УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.2

по предмету «Основы алгоритмизации и программирования»

Вариант 5

Выполнил:

Бражалович А. И.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Найти все простые числа, не превосходящие данного натурального числа P. Для решения задачи воспользоваться известным алгоритмом «решето Эратосфена». Для решения использовать множества.

**Код программы Delphi:**

Program Lab3\_2;

Uses

System.SysUtils;

Type

TSet = Set Of 2..255;

ERRORS\_LIST = (CORRECT, RANGE\_ERR, NUM\_ERR, NOT\_TXT, NOT\_EXIST, NOT\_READABLE, NOT\_WRITEABLE, CHOICE\_ERR, FILE\_EMPTY);

Const

MIN\_NUMBER = 2;

MAX\_NUMBER = 254;

FILE\_CHOICE = 1;

CONSOLE\_CHOICE = 2;

ERRORS: Array [ERRORS\_LIST] Of String = ('', 'Значение не попадает в диапазон!', 'Проверьте корректность ввода данных!', 'Расширение не txt!', 'Проверьте корректность ввода пути к файлу!', 'Файл закрыт для чтения!', 'Файл закрыт для записи!', 'Проверьте корректность выбора!', 'Файл пуст!');

Procedure PrintTask();

Begin

WriteLn('Данная программа ищет все простые числа до числа P', #13#10);

End;

Function CheckArea(Num: Integer; MIN, MAX: Integer) : ERRORS\_LIST;

Var

Errors: ERRORS\_LIST;

Begin

Errors := CORRECT;

If (Num < MIN) Or (Num > MAX) Then

Errors := RANGE\_ERR;

CheckArea := Errors;

End;

Procedure PrintError(Error: ERRORS\_LIST);

Begin

WriteLn(ERRORS[Error], #13#10'Повторите попытку: ');

End;

Function CheckNum(MIN, MAX: Integer) : Integer;

Var

Errors: ERRORS\_LIST;

Num: Integer;

Begin

Num := 0;

Repeat

Errors := CORRECT;

Try

Readln(Num);

Except

Errors := NUM\_ERR;

End;

If Errors = CORRECT Then

Errors := CheckArea(Num, MIN, MAX);

If Errors <> CORRECT Then

PrintError(Errors);

Until Errors = CORRECT;

CheckNum := Num;

End;

Function CheckInOut() : Boolean;

Var

Num: Integer;

Choose: Boolean;

Begin

Choose := False;

Num := CheckNum(FILE\_CHOICE, CONSOLE\_CHOICE);

If Num = 1 Then

Choose := True;

CheckInOut := Choose;

End;

Function ChooseFileInput() : Boolean;

Var

Choose: Boolean;

Begin

WriteLn('Вы хотите вводить число через файл? (Да - ', 1, ' / Нет - ', 2, ')');

Choose := CheckInOut();

ChooseFileInput := Choose;

End;

Function IsReadable (Var F: TextFile) : ERRORS\_LIST;

Var

Errors: ERRORS\_LIST;

Begin

Errors := CORRECT;

Try

Try

Reset(F);

Finally

CloseFile(F);

End;

Except

Errors := NOT\_READABLE;

End;

IsReadable := Errors;

End;

Procedure FileReading(Var F: TextFile);

Var

Errors: ERRORS\_LIST;

PathToFile: String;

Begin

PathToFile := '';

Repeat

Errors := CORRECT;

Write('Введите путь к файлу с расширением .txt: ');

Readln(PathToFile);

If ExtractFileExt(PathToFile) <> '.txt' Then

Errors := NOT\_TXT;

If Not FileExists(PathToFile) And (Errors = CORRECT) Then

Errors := NOT\_EXIST;

If EOF(F) And (Errors = CORRECT) Then

Errors := FILE\_EMPTY;

If Errors = CORRECT Then

AssignFile(F, PathToFile);

If (Errors = CORRECT) And (IsReadable(F) <> CORRECT) Then

Errors := NOT\_READABLE;

If Errors <> CORRECT Then

PrintError(Errors);

Until Errors = CORRECT;

End;

Procedure FileWriting(Var F: TextFile);

Var

PathToFile: String;

Errors: ERRORS\_LIST;

