УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.3

по предмету «Основы алгоритмизации и программирования»

Вариант 18

Выполнил:

Егоров А.С.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Сортировка естественным слиянием.

**Код программы Delphi:**

Program Program3;

Uses

System.SysUtils;

Const

MIN\_SIZE: Integer = 2;

Type

TArray = Array Of Integer;

// class reader

Type

TReader = Class

Public

Function InputSize(): Integer; Virtual; Abstract;

Function InputArray(Const SIZE: Integer): TArray; Virtual; Abstract;

Procedure InvalidTypeMessage();

End;

Procedure TReader.InvalidTypeMessage();

Begin

Writeln('Invalid type! Try again.');

End;

// class consoleReader

Type

TConsoleReader = Class(TReader)

Public

Function InputSize(): Integer; Override;

Function InputArray(Const SIZE: Integer): TArray; Override;

End;

Function TConsoleReader.InputSize(): Integer;

Var

Size: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := False;

Size := 0;

Repeat

Try

Writeln('Enter size of array:');

Readln(Size);

IsCorrect := True;

Except

InvalidTypeMessage;

End;

If (Size < MIN\_SIZE) Then

Begin

Writeln('Size of array cannot be less than ', MIN\_SIZE, '.');

IsCorrect := False;

End;

Until IsCorrect;

InputSize := Size;

End;

Function TConsoleReader.InputArray(Const SIZE: Integer): TArray;

Var

IsCorrect: Boolean;

I: Integer;

Arr: TArray;

Begin

SetLength(Arr, SIZE);

IsCorrect := False;

For I := 1 To SIZE Do

Begin

IsCorrect := False;

Repeat

Try

Writeln('Enter ', I, ' element:');

Readln(Arr[I - 1]);

IsCorrect := True;

Except

InvalidTypeMessage;

End;

Until IsCorrect;

End;

InputArray := Arr;

End;

// class FileReader

Type

TFileReader = Class(TReader)

Private

FileName: String;

InFile: TextFile;

Status: Boolean;

Function IsFileTxt(): Boolean;

Function IsFileReadable(): Boolean;

Public

Function GetStatus(): Boolean;

Procedure SetFileName();

Function InputSize(): Integer; Override;

Function InputArray(Const SIZE: Integer): TArray; Override;

Function IsFileGood(): Boolean;

End;

Function TFileReader.GetStatus: Boolean;

Begin

GetStatus := Status;

End;

Function TFileReader.IsFileTxt(): Boolean;

Var

FileType: String;

Begin

FileType := FileName.Substring(FileName.Length - 4);

If FileType = '.txt' Then

Status := True

Else

Status := False;

IsFileTxt := Status;

End;

Function TFileReader.IsFileReadable(): Boolean;

Begin

Try

Reset(InFile);

Status := True;

Except

Status := False;

End;

CloseFile(InFile);

IsFileReadable := Status;

End;

Function TFileReader.IsFileGood(): Boolean;

Begin

Status := False;

If (Not FileExists(FileName)) Then

Writeln('This file or the path to the file is specified incorrectly or does not

exist! Try again.')

Else If (Not IsFileTxt()) Then

Writeln('This file or path to the file isn', #39, 't .txt! Try again.')

Else If (Not IsFileReadable()) Then

Writeln('The program can', #39, 't read this file! Try again.')

Else

Status := True;

IsFileGood := Status;

End;

Procedure TFileReader.SetFileName;

Begin

Repeat

Writeln('Enter the name of file in this directory or path to this file

including name of file:');

Readln(FileName);

AssignFile(InFile, FileName);

Until IsFileGood();

End;

Function TFileReader.InputSize: Integer;

Var

Size: Integer;

Begin

Size := 0;

Reset(InFile);

Try

Read(InFile, Size);

Except

Writeln('Invalid type. Check data in the file.');

Status := False;

End;

If Size < MIN\_SIZE Then

Begin

Writeln('Matrix order cannot be less than ', MIN\_SIZE, '.');

Status := False;

End;

CloseFile(InFile);

InputSize := Size;

End;

Function TFileReader.InputArray(Const SIZE: Integer): TArray;

Var

Arr: TArray;

I: Integer;

Begin

SetLength(Arr, SIZE);

Reset(InFile);

For I := 1 To SIZE Do

Begin

If Status Then

Try

Read(InFile, Arr[I - 1]);

Except

Writeln('Invalid type. Check data in the file.');

Status := False;

End;

End;

CloseFile(InFile);

InputArray := Arr;

End;

// class Writer

Type

TWriter = Class

Public

Procedure OutputArray(Var Arr: TArray; Const SIZE: Integer);

Virtual; Abstract;

End;

// class ConsoleWriter

Type

TConsoleWriter = Class(TWriter)

Public

Procedure OutputArray(Var Arr: TArray; Const SIZE: Integer); Override;

End;

Procedure TConsoleWriter.OutputArray(Var Arr: TArray; Const SIZE: Integer);

Var

I: Integer;

Begin

Writeln('Sorted array:');

For I := 1 To SIZE Do

Write(Arr[I - 1], ' ');

Writeln;

End;

// class FileWriter

Type

TFileWriter = Class(TWriter)

Private

OutFile: TextFile;

FileName: String;

FileStatus: Boolean;

Public

Procedure OutputArray(Var Arr: TArray; Const SIZE: Integer); Override;

Function IsFileGood(): Boolean;

Function IsFileTxt(): Boolean;

Function IsFileWritable(): Boolean;

End;

Function TFileWriter.IsFileTxt(): Boolean;

Var

FileType: String;

Begin

FileType := FileName.Substring(FileName.Length - 4);

If FileType = '.txt' Then

FileStatus := True

Else

FileStatus := False;

IsFileTxt := FileStatus;

End;

Function TFileWriter.IsFileWritable(): Boolean;

Begin

Try

Rewrite(OutFile);

FileStatus := True;

Except

FileStatus := False;

End;

CloseFile(OutFile);

IsFileWritable := FileStatus;

End;

Function TFileWriter.IsFileGood(): Boolean;

Begin

FileStatus := False;

If (Not FileExists(FileName)) Then

Writeln('This file or the path to the file is specified incorrectly or does not

exist! Try again.')

Else If (Not IsFileTxt()) Then

Writeln('This file or path to the file isn', #39, 't .txt! Try again.')

Else If (Not IsFileWritable()) Then

Writeln('The program can', #39, 't write from this file! Try again.')

Else

FileStatus := True;

IsFileGood := FileStatus;

End;

Procedure TFileWriter.OutputArray(Var Arr: TArray; Const SIZE: Integer);

Var

IsCorrect: Boolean;

I: Integer;

Begin

IsCorrect := False;

Repeat

Writeln('Enter the name of file in this directory or path to this file

including name of file:');

Readln(FileName);

AssignFile(OutFile, FileName);

If IsFileGood() Then

Begin

Rewrite(OutFile);

Writeln(OutFile, 'Sorted array:');

For I := 1 To SIZE Do

Write(OutFile, Arr[I - 1], ' ');

Writeln(OutFile);

Writeln('Answer has been wrote successfully.');

IsCorrect := True;

CloseFile(OutFile);

End;

Until IsCorrect;

End;

// other function

Function InputMethod(): TReader;

Var

IsCorrect: Boolean;

Choice: String;

Reader1: TReader;

Begin

Reader1 := Nil;

IsCorrect := False;

Choice := '';

Writeln('The program works with console input or files.');

Repeat

Writeln('To use console enter ', #39, 'console', #39, '.', #13#10,

'To use a file enter ', #39, 'file', #39, '.'#13#10,

'Enter what type you want to use: ');

Readln(Choice);

If Choice = 'console' Then

Begin

IsCorrect := True;

Reader1 := TConsoleReader.Create;

End

Else If Choice = 'file' Then

Begin

Reader1 := TFileReader.Create;

IsCorrect := True;

End

Else

Writeln('The word ', Choice, ' don', #39,

't match any of method to input the data.');

Until IsCorrect;

InputMethod := Reader1;

End;

Function OutputMethod(): TWriter;

Var

IsCorrect: Boolean;

Choice: String;

Writer1: TWriter;

