**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №5**

**по дисциплине «Программирование»**

Тема: многомерные массивы

|  |  |  |
| --- | --- | --- |
| Студент гр. 1302 |  | Новиков Г.В. |
| Преподаватель |  | Калмычков В. А |

Санкт-Петербург

2021

**Оглавление**

[1. Исходная формулировка 3](#_Toc88692179)

[2. Особенности задания 3](#_Toc88692180)

[3. Контрольный пример 3](#_Toc88692181)

[4. Формальная постановка задачи 7](#_Toc88692182)

[5. Формат хранения данных 7](#_Toc88692183)

[Ограничения, обусловленные работой компьютера 8](#_Toc88692184)

[6. Макеты ввода/вывода 8](#_Toc88692185)

[7. Средства обеспечения ввода/вывода 9](#_Toc88692186)

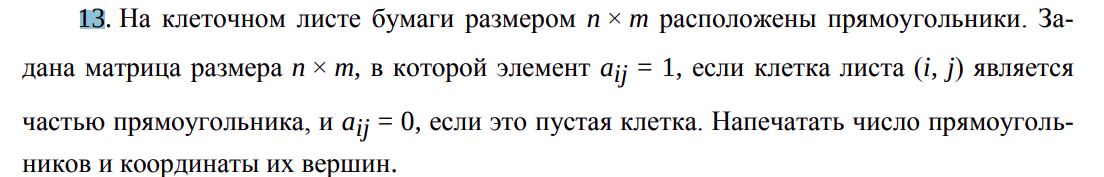
[8. Алгоритм решения 9](#_Toc88692187)

[9. Текст программы 10](#_Toc88692188)

[10. Результаты работы программы 13](#_Toc88692189)

[11. Вывод 15](#_Toc88692190)

1. Исходная формулировка

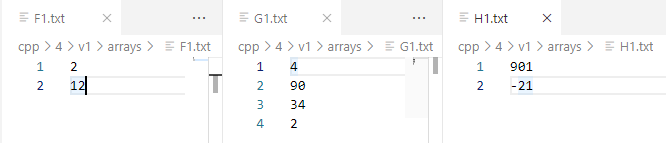


1. Особенности задания
2. Контрольный пример

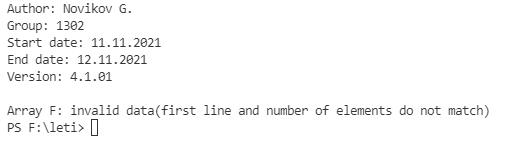
**Версия 1**

1. Все данные некорректны:

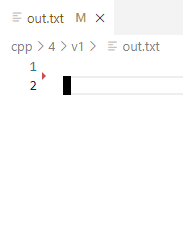
In:



Console out:

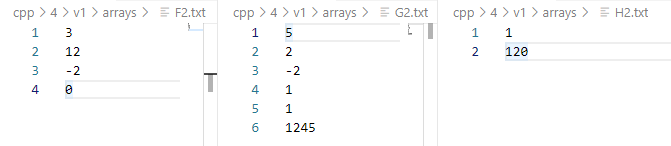


File out:

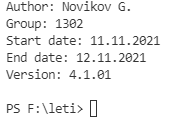


1. Все данные корректны:

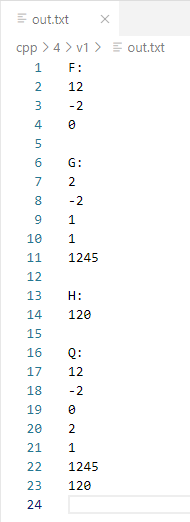
In:



Console out:

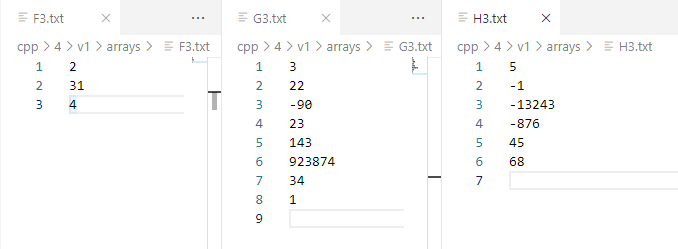


File out:

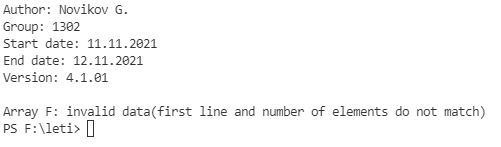


1. Часть данных некорректна:

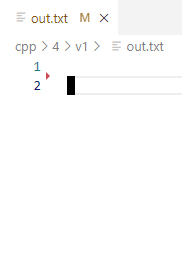
In:



Console out:



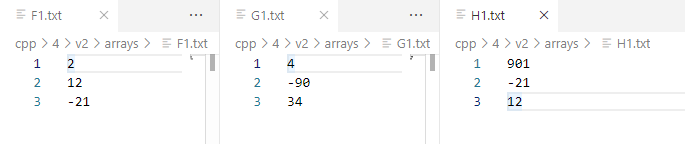
File out:



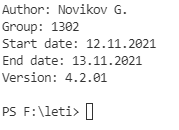
**Версия 2**

1. Все данные корректны:

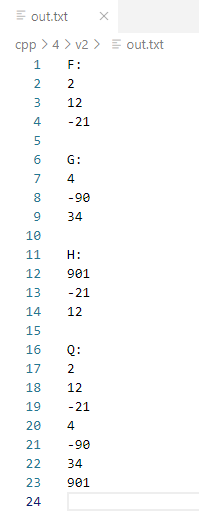
In:



Console out:

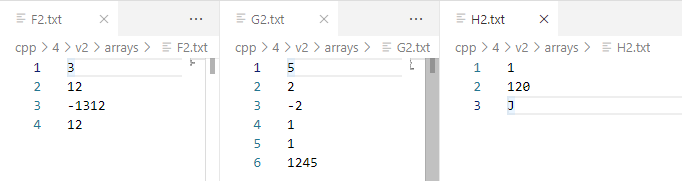


File out:

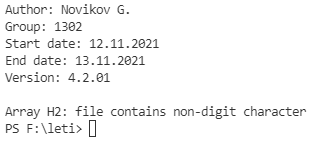


1. Часть данных некорректна:

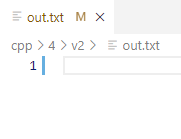
In:



Console out:



File out:



1. Формальная постановка задачи

Дано: 3 массива целых чисел.

Сформировать множество из объединения массивов.

Способ решения:

1. Сформировать массив Q из 3 введенных массивов
2. Создать массив Q\_set для множества Q
3. Пройти по Q(индекс i):
   1. Пройти по заполненной части Q\_set(индекс j)
   2. Если ни один элемент Q\_set не равен Q[i], добавить Q[i] в Q\_set
4. В итоге получаем множество Q\_set
5. Формат хранения данных

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Тип** | **Назначение** |
| test\_version | string | Переключение контрольных примеров |
| F\_size | int | Размер F |
| G\_size | int | Размер G |
| H\_size | int | Размер H |
| Q\_size | int | Размер Q |
| file | fstream | Ввод из файлов |
| line | string | Хранит считанную из файла строку |
| i | int | Итерация |
| F | int\* | Множество F |
| G | int\* | Множество G |
| H | int\* | Множество H |
| Q | int\* | Массив Q |
| Q\_set\_size | int | Размер Q\_set |
| Q\_set | int\* | Множество Q\_set |
| j | int | Итерация в двойных циклах |
| add | bool | Флаг для копирования из Q в Q\_set |
| Q\_res | int\* | Результат объединения с правильным размером |
| Q\_res\_size | int | Размер Q\_res |

1. Ограничения, обусловленные работой компьютера

Тип int ограничен промежутком: [–32768 ; 32767]

