меню) | command < 1; command > 5; | Сообщение об ошибке |
| command – целое число(выбор пункта меню) | command = «abc» | Сообщение об ошибке |
| n – целое число (кол-во строк массива)  m – целое число (кол-во столбцов массива) | n <= 0  m<=0 | Сообщение об ошибке |
| n – целое число (кол-во строк массива)  m – целое число (кол-во столбцов массива) | n > 0  m > 0 | Создается массив, программа просит ввести элементы |
| A[i][j] – целое число (элемент массива) | A[i][j] **∈** Z | Массив успешно заполнен |
| A[i][j] – целое число (элемент массива) | A[i][j] = «abc» | Сообщение об ошибке |

1. Ввести требуемые данные, выполнить расчёты и вывести результаты в консоль.
2. В случае ввода некорректных данных, выдать сообщение об ошибке и запросить повторный ввод.

**Программный код первой задачи**

**Файл Program.cs**

using System;

namespace Lab5

{

class Program

{

static void Main(string [] args)

{

int command;

bool shouldClose = false;

int[] oneDimArray = null;

int[,] twoDimArray = null;

int[][] raggedArray = null;

while(!shouldClose)

{

ShowMenu();

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out command))

{

switch (command)

{

case 1:

Console.Clear();

OneDimArrayWorker.StartWorkOneDimArray(ref oneDimArray);

Console.Clear();

break;

case 2:

Console.Clear();

TwoDimArrayWorker.StartWorkTwoDimArray(ref twoDimArray);

Console.Clear();

break;

case 3:

Console.Clear();

RaggedArrayWorker.StartWorkRaggedTwoDimArray(ref raggedArray);

Console.Clear();

break;

case 4:

shouldClose = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число\n");

Console.ResetColor();

}

}

}

static void ShowMenu()

{

Console.WriteLine("========= Главное меню =========");

Console.WriteLine("1. Работа с одномерным массивом");

Console.WriteLine("2. Работа с двумерным массивом");

Console.WriteLine("3. Работа с рваным массивом");

Console.WriteLine("4. Выход");

}

}

}

**Код первой задачи**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab5

{

class OneDimArrayWorker

{

static bool isInitialized = false;

public static void StartWorkOneDimArray(ref int[] oneDimArray)

{

bool shouldExitInMainMenu = false;

int command;

while (!shouldExitInMainMenu)

{

ShowMenu();

PrintStat(ref oneDimArray);

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out command))

{

switch (command)

{

case 1:

Console.Clear();

CreateArray(ref oneDimArray);

break;

case 2:

Console.Clear();

FillArray(ref oneDimArray);

break;

case 3:

Console.Clear();

PrintArray(ref oneDimArray);

break;

case 4:

Console.Clear();

RemoveODD(ref oneDimArray);

break;

case 5:

shouldExitInMainMenu = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число\n");

Console.ResetColor();

}

}

}

static void PrintStat(ref int[] oneDimArray)

{

if (oneDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Массив[null]");

Console.ResetColor();

return;

}

else

{

Console.ForegroundColor = ConsoleColor.Green;

Console.Write($"Массив[{oneDimArray.Length}]");

Console.ResetColor();

}

if (isInitialized)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("(Заполнен)");

Console.ResetColor();

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("(Пустой)");

Console.ResetColor();

}

}

static void ShowMenu()

{

Console.WriteLine("==== Одномерный массив ====");

Console.WriteLine("1. Создать массив");

Console.WriteLine("2. Заполнить массив");

Console.WriteLine("3. Напечатать массив");

Console.WriteLine("4. Удалить все нечётные элементы");

Console.WriteLine("5. Выйти в главное меню");

}

static void CreateArray(ref int[] oneDimArray)

{

Console.WriteLine("==== Создание одномерного массива ====\n");

bool successInput = false;

int size = 1;

if (oneDimArray != null)

{

Console.Write("Массив уже создан. Удалить старый и создать новый? (+/-) ");

if (Console.ReadLine() != "+")

{

Console.Clear();

return;

}

}

while (!successInput)

{

Console.Write("Введите размер массива: ");

if (int.TryParse(Console.ReadLine(), out size))

{

if (size > 0)

{

successInput = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: неверный размер массива\n");

Console.ResetColor();

}

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число\n");

Console.ResetColor();

}

}

try

{

oneDimArray = new int[size];

}

catch (OutOfMemoryException)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("FATAL ERROR: OUT OF MEMORY EXCEPTION!!!\n");

Console.ResetColor();

GC.Collect();

return;

}

isInitialized = false;

Console.Clear();

}

static void FillArray(ref int[] oneDimArray)

{

Console.WriteLine("==== Заполнение одномерного массива ====");

int command;

bool shouldExit = false;

Random rnd = new Random();

if (oneDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

while(!shouldExit)

{

Console.Write("1. Задать элементы вручную");

if (oneDimArray.Length > 10)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($" ({oneDimArray.Length} элементов!)");

Console.ResetColor();

}

else Console.WriteLine();

Console.WriteLine("2. Задать элементы через ДСЧ");

Console.WriteLine("3. Назад");

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out command))

{

switch (command)

{

case 1:

Console.WriteLine("\nn - Принудительное завершение ввода");

for(int i = 0; i < oneDimArray.Length; i++)