Begin

Writer1 := Nil;

IsCorrect := False;

Choice := '';

Writeln('The program works with console output or files.');

Repeat

Writeln('To use console enter ', #39, 'console', #39, '.', #13#10,

'To use a file enter ', #39, 'file', #39, '.'#13#10,

'Enter what type you want to use: ');

Readln(Choice);

If Choice = 'console' Then

Begin

Writer1 := TConsoleWriter.Create;

IsCorrect := True;

End

Else If Choice = 'file' Then

Begin

Writer1 := TFileWriter.Create;

IsCorrect := True;

End

Else

Writeln('The word ', Choice, ' don', #39,

't match any of method to output the data.');

Until IsCorrect;

OutputMethod := Writer1;

End;

Procedure InputTask();

Begin

Writeln('The program sorts array of integers.');

End;

// main functions for mergesort

Function Min(First, Second: Integer): Integer;

Var

MinNumber: Integer;

Begin

If First < Second Then

MinNumber := First

Else

MinNumber := Second;

Min := MinNumber;

End;

Procedure Merge(Var Arr: TArray; Beg1, End1, Beg2, End2: Integer);

Var

I, Left, Right, Size: Integer;

CopyArr: TArray;

Begin

Left := 0;

Right := Beg2 - Beg1;

Size := End2 - Beg1 + 1;

SetLength(CopyArr, Size);

For I := 1 To Size Do

CopyArr[I - 1] := Arr[Beg1 + I - 1];

I := beg1;

while (i < beg1 + size) do

begin

if (left + beg1 > end1) Then

Begin

Arr[i] := copyArr[right];

Inc(right);

End

else if (right + beg1 > end2) Then

begin

Arr[i] := copyArr[left];

Inc(Left);

end

else if (copyArr[left] < copyArr[right]) Then

Begin

arr[i] := copyArr[left];

Inc(Left);

End

else

Begin

arr[i] := copyArr[right];

Inc(Right);

End;

inc(I);

end;

End;

Procedure MergeSort(Var Arr: TArray; First, Last: Integer);

Var

Step, J: Integer;

Begin

Step := 1;

While (Step < Last) Do

Begin

J := First;

While (J < Last - Step) Do

Begin

Merge(Arr, J, J + Step - 1, J + Step, Min(J + Step \* 2 - 1, Last - 1));

J := J + Step \* 2;

End;

Step := Step \* 2;

End;

End;

Var

Reader1: TReader;

FileReader1: TFileReader;

Writer1: TWriter;

SizeOfArray: Integer;

Arr: TArray;

IsCorrect: Boolean;

Begin

Reader1 := Nil;

Writer1 := Nil;

FileReader1 := Nil;

InputTask();

Reader1 := InputMethod();

isCorrect := False;

If Reader1.ClassType = TFileReader Then

Begin

FileReader1 := Reader1 As TFileReader;

Repeat

FileReader1.SetFileName();

SizeOfArray := FileReader1.InputSize();

If FileReader1.GetStatus Then

Arr := FileReader1.InputArray(SizeOfArray);

If FileReader1.GetStatus Then

IsCorrect := True

Else

Writeln('Read error was detected in your file.', #13#10,

'This program can', #39, 't continue to read this file.');

Until IsCorrect;

End

Else

Begin

SizeOfArray := Reader1.InputSize();

Arr := Reader1.InputArray(SizeOfArray);

End;

MergeSort(Arr,0,SizeOfArray);

Writer1 := OutputMethod();

Writer1.OutputArray(Arr, SizeOfArray);

// free memory

Reader1 := Nil;

Writer1 := Nil;

End.

**Код программы С++:**

#include <iostream>

#include<fstream>

const int MIN\_SIZE = 2;

class Reader

{

public:

virtual size\_t inputSize() = 0;

virtual int\* inputArray(const size\_t) = 0;

protected:

};

class ConsoleReader : public Reader

{

public:

virtual size\_t inputSize() override;

virtual int\* inputArray(const size\_t) override;

private:

};

size\_t ConsoleReader::inputSize()

{

bool isIncorrect = true;

size\_t size = 0;

// asking for size of array

do

{

std::cout << "Enter the size of array:\n";

std::cin >> size;

if (std::cin.get() != '\n')

{

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

std::cerr << "Invalid type. Try again.\n";

}

else if (size < MIN\_SIZE)

std::cerr << "Size of array cannot be less than " << MIN\_SIZE << ".\n";

else

isIncorrect = false;

} while (isIncorrect);

return size;

}

int\* ConsoleReader::inputArray(size\_t size)

{

bool isIncorrect = true;

int\* arr = nullptr;

// memory allocation of array

arr = new int[size];

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

{

isIncorrect = true;

do

{

std::cout << "Enter " << i + 1 << " element of array:\n";

std::cin >> arr[i];

if (std::cin.get() != '\n')

{

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

std::cerr << "Invalid type. Try again.\n";

}

else

isIncorrect = false;

} while (isIncorrect);

}

return arr;

}

class File

{

public:

File() : fileCode((size\_t)2), fileStream(nullptr)

{}

File(std::string fileName, size\_t fileCode) : fileName(fileName), fileCode(fileCode)

{

if (fileCode == 0)

fileStream = new std::fstream(fileName, std::ios::in);

else

fileStream = new std::fstream(fileName);

}

~File() {

delete fileStream;

};

bool isGood();

bool getStatus();

size\_t getSize();

int\* getArray(const size\_t);

void printArray(int\*,const size\_t);

private:

// code info

// 0 - file for reading

// 1 - file for writting

// 2 - file for reading and writting

//

size\_t fileCode;

std::string fileName;

std::fstream\* fileStream;

bool status = false;

bool isFileExist();

bool isFileText();

bool isFileWorking();

bool isNotEmpty();

};

bool File::getStatus()

{

return status;

}

int\* File::getArray(const size\_t size)

{

fileStream->open(fileName);

int\* arr = new int[size];

\*fileStream >> arr[0];

for (size\_t i = 0; i < size; i++)

{

if(status)

{

\*fileStream >> arr[i];

if (fileStream->fail())

{

fileStream->clear();

std::cerr << "Invalid type. Check data in the file.\n";

status = false;

}

}

}

fileStream->close();

return arr;

}

size\_t File::getSize()

{

fileStream->open(fileName);

size\_t size = 0;

\*fileStream >> size;

if (fileStream->fail())

{

fileStream->clear();

std::cerr << "Invalid type. Check data in the file.\n";

status = false;

}

fileStream->close();

return size;

}

void File::printArray(int\* arr, const size\_t size)

{

fileStream->open(fileName, std::ios::out);

for (size\_t i = 0; i < size; i++)

{

\*fileStream >> arr[i];

}

std::cout << "Answer has been wrote successfully." << std::endl;

}

bool File::isFileExist()

{

if (fileStream->is\_open())

{

fileStream->close();

status = true;

}

else

{

status = false;

std::cerr << "This file or the path to the file is specified incorrectly or

does not exist! Try again.\n";

}

return status;

}

bool File::isNotEmpty()

{

fileStream->open(fileName, std::ios::in);

if (fileStream->peek() != std::fstream::traits\_type::eof())

{

status = true;

}

else

{

status = false;

std::cerr << "This file empty! Try again.\n";

}

fileStream->close();

return status;

}

bool File::isFileText()

{

std::string type = fileName.substr(fileName.length() - (size\_t)(4));

if (type == ".txt")

status = true;

else

{

status = false;

std::cerr << "This file or path to the file isn't .txt! Try again.\n";

}

return status;

}

// if we can read(write) from(to) this file

bool File::isFileWorking()

{

fileStream->open(fileName);

if (fileStream->good())

{

status = true;

}

else

{

status = false;

switch (fileCode)

{

case((size\_t)0):

std::cerr << "The program can't read from this file! Try again.\n";

break;

case((size\_t)1):

std::cerr << "The program can't write down this file! Try again.\n";

break;

default:

std::cerr << "The program can't write down or read from this file! Try

again.\n";

break;

}

}

fileStream->close();

return status;

}

bool File::isGood()

{

if (this->isFileExist() && this->isFileText() && this->isFileWorking())