Begin

PathToFile := '';

Repeat

Errors := CORRECT;

Write('Введите путь к файлу с расширением .txt: ');

Readln(PathToFile);

If ExtractFileExt(PathToFile) <> '.txt' Then

Errors := NOT\_TXT;

If Not FileExists(PathToFile) And (Errors = CORRECT) Then

Errors := NOT\_EXIST;

If Errors = CORRECT Then

AssignFile(F, PathToFile);

If (ERRORS = CORRECT) And (FileIsReadOnly(PathToFile)) Then

Errors := NOT\_WRITEABLE;

If Errors <> CORRECT Then

PrintError(Errors);

Until Errors = CORRECT;

End;

Function ReadSet() : TSet;

Var

RF: TextFile;

Numbers: TSet;

FromFile: Boolean;

Num: Integer;

Error: ERRORS\_LIST;

Begin

Error := CORRECT;

FromFile := ChooseFileInput();

Num := 0;

If FromFile Then

Begin

FileReading(RF);

Reset(RF);

Try

Readln(RF, Num);

Except

Error := NUM\_ERR;

End;

If Error = CORRECT Then

Error := CheckArea(Num, MIN\_NUMBER, MAX\_NUMBER);

CloseFile(RF);

Numbers := [2..Num];

If Error <> CORRECT Then

PrintError(Error);

End

Else

Begin

Writeln('Введите число до которого вы хотетие найти простые числа [', MIN\_NUMBER, ':', MAX\_NUMBER, ']');

Num := CheckNum(MIN\_NUMBER, MAX\_NUMBER);

Numbers := [2..Num];

End;

ReadSet := Numbers;

End;

Function SortSet(PrimeNumbers: TSet): TSet;

Var

Prime: Integer;

I, Count: Integer;

Begin

Count := 1;

Prime := MIN\_NUMBER;

For I := MIN\_NUMBER To MAX\_NUMBER Do

Begin

If (I In PrimeNumbers) Then

Inc(Count);

End;

While (Prime \* Prime < Count) Do

Begin

If Prime In PrimeNumbers Then

Begin

For I := 2 \* Prime To Count Do

If I Mod Prime = 0 Then

Exclude(PrimeNumbers, I);

End;

Inc(Prime);

End;

SortSet := PrimeNumbers;

End;

Function ChooseFileOutput() : Boolean;

Var

Choose: Boolean;

Begin

WriteLn('Вы хотите выводить ответ через файл? (Да - ', 1, ' / Нет - ', 2, ')');

Choose := CheckInOut();

ChooseFileOutput := Choose;

End;

Procedure PrintResult(PrimeNumbers: TSet);

Var

Num: Integer;

F: TextFile;

PrintToFile: Boolean;

Begin

Num := 0;

PrintToFile := ChooseFileOutput();

If PrintToFile Then

Begin

FileWriting(F);

Append(F);

Write(F, 'Множество простых чисел: ');

End

Else

Write('Множество простых чисел:');

For Num in PrimeNumbers Do

Begin

If PrintToFile Then

Write(F, '''', Num, '''; ')

Else

Write('''', Num, '''; ');

End;

If PrintToFile Then

CloseFile(F);

End;

Var

F: TextFile;

PrimeNumbers: TSet;

Begin

PrintTask();

PrimeNumbers := ReadSet();

PrimeNumbers := SortSet(PrimeNumbers);

PrintResult(PrimeNumbers);

ReadLn;

End.

**Код программы С++:**

#include <iostream>

#include <string>

#include <fstream>

#include <set>

using namespace std;

enum ERRORS\_LIST {

CORRECT, RANGE\_ERR, NUM\_ERR, NOT\_TXT, NOT\_EXIST, NOT\_READABLE, NOT\_WRITEABLE, FILE\_EMPTY

};

const string ERRORS[] = {

"", "Значение не попадает в диапазон!", "Проверьте корректность ввода данных!", "Расширение не txt!", "Проверьте корректность ввода пути к файлу!", "Файл закрыт для чтения!", "Файл закрыт для записи!", "Файл пуст!"