{

bool correctVal = false;

string buf;

while (!correctVal)

{

Console.Write("(" + (i + 1) + "/" + oneDimArray.Length + ") ");

buf = Console.ReadLine();

if (buf == "n")

{

Console.Clear();

return;

}

if (int.TryParse(buf, out oneDimArray[i]))

{

correctVal = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число");

Console.ResetColor();

}

}

}

isInitialized = true;

shouldExit = true;

break;

case 2:

Console.WriteLine("Массив заполняется...");

for (int i = 0; i < oneDimArray.Length; i++)

{

oneDimArray[i] = rnd.Next(1, 100);

}

shouldExit = true;

isInitialized = true;

break;

case 3:

shouldExit = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

}

Console.Clear();

}

static void PrintArray(ref int[] oneDimArray)

{

Console.WriteLine("==== Вывод элементов массива ====");

if (oneDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

if (oneDimArray.Length > 1000000)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"Массив слишком велик: {oneDimArray.Length} элементов");

Console.ResetColor();

Console.Write("Вывести массив? (+/-) ");

if (Console.ReadLine() != "+")

{

return;

}

}

if (!isInitialized)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Массив пустой\n");

Console.ResetColor();

}

Console.Write("Элементы массива: ");

for (int i = 0; i < oneDimArray.Length; i++)

{

Console.Write($"{oneDimArray[i]} ");

}

Console.WriteLine("\n");

}

static void RemoveODD(ref int[] oneDimArray)

{

Console.WriteLine("==== Удаление нечётных элементов ====");

if (oneDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

if (!isInitialized)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Массив пустой\n");

Console.ResetColor();

return;

}

int numEven = 0;

int numODD = oneDimArray.Length;

for (int i = 0; i < oneDimArray.Length; i++)

{

if (oneDimArray[i] % 2 == 0)

{

numEven++;

}

}

numODD -= numEven;

if (numEven > 0)

{

int[] newArray = new int[numEven];

int j = 0;

for (int i = 0; i < oneDimArray.Length; i++)

{

if (oneDimArray[i] % 2 == 0)

{

newArray[j] = oneDimArray[i];

j++;

}

}

oneDimArray = newArray;

}

else

{

oneDimArray = null;

isInitialized = false;

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"Удалено {numODD} нечётных элементов\n");

Console.ResetColor();

}

}

}

**Код второй задачи**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab5

{

class TwoDimArrayWorker

{

static bool isInitialized = false;

public static void StartWorkTwoDimArray(ref int[,] twoDimArray)

{

bool shouldExitInMainMenu = false;

while (!shouldExitInMainMenu)

{

ShowMenu();

PrintStat(ref twoDimArray);

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out int command))

{

switch (command)

{

case 1:

Console.Clear();

CreateArray(ref twoDimArray);

break;

case 2:

Console.Clear();

ShowFillArrayMenu(ref twoDimArray);

break;

case 3:

Console.Clear();

PrintArray(ref twoDimArray);

break;

case 4:

Console.Clear();

AddRows(ref twoDimArray);

break;

case 5:

shouldExitInMainMenu = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else

{

PrintParseErrorMes();

}

}

}

static void PrintParseErrorMes()

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число\n");

Console.ResetColor();

}

static void PrintStat(ref int[,] twoDimArray)

{

if (twoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Массив[null, null]");

Console.ResetColor();

return;

}

else

{

Console.ForegroundColor = ConsoleColor.Green;

Console.Write($"Массив[{twoDimArray.GetLength(0)}, {twoDimArray.GetLength(1)}]");

Console.ResetColor();

}

if (isInitialized)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("(Заполнен)");

Console.ResetColor();

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("(Пустой)");

Console.ResetColor();

}

}

static void ShowMenu()

{

Console.WriteLine("==== Двумерный массив ====");

Console.WriteLine("1. Создать массив");

Console.WriteLine("2. Заполнить массив");

Console.WriteLine("3. Напечатать массив");

Console.WriteLine("4. Добавить строки в начало массива");

Console.WriteLine("5. Выйти в главное меню");

}

static void CreateArray(ref int[,] twoDimArray)

{

Console.WriteLine("==== Создание двумерного массива ====\n");

bool successInputRows = false;

bool successInputColumns = false;

int rows = 0;

int columns = 0; ;

if (twoDimArray != null)

{

Console.Write("Массив уже создан. Удалить старый и создать новый? (+/-) ");

if (Console.ReadLine() != "+")

{

Console.Clear();

return;

}

}

while (!successInputRows)

{

Console.Write("Введите количество строк массива: ");

if (int.TryParse(Console.ReadLine(), out rows))

{

if (rows > 0)

{

successInputRows = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: неверное количество строк массива\n");

Console.ResetColor();

}

}

else

{

PrintParseErrorMes();

}

}

while (!successInputColumns)

{

Console.Write("Введите количество столбцов массива: ");

if (int.TryParse(Console.ReadLine(), out columns))

{

if (columns > 0)

{

successInputColumns = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: неверное количество столбцов массива\n");

Console.ResetColor();

}

}

else

{

PrintParseErrorMes();

}

}

try

{

twoDimArray = new int[rows, columns];

}

catch (OutOfMemoryException)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("FATAL ERROR: OUT OF MEMORY EXCEPTION!!!\n");