{

if (fileCode == 0 && this->isNotEmpty())

{

}

}

return status;

}

class FileReader : public Reader

{

public:

FileReader() : file(nullptr)

{};

size\_t inputSize() override;

int\* inputArray(const size\_t) override;

bool getStatus();

void setFileName();

private:

size\_t fileCode = (size\_t)0; // check info about code in class File

File\* file;

};

bool FileReader::getStatus()

{

return file->getStatus();

}

void FileReader::setFileName()

{

std::string fileName;

do

{

std::cout << "Enter the name of file in this directory or path to this file

including name of file:\n";

std::cin >> fileName;

file = new File(fileName, (size\_t)0);

} while (!file->isGood());

}

size\_t FileReader::inputSize()

{

return file->getSize();

}

int\* FileReader::inputArray(const size\_t size)

{

return file->getArray(size);

}

class Writer

{

public:

virtual void outputArray(int\*, const size\_t) = 0;

protected:

};

class ConsoleWriter : public Writer

{

public:

void outputArray(int\*, const size\_t) override;

private:

};

void ConsoleWriter::outputArray(int\* arr, const size\_t size)

{

std::cout << "Sorted array:\n";

for (int i = 0; i < size; i++) {

std::cout << arr[i] << " ";

}

std::cout << std::endl;

}

class FileWriter : public Writer

{

public:

FileWriter() : fileCode((size\_t)1)

{ };

void outputArray(int\*, const size\_t) override;

private:

std::string fileName;

size\_t fileCode = (size\_t)1; // check info about code in class File

};

void FileWriter::outputArray(int\* arr, const size\_t size)

{

File\* out = nullptr;

do

{

delete out;

std::cout << "Enter the name of file in this directory or path to this file

including name of file:\n";

std::cin >> fileName;

out = new File(fileName, fileCode);

} while (!out->isGood());

out->printArray(arr,size);

delete out;

}

//functions for main algorithm

void merge(int\*& arr, size\_t beg1, size\_t end1, size\_t beg2, size\_t end2)

{

size\_t left = 0;

size\_t right = beg2 - beg1;

size\_t size = end2 - beg1 + 1;

int\* copyArr = new int[size];

//copy part of array

for (size\_t i = 0; i < size; i++)

{

copyArr[i] = arr[beg1 + i];

}

for (size\_t i = beg1; i < beg1 + size; i++)

{

if (left + beg1 > end1) {

arr[i] = copyArr[right++];

}

else if (right + beg1 > end2)

{

arr[i] = copyArr[left++];

}

else if (copyArr[left] < copyArr[right])

{

arr[i] = copyArr[left++];

}

else

{

arr[i] = copyArr[right++];

}

}

delete[] copyArr;

}

void mergeSort(int\*& arr, size\_t beg, size\_t end)

{

size\_t step = 1;

while (step < end)

{

for (size\_t j = beg; j < end - step; j += step \* 2)

{

merge(arr, j, j + step - (size\_t)1, j + step,

std::min(j + step \* (size\_t)2 - (size\_t)1, end - (size\_t)1));

}

step \*= 2;

}

}

Reader\* inputMethod()

{

bool isIncorrect = true;

Reader\* reader = nullptr;

std::string choice = "\0";

// asking what the type user want to use

std::cout << "The program works with console input or files.\n";

do

{

std::cout << "To use console enter 'console'.\n"

<< "To use a file enter 'file'.\n"

<< "Enter what type you want to use: \n";

std::cin >> choice;

if (choice == "console")

{

reader = new ConsoleReader();

isIncorrect = false;

}

else if (choice == "file")

{

reader = new FileReader();

isIncorrect = false;

}

else // wrong input

std::cerr << "The word '" << choice << "' don't match any of method to

input the data.\n";

} while (isIncorrect);

return reader;

}

Writer\* outputMethod()

{

bool isIncorrect = true;

std::string choice = "\0";

Writer\* writter = nullptr;

// asking what the type user want to use

std::cout << "The program is ready to show answer.\n";

do

{

std::cout << "To output in console enter 'console'.\n"

<< "To output in a file enter 'file'.\n"

<< "Enter what type you want to use:\n";

std::cin >> choice;

if (choice == "console")

{

writter = new ConsoleWriter();

isIncorrect = false;

}

else if (choice == "file")

{

writter = new FileWriter();

isIncorrect = false;

}

else // wrong input

std::cerr << "The word '" << choice << "' don't match any of method to

output the data.\n";

} while (isIncorrect);

return writter;

}

void inputTask()

{

std::cout << "The program sorts array of integers." << std::endl;

}

int main()

{

bool isIncorrect = true;

size\_t sizeOfArray = 0;

int\* arr = nullptr;

Reader\* reader = nullptr;

Writer\* writer = nullptr;

inputTask();

reader = inputMethod();

if (FileReader\* fileReader = dynamic\_cast<FileReader\*>(reader))

{

do

{

fileReader->setFileName();

sizeOfArray = fileReader->inputSize();

if (fileReader->getStatus()) {

arr = fileReader->inputArray(sizeOfArray);

}

if (fileReader->getStatus()) {

isIncorrect = false;

}

else

{

std::cout << "Read error was detected in your file.\n"

<< "This program can't continue to read this file.\n";

}

} while (isIncorrect);

}

else

{

sizeOfArray = reader->inputSize();

arr = reader->inputArray(sizeOfArray);

}

mergeSort(arr, 0, sizeOfArray);

writer = outputMethod();

writer->outputArray(arr, sizeOfArray);

delete[] arr;

delete writer;

delete reader;

return 0;

}

**Код программы Java:**

**Код Main.java**

package org.example;  
  
import java.util.Scanner;  
  
public class Main {  
  
 //const  
 final static int MIN\_SIZE = 2;  
 static Scanner in = new Scanner(System.in);  
  
 //functions for input  
 static MyReader inputMethod(){  
 MyReader reader = null;  
 boolean isIncorrect = true;  
 String choice = "";  
 System.out.println("The program works with console input or files.");  
 do {  
 System.out.print("""  
 To use console enter 'console'.  
 To use a file enter 'file'.  
 Enter what type you want to use:  
 """);  
 choice = in.nextLine();  
 if (choice.equals("console")) {  
 reader = new ConsoleReader();  
 isIncorrect = false;  
 }  
 else if (choice.equals("file")) {  
 reader = new MyFileReader();  
 isIncorrect = false;  
 }  
 // wrong input  
 else {  
 System.err.println("The word " + choice + " don't match any of method

to input the data.");  
 }  
 } while(isIncorrect);  
 return reader;  
 }  
  
 static MyWriter outputMethod() {  
 MyWriter writer = null;  
 boolean isIncorrect = true;  
 String choice = "";  
 System.out.println("The program is ready to show answer.");  
 do {  
 System.out.print("""  
 To output in console enter 'console'.  
 To output in a file enter 'file'.  
 Enter what type you want to use:  
 """);  
 choice = in.nextLine();  
 if (choice.equals("console"))  
 {  
 writer = new ConsoleWriter();  
 isIncorrect = false;  
 }  
 else if (choice.equals("file"))  
 {  
 writer = new MyFileWriter();  
 isIncorrect = false;  
 }  
 else // wrong input  
 System.err.println("The word '" + choice + "' don't match any of method

to output the data.");  
 } while(isIncorrect);  
 return writer;  
 }  
  
  
 //functions for mergeSort  
 public static int min(int first, int second){  
 return ((first < second) ? first : second);  
 }  
  
 public static void merge(int[] arr, int beg1,int end1,int beg2,int end2){  
 int left = 0;  
 int right = beg2 - beg1;  
 int size = end2-beg1 + 1;  
 int[] copyArr = new int[size];  
 //copy part of array  
 for (int i = 0; i < size; i++) {  
 copyArr[i] = arr[beg1 + i];  
 }  
 for (int i = beg1; i < beg1 + size; i++){  
 if (left+beg1 > end1){  
 arr[i] = copyArr[right++];  
 }  
 else if (right + beg1 > end2){  
 arr[i] = copyArr[left++];  
 }  
 else if (copyArr[left] < copyArr[right]) {  
 arr[i] = copyArr[left++];  
 }  
 else {  
 arr[i] = copyArr[right++];  
 }  
 }  
 }  
  