};

constexpr int MIN\_NUMBER = 2;

constexpr int MAX\_NUMBER = 10000;

void printTask()

{

cout << "Данная программа ищет все простые числа до числа P:\n\n";

}

ERRORS\_LIST checkArea(int num, const int MIN, const int MAX)

{

ERRORS\_LIST error;

error = CORRECT;

if (num < MIN || num > MAX)

error = RANGE\_ERR;

return error;

}

void printError(ERRORS\_LIST error)

{

cout << ERRORS[error] << "\nПовторите попытку: ";

}

int checkNum(int MIN, int MAX)

{

ERRORS\_LIST error;

int num;

num = 0;

do

{

error = CORRECT;

cin >> num;

if (cin.fail())

{

error = NUM\_ERR;

cin.clear();

while (cin.get() != '\n');

}

if (error == CORRECT && cin.get() != '\n')

{

error = NUM\_ERR;

cout << "Некорректный выбор!\n";

while (cin.get() != '\n');

}

if (error == CORRECT)

error = checkArea(num, MIN, MAX);

if (error != CORRECT)

printError(error);

} while (error != CORRECT);

return num;

}

bool checkInOut()

{

const int FILE\_CHOICE = 1;

const int CONSOLE\_CHOICE = 2;

int num;

bool choose;

choose = false;

num = checkNum(FILE\_CHOICE, CONSOLE\_CHOICE);

if (num == 1)

choose = true;

return choose;

}

bool chooseFileInput()

{

bool choose;

choose = false;

cout << "Вы хотите вводить число через файл? (Да - " << 1 << " / Нет - " << 2 << ")\n";

choose = checkInOut();

return choose;

}

void fileReading(string& pathToFile)

{

ERRORS\_LIST error;

do

{

error = CORRECT;

cout << "Введите путь к файлу с расширением .txt: ";

getline(cin, pathToFile);

if (pathToFile.substr(pathToFile.length() - 4, 4) != ".txt")

error = NOT\_TXT;

else if (!ifstream(pathToFile))

error = NOT\_EXIST;

else

{

ifstream file(pathToFile);

if (!file.is\_open())

error = NOT\_READABLE;

else if (file.peek() == ifstream::traits\_type::eof())

{

error = FILE\_EMPTY;

file.close();

}

file.close();

}

if (error != CORRECT)

printError(error);

} while (error != CORRECT);

}

void fileWriting(string& pathToFile)

{

ERRORS\_LIST error;

do

{

error = CORRECT;

cout << "Введите путь к файлу с расширением .txt: ";

getline(cin, pathToFile);

if (pathToFile.substr(pathToFile.length() - 4, 4) != ".txt")

error = NOT\_TXT;

else if (!ifstream(pathToFile))

error = NOT\_EXIST;

else

{

ifstream file(pathToFile);

if (!file.is\_open())

error = NOT\_WRITEABLE;

}

if (error != CORRECT)

printError(error);

} while (error != CORRECT);

}

set<int> readSet()

{

set<int> numbers;

bool fromFile;

int num;

ERRORS\_LIST error;

string pathToFile;

num = 0;

error = CORRECT;

pathToFile= "";

fromFile = chooseFileInput();

if (!fromFile)

{

cout << "Введите число до которого вы хотетие найти простые числа [" << MIN\_NUMBER << ":" << MAX\_NUMBER << "] ";

num = checkNum(MIN\_NUMBER, MAX\_NUMBER);

for (int i = 2; i <= num; ++i)

numbers.insert(i);

}

else

{

do

{

fileReading(pathToFile);

ifstream file(pathToFile);

file >> num;

if (file.fail())

{

error = NUM\_ERR;

file.clear();

}

if (error == CORRECT)

if (file.peek() != '\n')

error = NUM\_ERR;

if (error == CORRECT)

error = checkArea(num, MIN\_NUMBER, MAX\_NUMBER);

file.close();

if (error != CORRECT)

printError(error);

} while (error != CORRECT);

for (int i = 2; i <= num; ++i)

numbers.insert(i);

}

return numbers;

}

set<int> sortSet(const set<int> primeNumbers)