Console.ResetColor();

twoDimArray = null;

isInitialized = false;

GC.Collect();

return;

}

isInitialized = false;

Console.Clear();

}

private static void ShowFillArrayMenu(ref int[,] twoDimArray, int rows = 0)

{

Console.WriteLine("==== Заполнение двумерного массива ====");

bool shouldExit = false;

if (twoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

while (!shouldExit)

{

Console.Write("1. Задать элементы вручную");

if (twoDimArray.Length > 10 && rows == 0)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($" ({twoDimArray.Length} элементов!)");

Console.ResetColor();

}

else if (rows != 0 && rows \* twoDimArray.GetLength(1) > 10)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($" ({rows \* twoDimArray.GetLength(1)} элементов!)");

Console.ResetColor();

}

else Console.WriteLine();

Console.WriteLine("2. Задать элементы через ДСЧ");

Console.WriteLine("3. Назад");

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out int command))

{

switch (command)

{

case 1:

ManuallyFillingArray(ref twoDimArray, rows);

shouldExit = true;

break;

case 2:

RndFillArray(ref twoDimArray, rows);

shouldExit = true;

isInitialized = true;

break;

case 3:

shouldExit = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else PrintParseErrorMes();

}

Console.Clear();

}

private static void ManuallyFillingArray(ref int[,] twoDimArray, int rows = 0)

{

Console.WriteLine("\nn - Принудительное завершение ввода");

if (rows == 0)

{

for (int i = 0; i < twoDimArray.GetLength(0); i++)

{

for (int j = 0; j < twoDimArray.GetLength(1); j++)

{

bool correctVal = false;

while (!correctVal)

{

Console.Write($"[{i + 1}/{twoDimArray.GetLength(0)}, {j + 1}/{twoDimArray.GetLength(1)}] ");

string buf = Console.ReadLine();

if (buf == "n")

{

Console.Clear();

return;

}

if (int.TryParse(buf, out twoDimArray[i, j]))

{

correctVal = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число");

Console.ResetColor();

}

}

}

}

}

else

{

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < twoDimArray.GetLength(1); j++)

{

bool correctVal = false;

while (!correctVal)

{

Console.Write($"[{i + 1}/{rows}, {j + 1}/{twoDimArray.GetLength(1)}] ");

string buf = Console.ReadLine();

if (buf == "n")

{

Console.Clear();

return;

}

if (int.TryParse(buf, out twoDimArray[i, j]))

{

correctVal = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число");

Console.ResetColor();

}

}

}

}

}

isInitialized = true;

}

private static void RndFillArray(ref int[,] twoDimArray, int rows = 0)

{

Console.WriteLine("Массив заполняется...");

Random rnd = new Random();

if (rows == 0)

{

for (int i = 0; i < twoDimArray.GetLength(0); i++)

{

for (int j = 0; j < twoDimArray.GetLength(1); j++)

{

twoDimArray[i, j] = rnd.Next(1, 100);

}

}

}

else

{

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < twoDimArray.GetLength(1); j++)

{

twoDimArray[i, j] = rnd.Next(1, 100);

}

}

}

}

static void PrintArray(ref int[,] twoDimArray)

{

Console.WriteLine("==== Вывод элементов массива ====");

if (twoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

if (twoDimArray.Length > 1000000)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"Массив слишком велик: {twoDimArray.Length} элементов");

Console.ResetColor();

Console.Write("Вывести массив? (+/-) ");

if (Console.ReadLine() != "+")

{

return;

}

}

Console.Write("Элементы массива:\n");

for (int i = 0; i < twoDimArray.GetLength(0); i++)

{

for (int j = 0; j < twoDimArray.GetLength(1); j++)

{

Console.Write($"{twoDimArray[i, j]}\t");

}

Console.WriteLine();

}

Console.WriteLine("\n");

}

static void AddRows(ref int[,] twoDimArray)

{

Console.WriteLine("==== Добавление строк в начало массива ====");

if (twoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

int numNewRows = 0;

bool correctVal = false;

while (!correctVal)

{

Console.Write("Введите количество новых строк: ");

if (int.TryParse(Console.ReadLine(), out numNewRows))

{

if (numNewRows > 0)

{

correctVal = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: неверное количество строк\n");

Console.ResetColor();

}

}

else PrintParseErrorMes();

}

int[,] newArray;

try

{

newArray= new int[numNewRows + twoDimArray.GetLength(0), twoDimArray.GetLength(1)];

}

catch (OutOfMemoryException)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("FATAL ERROR: OUT OF MEMORY EXCEPTION!!!\n");

Console.ResetColor();

GC.Collect();

return;

}

//int[,] buf = new int[numNewRows, twoDimArray.GetLength(1)];

Console.Write("Заполнить новые строки? (+/-) ");

if (Console.ReadLine() == "+")

{

ShowFillArrayMenu(ref newArray, numNewRows);

}

else isInitialized = false;

Console.WriteLine("Перенос старых элементов в новый массив...");

for (int i = numNewRows; i < newArray.GetLength(0); i++)

{

for (int j = 0; j < newArray.GetLength(1); j++)

{

newArray[i, j] = twoDimArray[i - numNewRows, j];

}

}

twoDimArray = newArray;

Console.Clear();