 public static void mergeSort(int[] arr, int beg, int end){  
 int step = 1;  
 while (step < end){  
 for (int j = beg; j < end - step; j += step \* 2){  
 merge(arr,j,j+step-1,j+step, min(j + step \* 2 - 1, end -1));  
 }  
 step \*= 2;  
 }  
 }  
  
 public static void inputTask(){  
 System.out.println("The program sorts array of integers.");  
 }  
  
 public static void main(String[] args) {  
 boolean isIncorrect = true;  
 int sizeOfArray = 0;  
 int[] array = null;  
 inputTask();  
 MyReader myReader = inputMethod();  
 if (myReader instanceof MyFileReader fileReader) {  
 do {  
 fileReader.setFileName();  
 sizeOfArray = fileReader.inputSize();  
 if (fileReader.getStatus()) {  
 array = fileReader.inputArray(sizeOfArray);  
 }  
 if (fileReader.getStatus()) {  
 isIncorrect = false;  
 }  
 else {  
 System.out.println("Read error was detected in your file.\n" +  
 "This program can't continue to read this

file.");  
 }  
 } while(isIncorrect);  
 }  
 else {  
 sizeOfArray = myReader.inputSize();  
 array = myReader.inputArray(sizeOfArray);  
 }  
 mergeSort(array,0,sizeOfArray);  
 MyWriter myWriter = outputMethod();  
 if (myWriter instanceof MyFileWriter myFileWriter){  
 myFileWriter.setFileName();  
 myFileWriter.outputArray(array);  
 }  
 else {  
 myWriter.outputArray(array);  
 }  
  
 }  
}

**Код MyReader.java**

package org.example;  
  
public abstract class MyReader {  
 public abstract int[] inputArray(final int Size);  
  
 public abstract int inputSize();  
  
 public void invalidTypeMessage(){  
 System.err.println("Invalid type! Try again.");  
 }  
 public void emptyStringMessage()  
 {  
 System.err.println("Your string Empty! Try again.");  
 }  
}

**Код ConsoleReader.java**

package org.example;

public class ConsoleReader extends MyReader {  
 @Override  
 public int inputSize(){  
 boolean isIncorrect = true;  
 int size = 0;  
 // asking for Size of array  
 do {  
 try  
 {  
 System.out.println("Enter size of array:");  
 size = Integer.parseInt(Main.in.nextLine());  
 isIncorrect = false;  
 }  
 catch (NumberFormatException exception)  
 {  
 invalidTypeMessage();  
 }  
 if (size < Main.MIN\_SIZE) {  
 System.err.printf("Size of array cannot be less than %d! Try again.\n",

Main.MIN\_SIZE);  
 isIncorrect = true;  
 }  
 }while(isIncorrect);  
 return size;  
 }  
 @Override  
 public int[] inputArray(final int size) {  
 String inputtedString = "";  
 boolean isIncorrect = true;  
 int inputtedNumber = 0;  
 int[] array = new int[size];  
 System.out.println("Input array by elements.");  
 for (int i = 0; i < size; i++) {  
 do {  
 try {  
 System.out.println("Enter " + (i + 1) + " element:");  
 inputtedNumber = Integer.parseInt(Main.in.nextLine());  
 array[i] = inputtedNumber;  
 isIncorrect = false;  
 } catch (NumberFormatException ex){  
 invalidTypeMessage();  
 }  
 }while(isIncorrect);  
 }  
 return array;  
 }  
}

**Код MyFileReader.java**

package org.example;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class MyFileReader extends MyReader {  
 private String fileName;  
  
 private boolean status;  
 MyFileReader()  
 {}  
 MyFileReader(String fileName){  
 this.fileName = fileName;  
 }  
  
 public boolean getStatus() {  
 return status;  
 }  
  
 @Override  
 public int inputSize() {  
 int size = 0;  
 char currentChar = '\0';  
 boolean wasNumber = false;  
 boolean isCorrect = false; // for exit the loop  
 String number = ""; // string for parse Number  
 int codeOfChar = 0;  
 try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {  
 while(status && !isCorrect && (codeOfChar = reader.read()) != -1) {  
 currentChar = (char)(codeOfChar);  
 if (Character.isAlphabetic(currentChar)) {  
 invalidTypeMessage();  
 status = false;  
 wasNumber = false;  
 }  
 else if ((Character.isDigit(currentChar))) {  
 number += currentChar;  
 wasNumber = true;  
 }  
 else if (wasNumber) {  
 size = Integer.parseInt(number);  
 isCorrect = true;  
 }  
 }  
 }  
 catch(IOException ex){  
 System.err.println("Oops! Something went wrong.");  
 }  
 if (size < Main.MIN\_SIZE && wasNumber)  
 {  
 System.err.printf("Matrix order cannot be less than %d.\n", Main.MIN\_SIZE);  
 status = false;  
 }  
 return size;  
 }  
 @Override  
 public int[] inputArray(final int size) {  
 boolean isIncorrect = true;  
 char currentChar = '\0';  
 boolean wasNumber = false;  
 boolean isCorrect = false; // for exit the loop  
 String number = ""; // string for parse Number  
 int character = 0;  
 int counter = 0;  
 int[] array = new int[size]; // memory allocation  
 try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {  
 reader.readLine();  
 for (int i = 0; i < size; i++) {  
 isCorrect = false;  
 while (status && !isCorrect && (character = reader.read()) != -1) {  
 currentChar = (char) (character);  
 if (Character.isAlphabetic(currentChar)) {  
 invalidTypeMessage();  
 status = false;  
 } else if ((Character.isDigit(currentChar)) || currentChar == '.') {  
 number += currentChar;  
 wasNumber = true;  
 } else if (wasNumber) {  
 array[i] = Integer.parseInt(number);  
 number = "";  
 isCorrect = true;  
 counter++;  
 }  
 }  
 if (!number.isEmpty()){  
 array[i] = Integer.parseInt(number);  
 counter++;  
 }  
 }  
 } catch (IOException ex) {  
 System.err.println("Oops! Something went wrong.");  
 }  
 if (counter != size){  
 status = false;  
 System.err.println("Function didn't found all numbers.");  
 }  
 return array;  
 }  
  
 public void setFileName(){  
 do {  
 System.out.println("Enter the name of file in this directory or path to

this file including name of file:");  
 fileName = Main.in.nextLine();  
 }while(!isFileGood());  
 }  
 public boolean isFileGood() {  
 status = false;  
 File inputfile = new File(fileName);  
 // if file doesn't exist  
 if (!inputfile.exists()) {  
 System.err.println("This file or the path to the file is specified

incorrectly or does not exist! Try again.");  
 }  
 // if file isn't .txt  
 else if (!fileName.endsWith(".txt")) {  
 System.err.println("This file or path to the file isn't .txt! Try again.");  
 }  
 else if (!inputfile.canRead()) {  
 System.err.println("The program can't read this file! Try again.");  
 }  
 else if (inputfile.length() == 0){  
 emptyStringMessage();  
 }  
 else {  
 status = true;  
 }  
 return status;  
 }  
}

**Код Writer.java**

package org.example;  
  
abstract public class MyWriter {  
 abstract public void outputArray(int[] array);  
}

**Код ConsoleWriter.java**

package org.example;  
  
public class ConsoleWriter extends MyWriter {  
 @Override  
 public void outputArray(int[] array) {  
 int size = array.length;  
 System.out.println("Sorted array:");  
 for (int i = 0; i < size; i++) {  
 System.out.print(array[i] + " ");  
 }  
 System.out.println();  
 }  
}