{

set<int> sortedSet;

int prime;

int length;

prime= MIN\_NUMBER;

sortedSet = primeNumbers;

length = sortedSet.size() + 1;

while (prime \* prime <= length) {

if (sortedSet.count(prime) > 0)

for (int i = 2 \* prime; i <= length; i += prime)

sortedSet.erase(i);

++prime;

}

return sortedSet;

}

bool chooseFileOutput()

{

bool choose;

cout << "\nВы хотите выводить ответ через файл? (Да - " << 1 << " / Нет - " << 2 << ")\n";

choose = checkInOut();

return choose;

}

void printResult(set<int> primeNumbers)

{

int num;

string pathToFile;

bool printToFile;

num = 0;

printToFile = chooseFileOutput();

pathToFile = ' ';

if (printToFile)

{

fileWriting(pathToFile);

ofstream fileOut(pathToFile, std::ios::app);

fileOut << "Множество простых чисел:";

}

else

cout << "Множество простых чисел:\n";

ofstream fileOut(pathToFile, std::ios::app);

for (int num : primeNumbers)

{

if (printToFile)

fileOut << num << " ";

else

cout << num << " ";

}

if (printToFile)

fileOut.close();

}

int main()

{

set<int> primeNumbers;

setlocale(LC\_ALL, "RU");

printTask();

primeNumbers = readSet();

primeNumbers = sortSet(primeNumbers);

printResult(primeNumbers);

return 0;

}

**Код программы Java:**

import java.io.\*;

import java.util.Scanner;

import java.util.Set;

import java.util.HashSet;

public class Main {

public static final int MIN\_NUMBER = 2;

public static final int MAX\_NUMBER = 10000;

public enum ERRORS\_LIST {

CORRECT, RANGE\_ERR, NUM\_ERR, NOT\_TXT, NOT\_EXIST, NOT\_READABLE, NOT\_WRITEABLE, FILE\_EMPTY

}

public static final String[]

ERRORS = {

"", "Значение не попадает в диапазон!", "Проверьте корректность ввода данных!", "Расширение не txt!", "Проверьте корректность ввода пути к файлу!", "Файл закрыт для чтения!", "Файл закрыт для записи!", "Файл пуст!" };

public static Scanner scanConsole = new Scanner(System.in);

public static File file;

public static void printTask() {

System.out.println("Данная программа ищет все простые числа до числа P:\n");

}

public static void printError (ERRORS\_LIST error) {

System.out.println(ERRORS[error.ordinal()] + "\nПовторите попытку: ");

}

public static ERRORS\_LIST checkArea(int num, final int MIN, final int MAX) {

ERRORS\_LIST error;

error = ERRORS\_LIST.CORRECT;

if (num < MIN || num > MAX)

error = ERRORS\_LIST.RANGE\_ERR;

return error;

}

public static int checkNum(int MIN, int MAX) {

ERRORS\_LIST error;

int num;

num = 0;

do {

error = ERRORS\_LIST.CORRECT;

try {

num = scanConsole.nextInt();

scanConsole.nextLine();

} catch (NumberFormatException e) {

error = ERRORS\_LIST.NUM\_ERR;

}

if (error != ERRORS\_LIST.CORRECT)

printError(error);

} while (error != ERRORS\_LIST.CORRECT);

return num;

}

public static boolean checkInOut() {

final int FILE\_CHOICE = 1;

final int CONSOLE\_CHOICE = 2;

int num;

boolean choose;

choose = false;

num = checkNum(FILE\_CHOICE, CONSOLE\_CHOICE);

if (num == 1)

choose = true;

return choose;

}

public static boolean chooseFileInput() {

boolean choose = true;

System.out.println("Вы хотите вводить число через файл? (Да - " + 1 + " / Нет - " + 2 + ")");

choose = checkInOut();

return choose;

}

public static String ReadPath(){

ERRORS\_LIST error;

String pathToFile;

pathToFile = " ";

scanConsole = new Scanner(System.in);

do {

error = ERRORS\_LIST.CORRECT;

System.out.print("Введите путь к файлу с расширением .txt: ");

pathToFile = scanConsole.nextLine();

if (!pathToFile.endsWith(".txt"))

error = ERRORS\_LIST.NOT\_TXT;

if (error != ERRORS\_LIST.CORRECT) {

printError(error);