}

}

}

**Код третей задачи**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab5

{

class RaggedArrayWorker

{

static bool isInitialized = false;

public static void StartWorkRaggedTwoDimArray(ref int[][] raggedTwoDimArray)

{

bool shouldExitInMainMenu = false;

while (!shouldExitInMainMenu)

{

ShowMenu();

PrintStat(ref raggedTwoDimArray);

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out int command))

{

switch (command)

{

case 1:

Console.Clear();

CreateArray(ref raggedTwoDimArray);

break;

case 2:

Console.Clear();

ShowFillArrayMenu(ref raggedTwoDimArray);

break;

case 3:

Console.Clear();

PrintArray(ref raggedTwoDimArray);

break;

case 4:

Console.Clear();

DeleteRow(ref raggedTwoDimArray);

break;

case 5:

shouldExitInMainMenu = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else

{

PrintParseErrorMes();

}

}

}

static void PrintParseErrorMes()

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число\n");

Console.ResetColor();

}

static void PrintStat(ref int[][] raggedTwoDimArray)

{

if (raggedTwoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Массив[null][null]");

Console.ResetColor();

return;

}

else

{

Console.ForegroundColor = ConsoleColor.Green;

Console.Write($"Массив[{raggedTwoDimArray.GetLength(0)}]");

Console.ResetColor();

}

if (isInitialized)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("(Заполнен)");

Console.ResetColor();

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("(Пустой)");

Console.ResetColor();

}

}

static void ShowMenu()

{

Console.WriteLine("==== Рваный двумерный массив ====");

Console.WriteLine("1. Создать массив");

Console.WriteLine("2. Заполнить массив");

Console.WriteLine("3. Напечатать массив");

Console.WriteLine("4. Удалить строку");

Console.WriteLine("5. Выйти в главное меню");

}

static void CreateArray(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("==== Создание рваного двумерного массива ====\n");

if (raggedTwoDimArray != null)

{

Console.Write("Массив уже создан. Удалить старый и создать новый? (+/-) ");

if (Console.ReadLine() != "+")

{

Console.Clear();

return;

}

}

GetCorrectVal("Введите количество строк(1 - 22360): ", "ERROR: невеное количество строк", out int rows, 1, 22360);

try

{

raggedTwoDimArray = new int[rows][];

Console.WriteLine("1. Строки одинаковой длины");

Console.WriteLine("2. Строки случайной длины(1 - 10)");

Console.WriteLine("3. Задать длину вручную для каждой строки");

GetCorrectVal(">", "Неверный код", out int command, 1, 3);

int columns;

switch (command)

{

case 1:

GetCorrectVal("Введите длину строк(1 - 22360): ", "ERROR: Неверная длина строк", out columns, 1, 22360);

Console.WriteLine("Создание строк...");

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

raggedTwoDimArray[i] = new int[columns];

}

break;

case 2:

Console.WriteLine("Создание строк...");

Random rnd = new Random();

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

raggedTwoDimArray[i] = new int[rnd.Next(1, 10)];

}

break;

case 3:

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

GetCorrectVal($"Введите длину строки({i + 1}/{raggedTwoDimArray.GetLength(0)}): ", "ERROR: Неверная длина строки", out columns, 1);

raggedTwoDimArray[i] = new int[columns];

}

break;

}

}

catch (OutOfMemoryException)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("FATAL ERROR: OUT OF MEMORY EXCEPTION!!!\n");

Console.ResetColor();

raggedTwoDimArray = null;

isInitialized = false;

GC.Collect();

return;

}

isInitialized = false;

Console.Clear();

}

private static void ShowFillArrayMenu(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("==== Заполнение двумерного массива ====");

bool shouldExit = false;

if (raggedTwoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

while (!shouldExit)

{

Console.Write("1. Задать элементы вручную");

int size = 0;

for (int i = 0; i < raggedTwoDimArray.Length; i++)

{

size += raggedTwoDimArray[i].Length;

}

if (size > 10)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($" ({size} элементов!)");

Console.ResetColor();

}

else Console.WriteLine();

Console.WriteLine("2. Задать элементы через ДСЧ");

Console.WriteLine("3. Назад");

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out int command))

{

switch (command)

{

case 1:

ManuallyFillingArray(ref raggedTwoDimArray);

shouldExit = true;

break;

case 2:

RndFillArray(ref raggedTwoDimArray);

shouldExit = true;

isInitialized = true;

break;

case 3:

shouldExit = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else PrintParseErrorMes();

}

Console.Clear();

}

private static void GetCorrectVal(string mess, string errorMes, out int val, int min = int.MinValue, int max = int.MaxValue)

{

bool successInput = false;

val = 0;

while (!successInput)

{

Console.Write(mess);

if (int.TryParse(Console.ReadLine(), out val))

{

if (min <= val && val <= max)

{

successInput = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(errorMes + "\n");

Console.ResetColor();

}

}

else

{

PrintParseErrorMes();

}

}

}

private static void ManuallyFillingArray(ref int[][] raggedTwoDimArray, int rows = 0)

{

Console.WriteLine("\nn - Принудительное завершение ввода");

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

for (int j = 0; j < raggedTwoDimArray[i].Length; j++)

{

bool correctVal = false;

while (!correctVal)