**Код FileWriter.java**

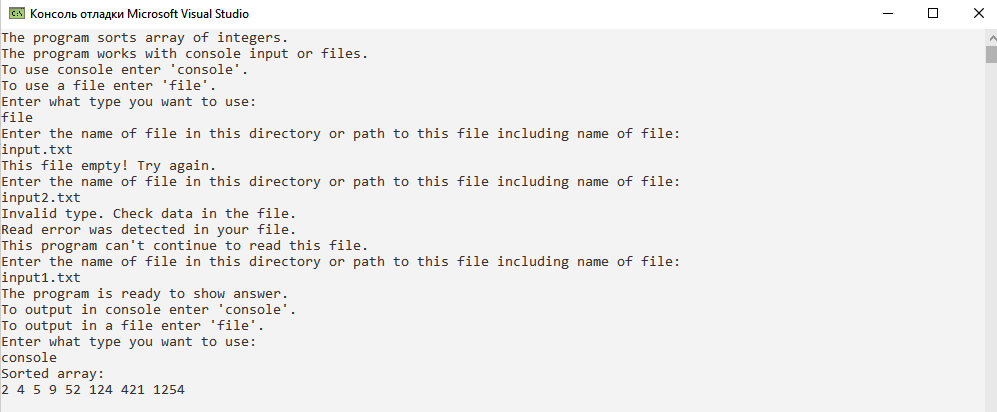
package org.example;  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class MyFileWriter extends MyWriter {  
  
 private String fileName;  
  
 private boolean status;  
 MyFileWriter()  
 {}  
 MyFileWriter(String fileName){  
 this.fileName = fileName;  
 }  
 public boolean isFileGood()  
 {  
 status = false;  
 File inputfile = new File(fileName);  
 // if file doesn't exist  
 if (!inputfile.exists()) {  
 System.err.println("This file or the path to the file is specified

incorrectly or does not exist! Try again.");  
 }  
 // if file isn't .txt  
 else if (!fileName.endsWith(".txt")) {  
 System.err.println("This file or path to the file isn't .txt! Try again.");  
 }  
 else if (!inputfile.canWrite()) {  
 System.err.println("The program can't read this file! Try again.");  
 }  
 else {  
 status = true;  
 }  
 return status;  
 }  
  
 public void setFileName(){  
 do {  
 System.out.println("Enter the name of file in this directory or path to

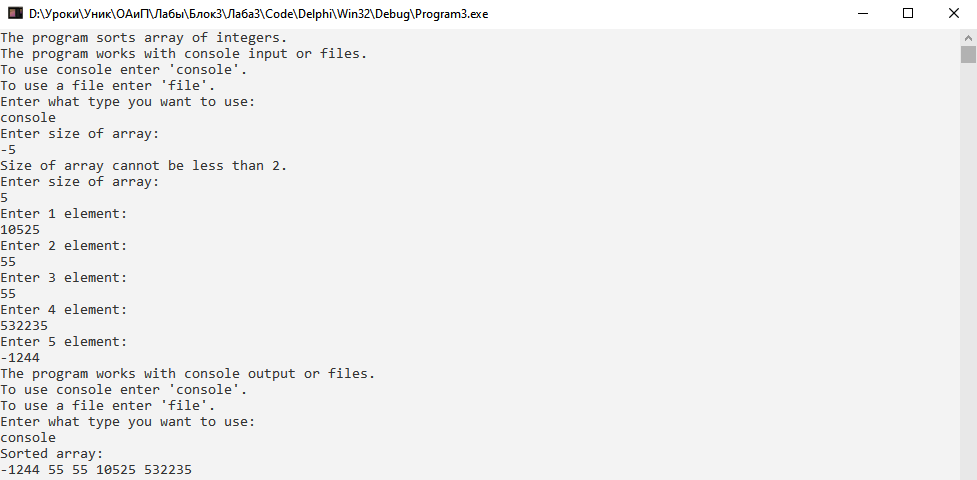
this file including name of file:");  
 fileName = Main.in.nextLine();  
 }while(!isFileGood());  
 }  
  
 @Override  
 public void outputArray(int[] array) {  
 try (BufferedWriter out = new BufferedWriter(new FileWriter(fileName))) {  
 int size = array.length;  
 out.write("Sorted array:\n");  
 for (int i = 0; i < size; i++) {  
 out.write(array[i] + " ");  
 }  
 out.write('\n');  
 System.out.println("Answer has been wrote successfully.");  
 } catch (IOException ex) {  
 System.err.println("Oops! Something went wrong.");  
 }  
 }  
}