}

} while (error != ERRORS\_LIST.CORRECT);

return pathToFile;

}

public static void fileReading () {

ERRORS\_LIST error;

String pathToFile;

pathToFile = "";

do {

pathToFile = ReadPath();

error = ERRORS\_LIST.CORRECT;

file = new File(pathToFile);

if (!(file.exists()))

error = ERRORS\_LIST.NOT\_EXIST;

else {

if (!file.canRead())

error = ERRORS\_LIST.NOT\_READABLE;

else {

try (Scanner fileScanner = new Scanner(file)){

if (!fileScanner.hasNext())

error = ERRORS\_LIST.FILE\_EMPTY;

} catch (FileNotFoundException e) {

printError(error);

}

}

}

if (error != ERRORS\_LIST.CORRECT)

printError(error);

} while (error != ERRORS\_LIST.CORRECT);

}

public static String fileWriting() {

ERRORS\_LIST error;

String pathToFile;

pathToFile= "";

do {

pathToFile = ReadPath();

error = ERRORS\_LIST.CORRECT;

if (!(new File(pathToFile)).exists())

error = ERRORS\_LIST.NOT\_EXIST;

else {

File file = new File(pathToFile);

if (!file.canWrite())

error = ERRORS\_LIST.NOT\_WRITEABLE;

}

if (error != ERRORS\_LIST.CORRECT)

printError(error);

} while (error != ERRORS\_LIST.CORRECT);

return pathToFile;

}

public static int readSetFile() {

int num;

ERRORS\_LIST error;

num = 0;

error = ERRORS\_LIST.CORRECT;

do {

fileReading();

try (Scanner scanFile = new Scanner(file)) {

num = scanFile.nextInt();

error = checkArea(num, MIN\_NUMBER, MAX\_NUMBER);

}

catch (Exception e) {

error = ERRORS\_LIST.NUM\_ERR;

}

if (error != ERRORS\_LIST.CORRECT)

printError(error);

} while (error != ERRORS\_LIST.CORRECT);

return num;

}

public static Set<Integer> readSet() {

boolean fromFile;

Set<Integer> numbers = new HashSet<>();

int num;

num = 0;

fromFile = chooseFileInput();

if (!fromFile) {

System.out.print("Введите число до которого вы хотетие найти простые числа [" + MIN\_NUMBER + ":" + MAX\_NUMBER + "] ");

num = checkNum(MIN\_NUMBER, MAX\_NUMBER);

}

else

num = readSetFile();

for (int i = 2; i <= num; ++i)

numbers.add(i);

return numbers;

}

public static Set<Integer> sortSet(final Set<Integer> primeNumbers) {

int prime;

int length;

prime = MIN\_NUMBER;

length = primeNumbers.size() + 1;

while (prime \* prime <= length) {

if (primeNumbers.contains(prime))

for (int i = 2 \* prime; i <= length; i += prime)

primeNumbers.remove(i);

prime++;

}

return primeNumbers;

}

public static boolean chooseFileOutput() {

boolean choose;

System.out.println("Вы хотите выводить матрицу через файл? (Да - " + 1 + " / Нет - " + 2 + ")");

choose = checkInOut();

return choose;

}

public static void printResult(Set<Integer> primeNumbers) {

String pathToFile;

boolean printToFile;

printToFile = chooseFileOutput();

pathToFile = "";

if (printToFile) {

pathToFile = fileWriting();

try (FileWriter fileWriter = new FileWriter(pathToFile, true)) {

fileWriter.write("Множество простых чисел:");

} catch (IOException e) {

};

} else {

System.out.println("Множество простых чисел:");

}

for (int num : primeNumbers) {

if (printToFile) {

try (FileWriter fileWriter = new FileWriter(pathToFile, true)) {

fileWriter.write(num + " ");

} catch (IOException e) {}

} else

System.out.print(num + " ");

}

}

public static void main(String[] args) {

Set<Integer> primeNumbers = new HashSet<>();

printTask();

primeNumbers = readSet();

primeNumbers = sortSet(primeNumbers);

printResult(primeNumbers);

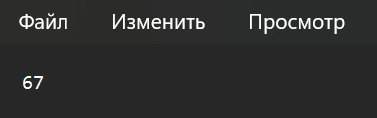
scanConsole.close();