{

Console.Write($"[{i + 1}/{raggedTwoDimArray.GetLength(0)}, {j + 1}/{raggedTwoDimArray[i].Length}] ");

string buf = Console.ReadLine();

if (buf == "n")

{

Console.Clear();

return;

}

if (int.TryParse(buf, out raggedTwoDimArray[i][j]))

{

correctVal = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число");

Console.ResetColor();

}

}

}

}

isInitialized = true;

}

private static void RndFillArray(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("Массив заполняется...");

Random rnd = new Random();

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

for (int j = 0; j < raggedTwoDimArray[i].Length; j++)

{

raggedTwoDimArray[i][j] = rnd.Next(1, 100);

}

}

}

static void PrintArray(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("==== Вывод элементов массива ====");

if (raggedTwoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

if (raggedTwoDimArray.Length > 1000000)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"Массив слишком велик: {raggedTwoDimArray.Length} элементов");

Console.ResetColor();

Console.Write("Вывести массив? (+/-) ");

if (Console.ReadLine() != "+")

{

return;

}

}

Console.Write("Элементы массива:\n");

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

for (int j = 0; j < raggedTwoDimArray[i].Length; j++)

{

Console.Write($"{raggedTwoDimArray[i][j]}\t");

}

Console.WriteLine();

}

Console.WriteLine("\n");

}

static void DeleteRow(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("==== Удаление строк из массива ====");

if (raggedTwoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

GetCorrectVal($"Введите номер строки(1 - {raggedTwoDimArray.GetLength(0)}): ", "ERROR: неверный номер строки", out int delIndex, 1, raggedTwoDimArray.GetLength(0));

int[][] newArray;

newArray = new int[raggedTwoDimArray.GetLength(0) - 1][];

int j = 0;

Console.WriteLine("Перенос старых элементов в новый массив...");

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

if (i != delIndex - 1)

{

newArray[j] = new int[raggedTwoDimArray[i].Length];

for (int p = 0; p < newArray[j].Length; p++)

{

newArray[j][p] = raggedTwoDimArray[i][p];

}

j++;

}

}

if (newArray.Length == 0)

{

raggedTwoDimArray = null;

isInitialized = false;

}

else raggedTwoDimArray = newArray;

Console.Clear();