**Скриншоты:**

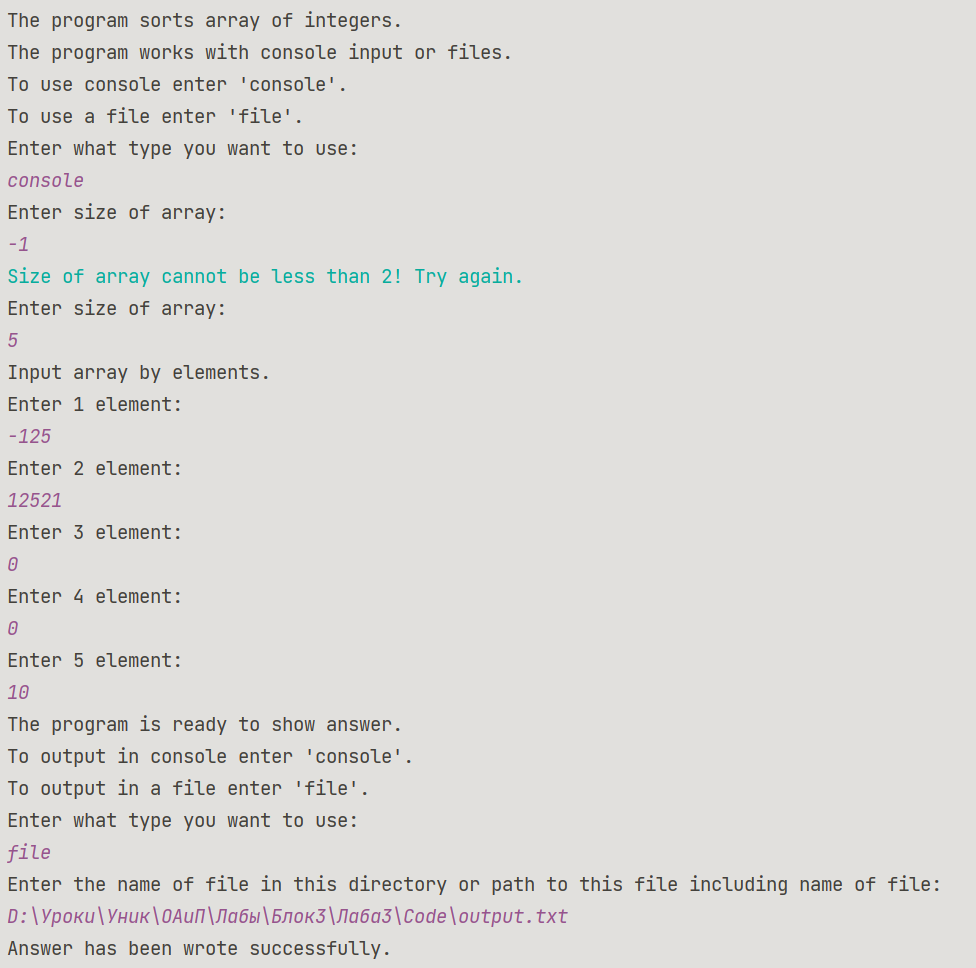
**C++:**



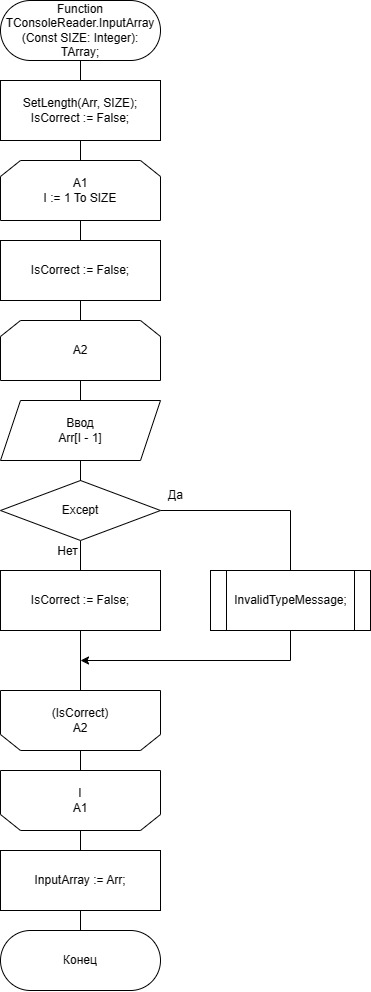
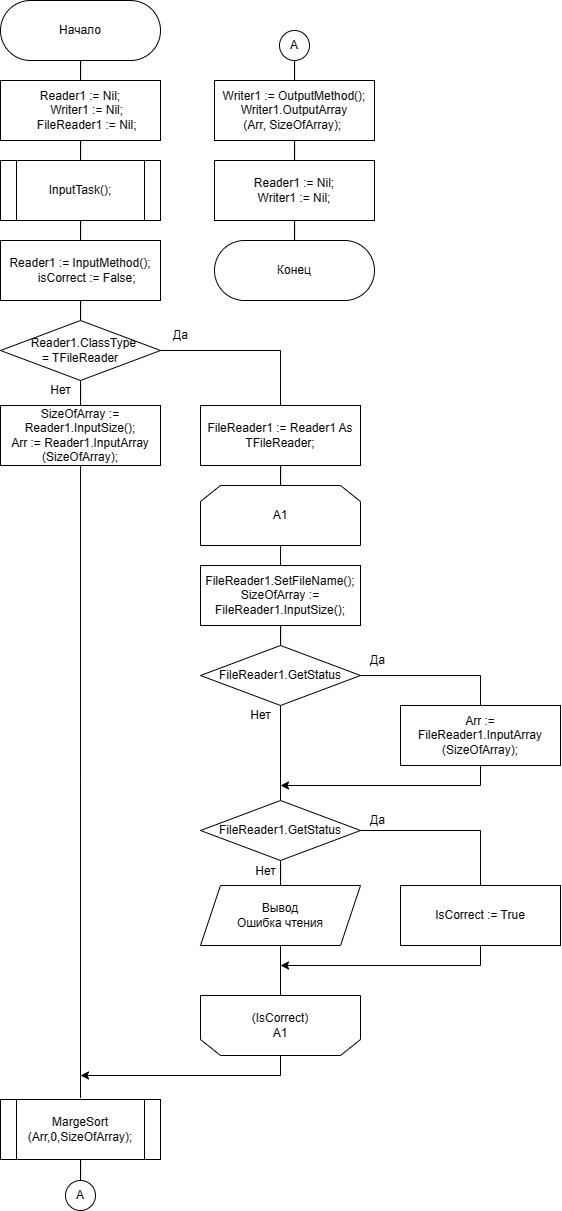
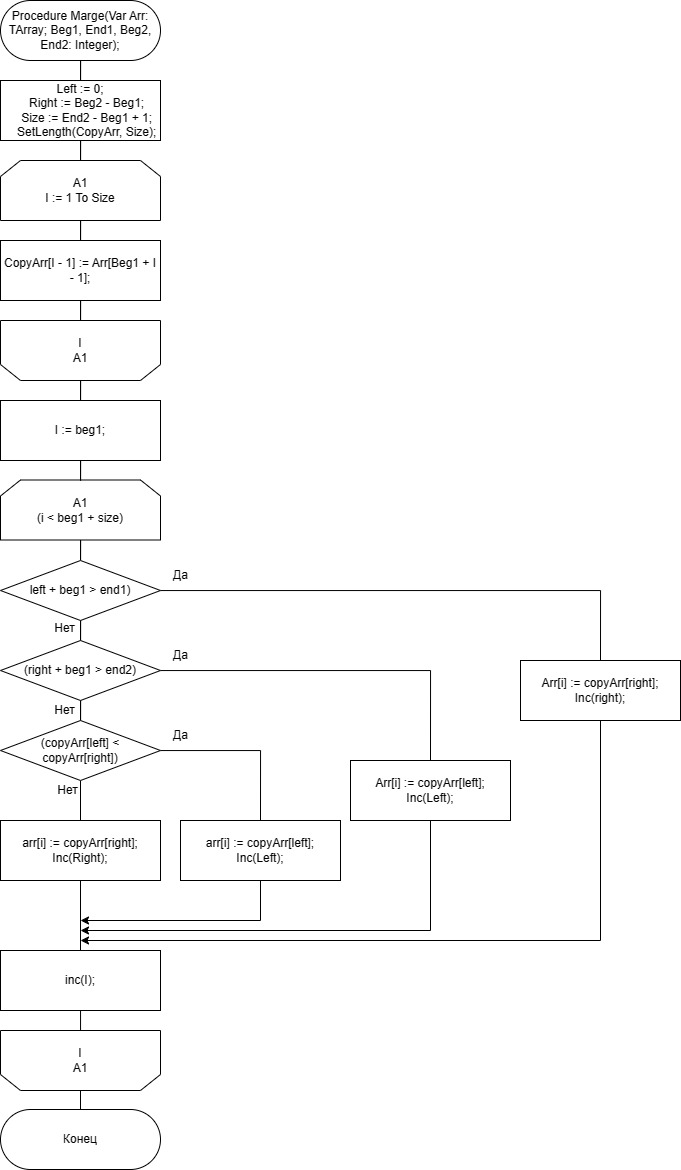
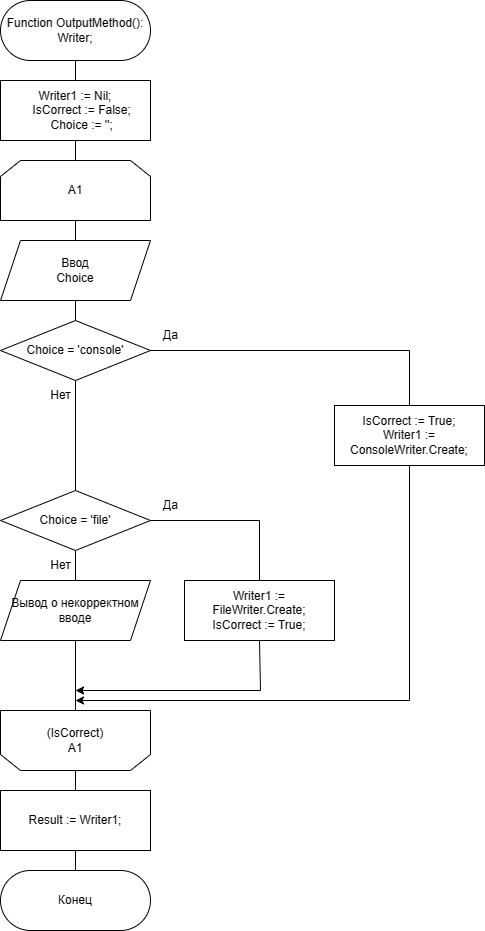