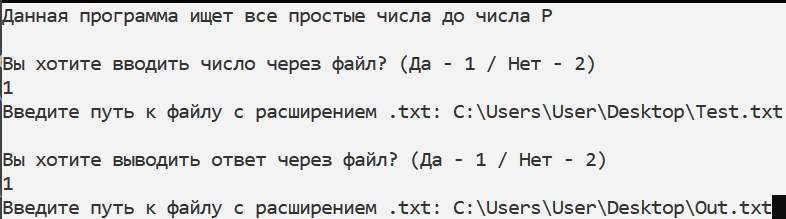
}

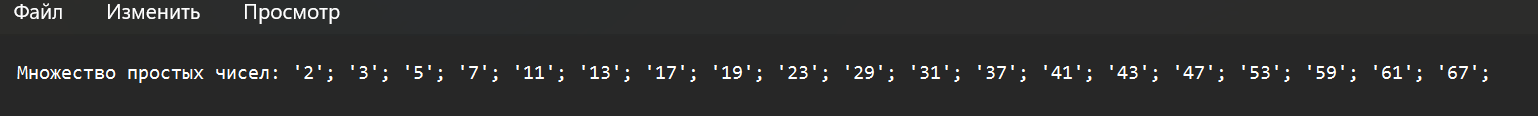
}

**Скриншоты:**

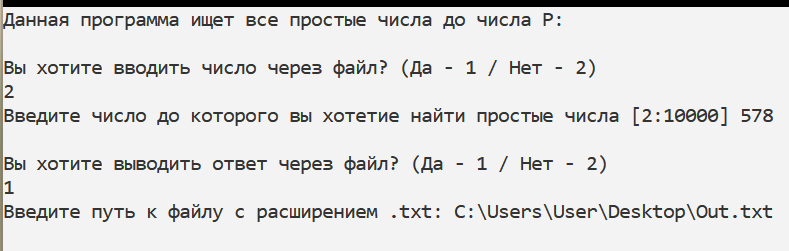
**Delphi:**

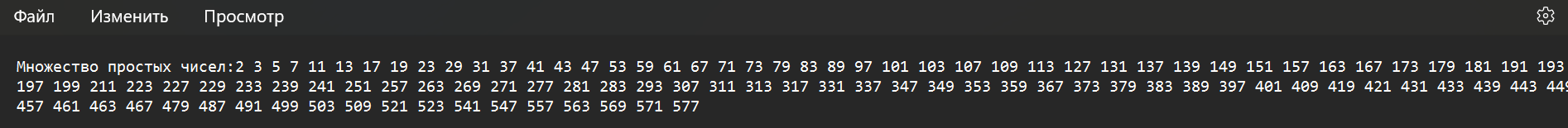
****

****

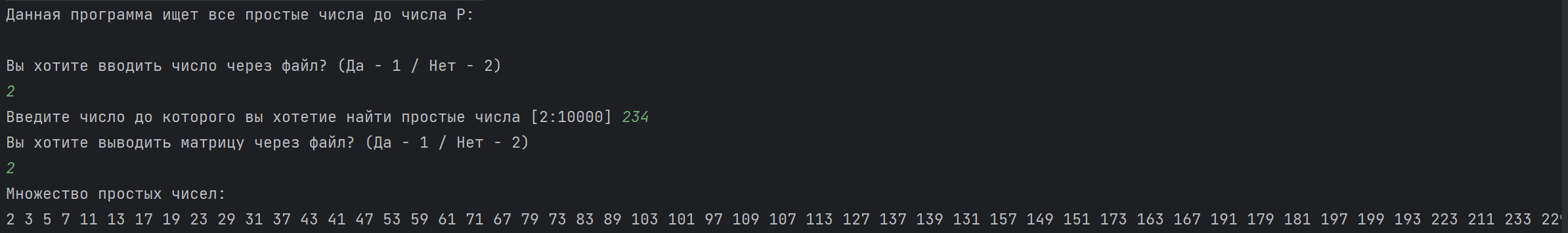
****

**C++:**

****

****

**Java:**

****

**Блок-схема:**