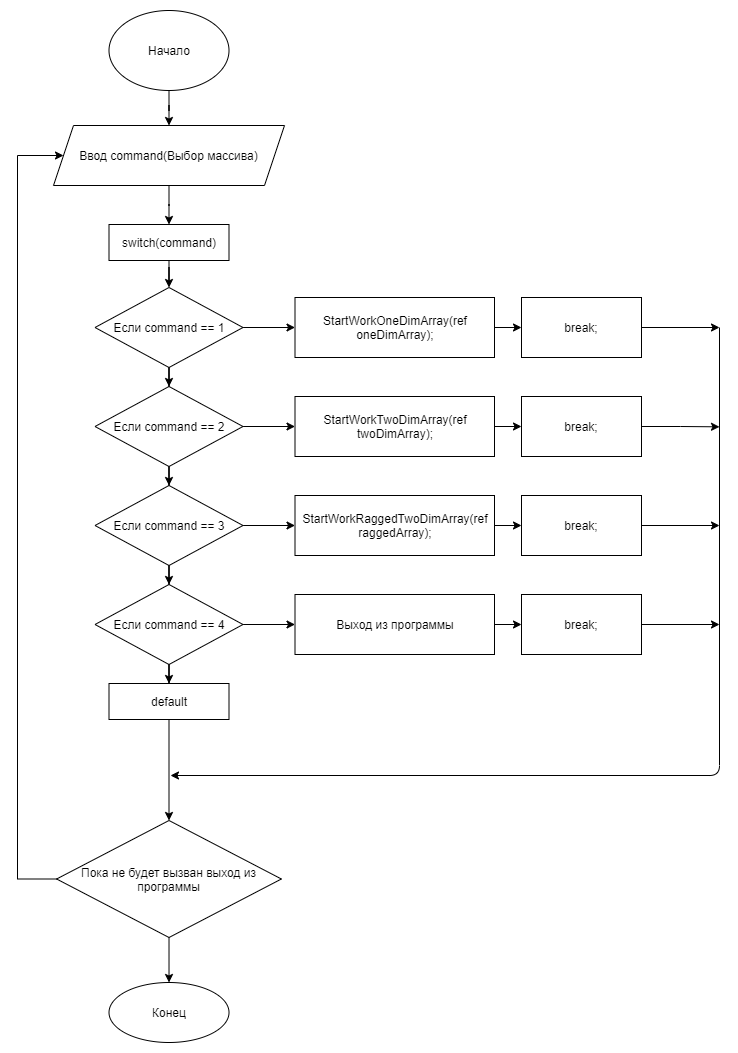
}

}

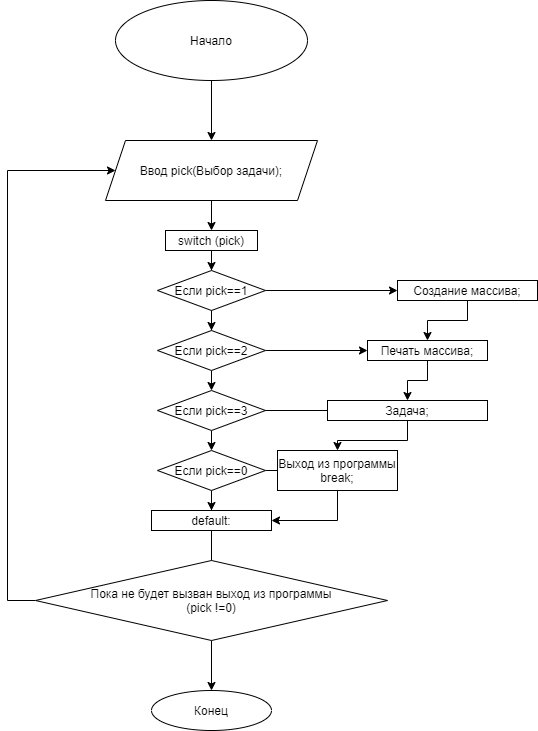
}

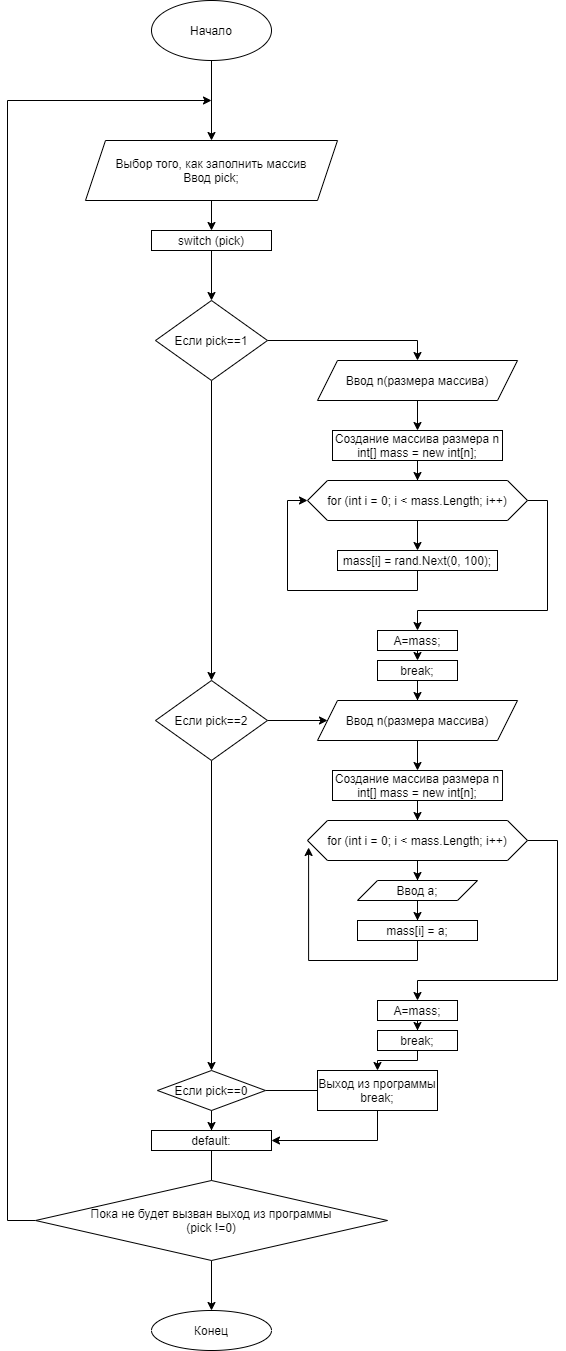
**Блок – схемы**

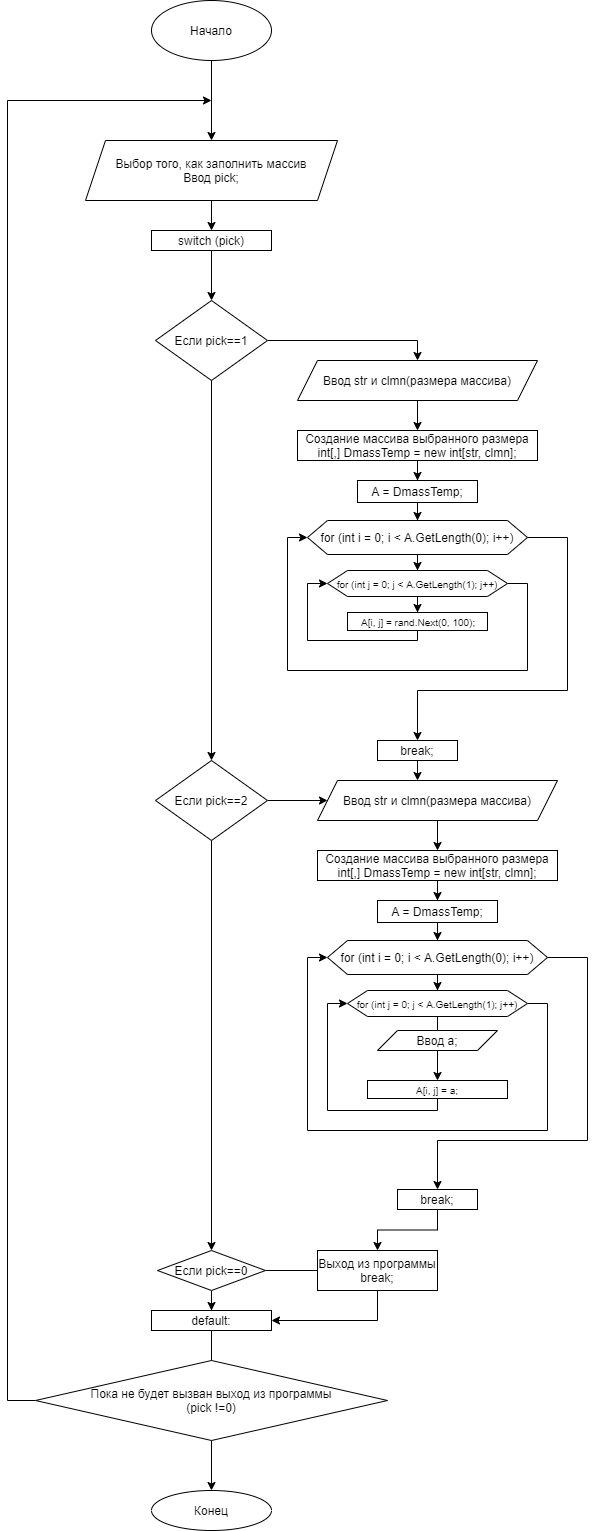