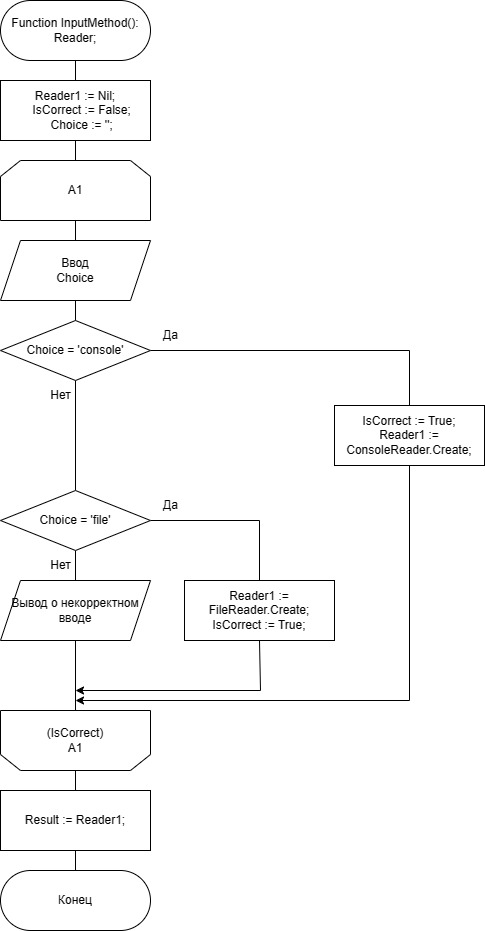
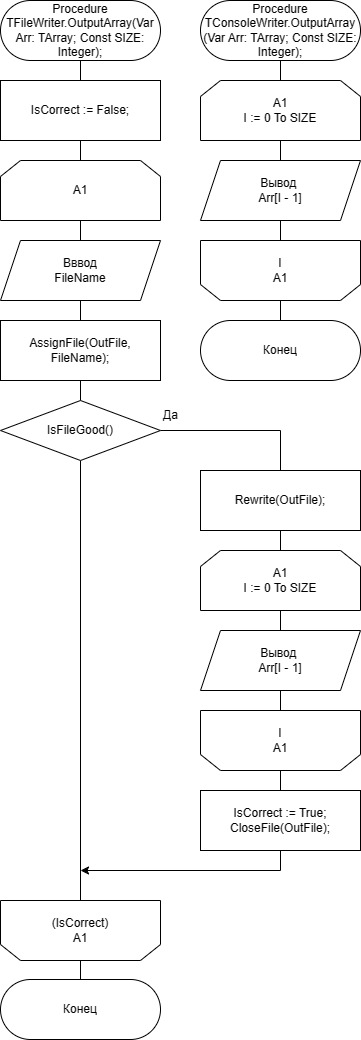
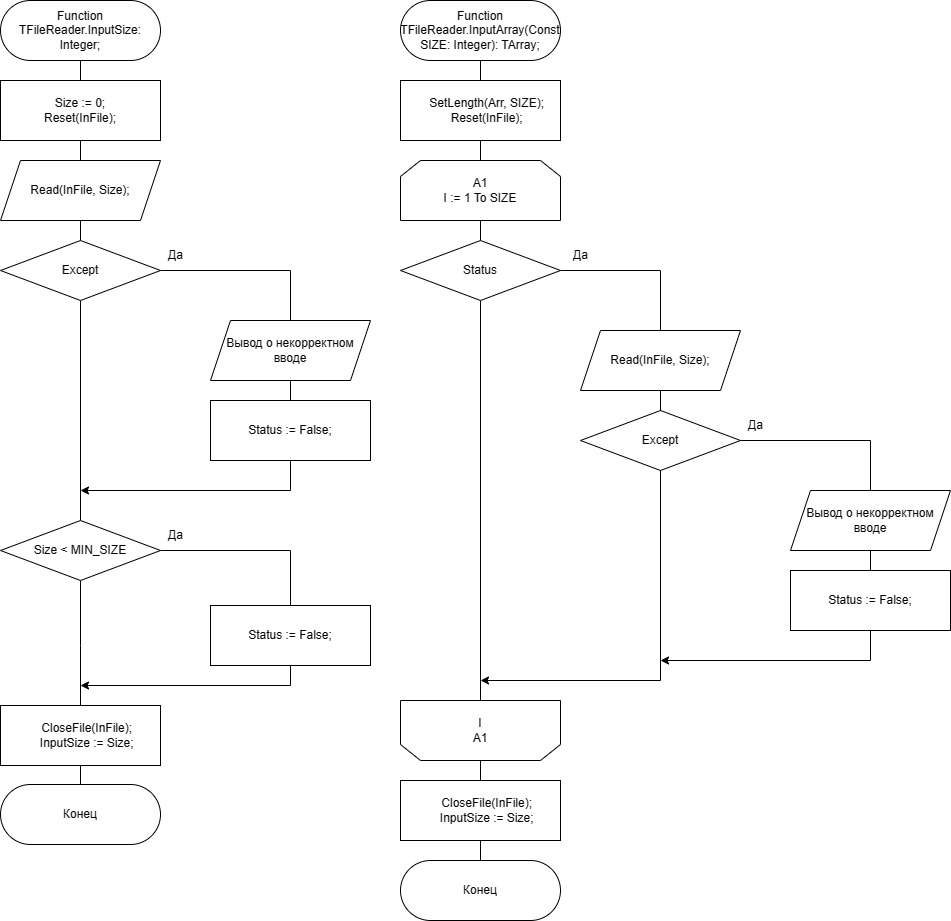
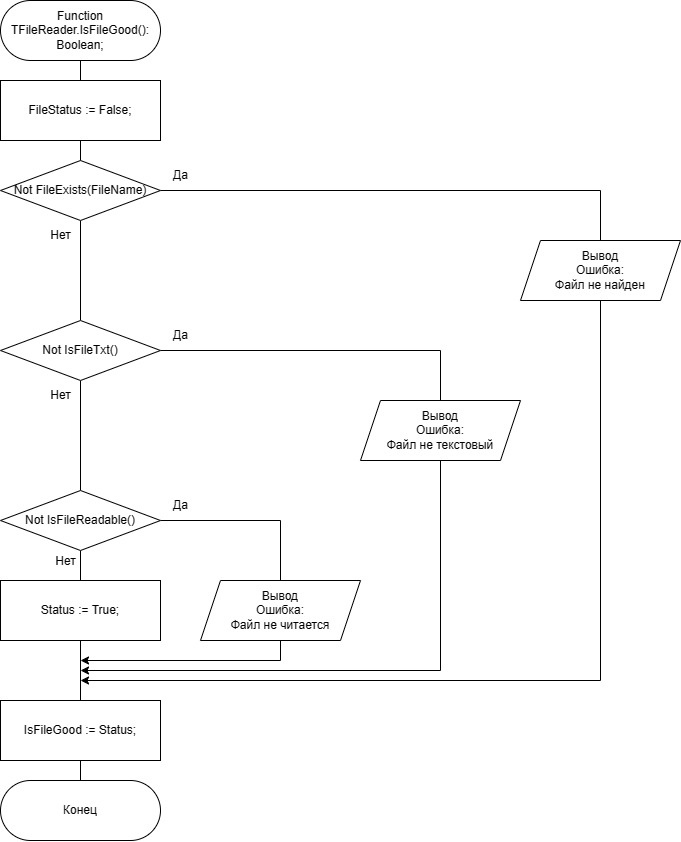
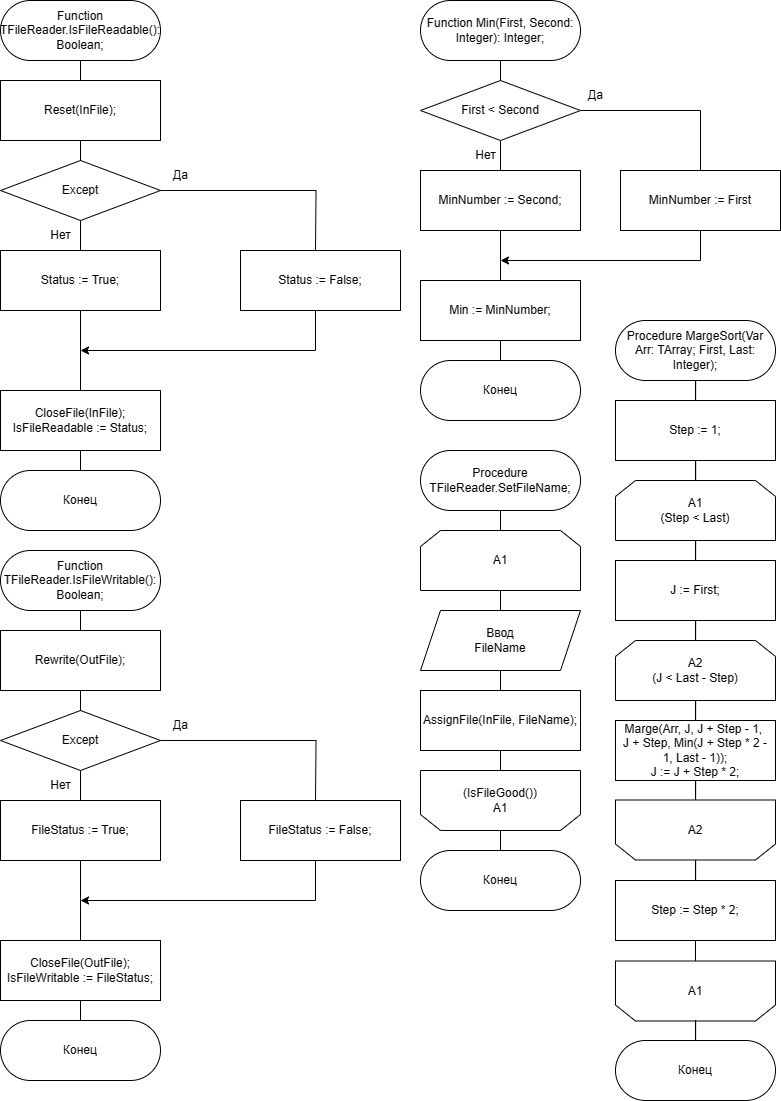
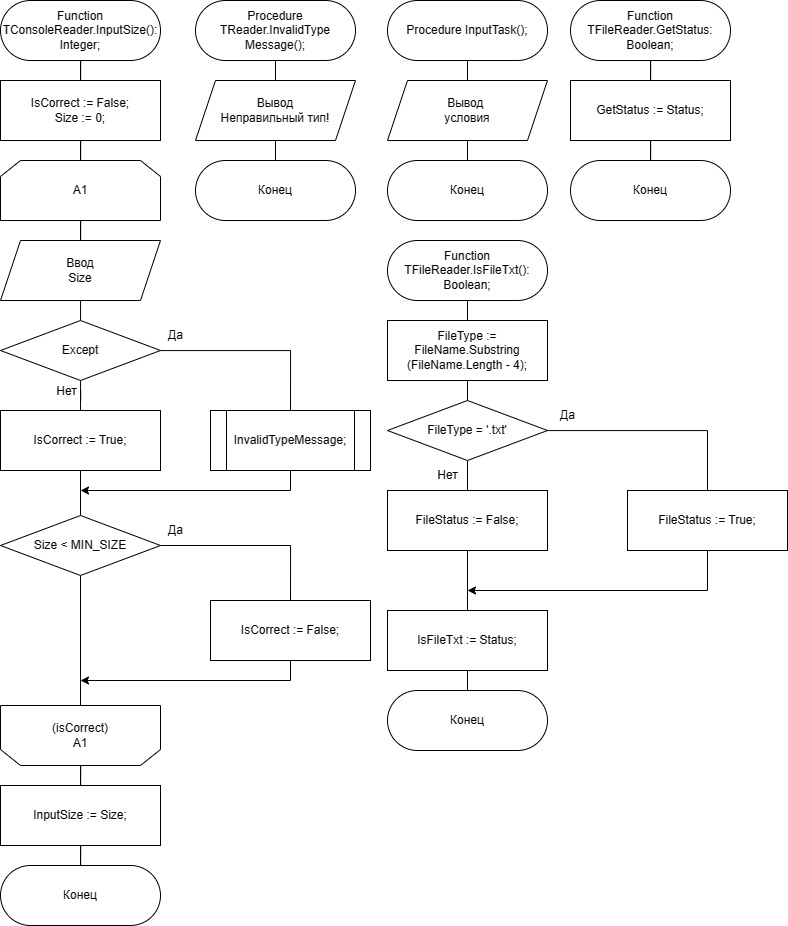
**Delphi:**