**Main()**

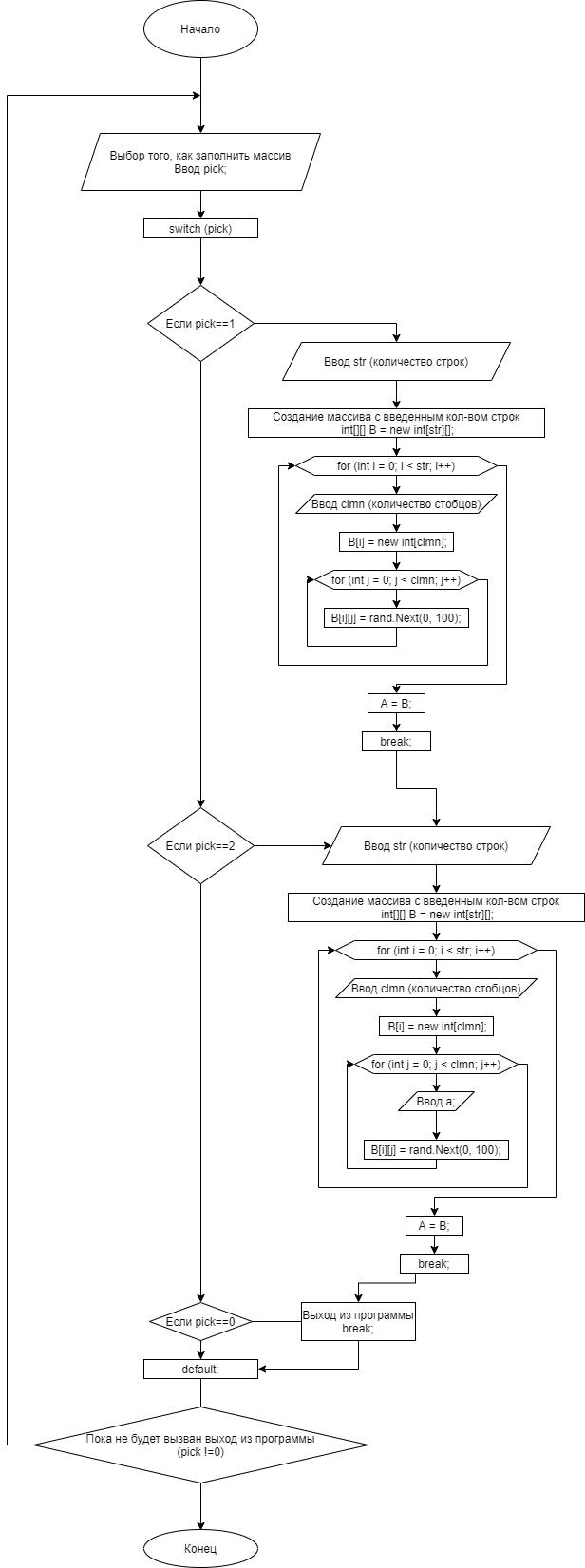


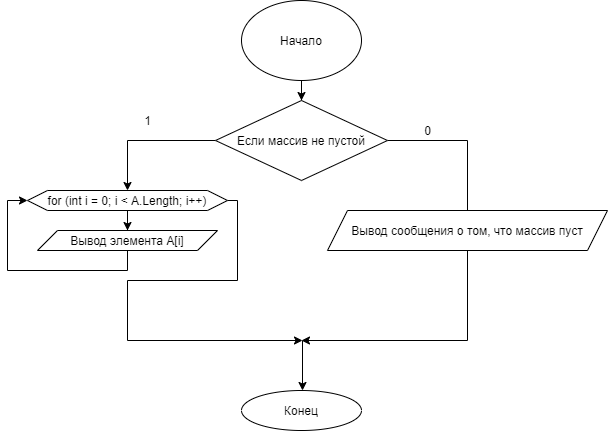
**Меню выбора функций**

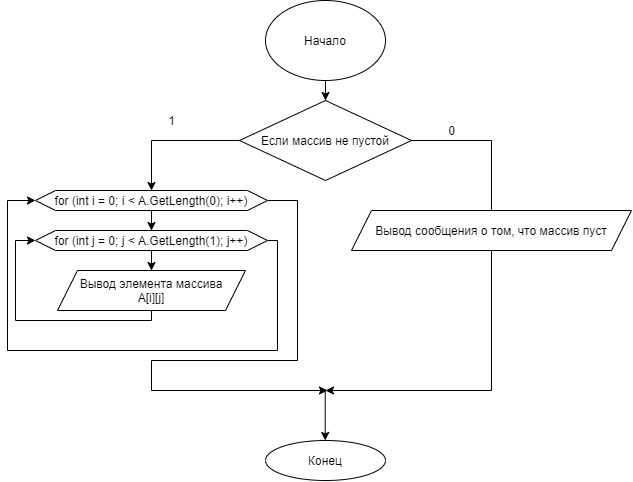
****

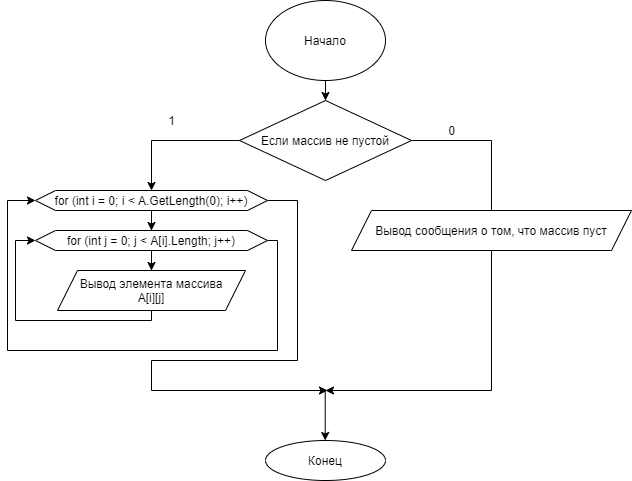
Создание одномерного массива

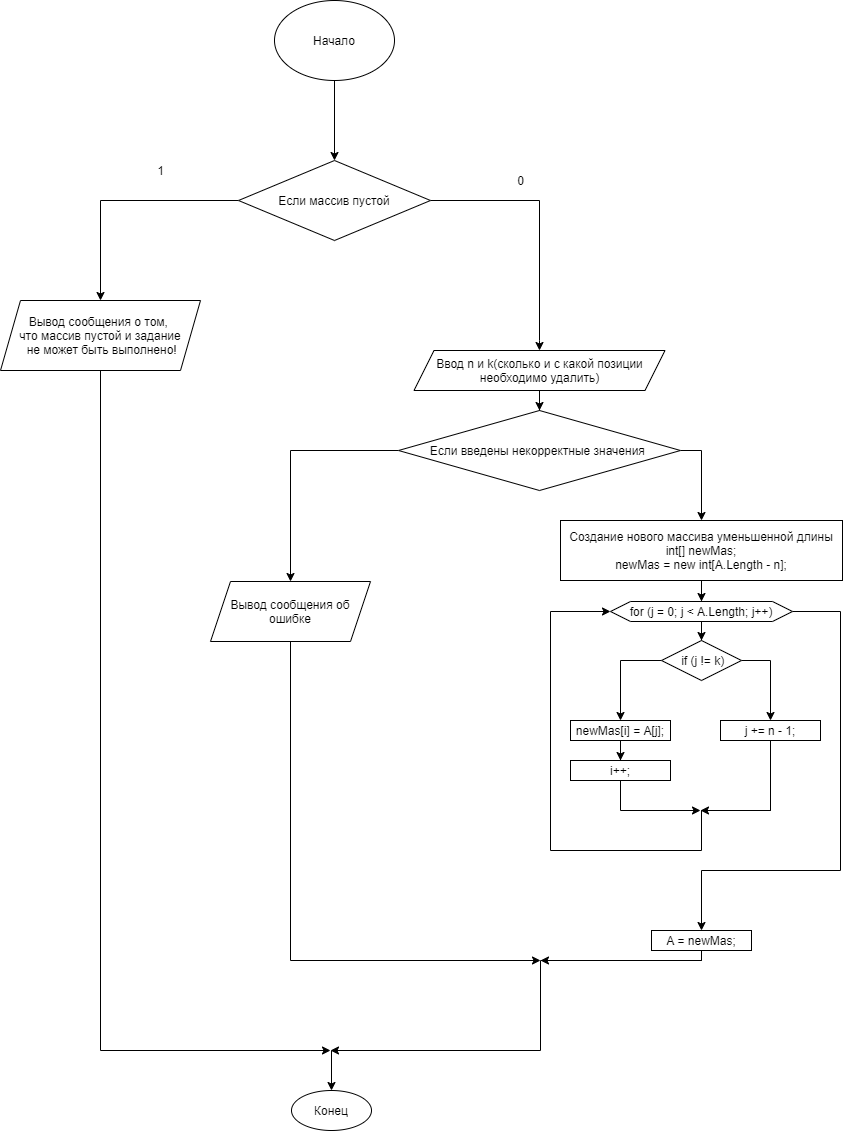
 Создание двумерного массива

 Создание рваного массива

Функции печати массивов





 Задача

Задача