**Java:**



**Блок-схема:**



**Юнит Тесты (Unit Tests)**

**Код Unit Tests**

**Tests.java**

package org.example;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import org.junit.jupiter.api.Test;  
class FunctionMinTest {  
  
 @Test  
 void Min\_FirstMoreThanSecond\_FirstNumber(){  
 assertEquals(Main.min(10,20),10);  
 }  
  
 @Test  
 void Min\_FirstLowerThanSecond\_SecondNumber() {  
 assertEquals(Main.min(15,5),5);  
 }  
  
 @Test  
 void Min\_FirstEqualsSecond\_FirstNumber() {  
 assertEquals(Main.min(1,1),1);  
 }  
}  
  
class FunctionMergeSort {  
 @Test  
 void MergeSort\_SortedArray\_TheSameArray(){  
  
 int[] testArray = {-1,0,121,224,636,800,12121};  
 Main.mergeSort(testArray,0,7);  
 int[] answer = {-1,0,121,224,636,800,12121};  
 assertArrayEquals(testArray,answer);  
 }  
   
 @Test  
 void MergeSort\_UnSortedArrayOfUnsignedNumbers\_SortedArray(){  
 int[] testArray = {12121,10,121,0,224,800,636} ;  
 Main.mergeSort(testArray,0,7);  
 int[] answer = {0,10,121,224,636,800,12121};  
 assertArrayEquals(testArray,answer);  
 }  
  
 @Test  
 void MergeSort\_UnSortedArrayOfAllNumbers\_SortedArray(){  
 int[] testArray = {12121,-21512,121,0,-2552,224,800,636};  
 Main.mergeSort(testArray,0,8);  
 int[] answer = {-21512,-2552,0,121,224,636,800,12121};  
 assertArrayEquals(testArray,answer);  
 }  
  
 @Test  
 void MergeSort\_UnSortedArrayOfUnsignedNumbersWithDuplicates\_SortedArray(){  
 int[] testArray = {800,12121,10,121,0,224,800,10,636};  
 Main.mergeSort(testArray,0,9);  
 int[] answer = {0,10,10,121,224,636,800,800,12121};  
 assertArrayEquals(testArray,answer);  
 }  
  
 @Test  
 void MergeSort\_UnSortedArrayOfAllNumbersWithDuplicates\_SortedArray(){  
 int[] testArray = {0,12121,-21512,121,0,-2552,224,800,-21512,636};  
 Main.mergeSort(testArray,0,10);  
 int[] answer = {-21512,-21512,-2552,0,0,121,224,636,800,12121};  
 assertArrayEquals(testArray,answer);  
 }  
}  
  
  
class FunctionIsFileGoodFromMyFileReader{  
  
 @Test  
 void IsFileGood\_givenMp3File\_False(){  
 MyFileReader testFileReader = new MyFileReader("D:\\Уроки\\Уник\\ОАиП\\

Лабы\\Блок3\\Лаба3\\Code\\

music.mp3");  
 assertFalse(testFileReader.isFileGood());  
 }  
  
 @Test  
 void IsFileGood\_givenUnexistFile\_False(){  
 MyFileReader testFileReader = new MyFileReader("D:\\Уроки\\Уник\\ОАиП\\

Лабы\\Блок3\\Лаба3\\Code\\

music.txt");  
 assertFalse(testFileReader.isFileGood());  
 }  
  
 @Test  
 void IsFileGood\_givenEmptyFile\_False(){  
 MyFileReader testFileReader = new MyFileReader("D:\\Уроки\\Уник\\ОАиП\\

Лабы\\Блок3\\Лаба3\\Code\\

input.txt");  
 assertFalse(testFileReader.isFileGood());  
 }  
  
 @Test  
 void IsFileGood\_givenGoodFile\_True(){  
 MyFileReader testFileReader = new MyFileReader("D:\\Уроки\\Уник\\ОАиП\\

Лабы\\Блок3\\Лаба3\\Code\\

input1.txt");  
 assertTrue(testFileReader.isFileGood());  
 }  
  
}  
  
  
class FunctionIsFileGoodFromMyFileWriter{  
  
 @Test  
 void IsFileGood\_givenMp3File\_False(){  
 MyFileWriter testFileWriter = new MyFileWriter("D:\\Уроки\\Уник\\ОАиП\\

Лабы\\Блок3\\Лаба3\\Code\\

music.mp3");  
 assertFalse(testFileWriter.isFileGood());  
 }  
  
 @Test  
 void IsFileGood\_givenUnexistFile\_False(){  
 MyFileWriter testFileWriter = new MyFileWriter("D:\\Уроки\\Уник\\ОАиП\\

Лабы\\Блок3\\Лаба3\\Code\\

music.txt");  
 assertFalse(testFileWriter.isFileGood());  
 }  
  
 @Test  
 void IsFileGood\_givenGoodFile\_True(){  
 MyFileWriter testFileWriter = new MyFileWriter("D:\\Уроки\\Уник\\ОАиП\\

Лабы\\Блок3\\Лаба3\\Code\\

output.txt");  
 assertTrue(testFileWriter.isFileGood());  
 }  
  
}

**Скриншоты:**