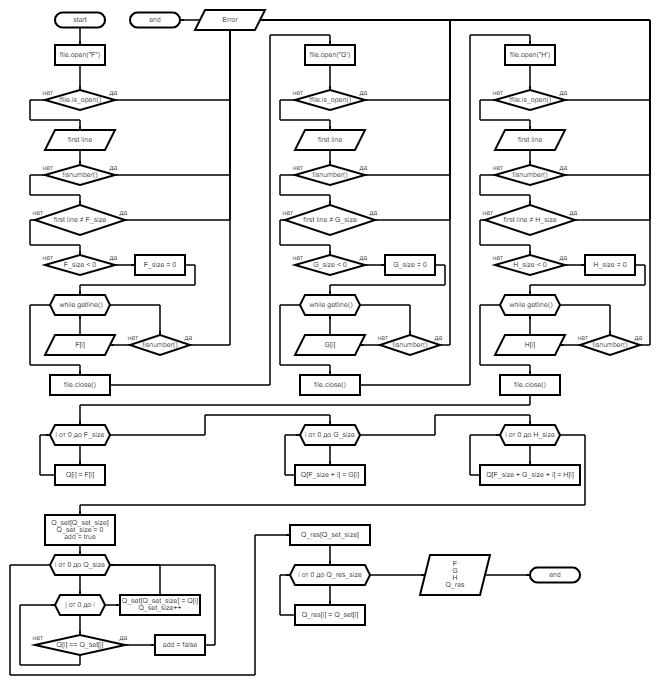
1. Макеты ввода/вывода

|  |  |  |  |
| --- | --- | --- | --- |
| № макета | Действие | Макет | Назначение |
|  | вывод | Author: Novikov G.  Group: 1302  Start date: 11.11.2021  End date: 12.11.2021  Version: 4.1.01 | Приветствие |
|  | вывод | Array <array name>: <error text> | Указание об ошибке |
|  | входной файл (1-я версия) | N  int  …  int | Файл, содержащий входные данные для первой версии |
|  | входной файл (2я версия) | int  …  int | Файл, содержащий входные данные для второй версии |
|  | Выходной файл | F:  int  …  int  G:  int  …  int  H:  int  …  Int  Q:  int  …  int | Файл, содержащий выходные данные |

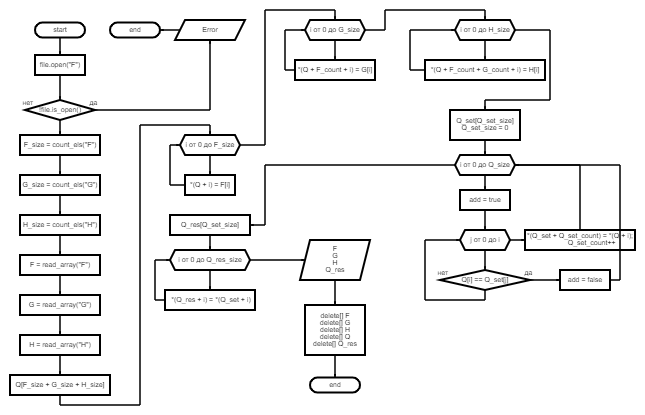
1. Средства обеспечения ввода/вывода

|  |  |
| --- | --- |
| getline() | Потоковый ввод из файла |
| cout << | Потоковый вывод в консоль |
| file << | Потоковый вывод в файл |

1. Алгоритм решения

Версия 1:

Версия 2:



1. Текст программы

Версия 1:

// Автор: Новиков Г.В.

// Группа: 1302

// Дата начала: 11.11.2021

// Дата окончания: 12.11.2021

// Версия: 4.1.01

#include <iostream>

#include <fstream>

using namespace std;

bool isnumber(string str) {

for (char c: str) {

if (!isdigit(c) and c != '-') return false;

}

return true;

}

void out(int\* array, string arrayname, int arraysize, bool clear\_file = false) {

fstream file;

if (clear\_file) {

file.open("out.txt", ios::out);

if (!file.is\_open()) {

cerr << "Error opening file" << endl;

exit(1);

}

} else {

file.open("out.txt", ios::app);

file << endl;

if (!file.is\_open()) {

cerr << "Error opening file" << endl;

exit(1);

}

}

file << arrayname << ":" << endl;

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

file << array[i] << endl;

}

file.close();

}

int main(int argc, char const \*argv[])

{

cout << "Author: Novikov G. \n"

“Group: 1302 \n"

"Start date: 11.11.2021 \n"

"End date: 12.11.2021 \n"

"Version: 4.1.01 \n" << endl;

string test\_version = "2";

int F\_size = 3;

int G\_size = 5;

int H\_size = 1;

int Q\_size;

if (F\_size > 0) Q\_size += F\_size;

if (G\_size > 0) Q\_size += G\_size;

if (H\_size > 0) Q\_size += H\_size;

int Q[Q\_size];

// F

// open file

fstream file;

file.open("arrays/F" + test\_version + ".txt", ios::in);

if (!file.is\_open()) {

cerr << "Error opening file('arrays/F" << test\_version << ".txt')" << endl;

exit(1);

}

// read size

string line;

getline(file, line);

if (!isnumber(line)) {

cerr << "Array F: file contains non-digit character" << endl;

return 1;

}

if (F\_size != atoi(line.c\_str())) {

cerr << "Array F: invalid data(first line and number of elements do not match)" << endl;

return 1;

}

if (F\_size < 0) {

F\_size = 0;

}

// read array

int F[F\_size];

int i = 0;

while (getline(file, line)) {

if (i > F\_size) {

cerr << "Array F: invalid data(first line and number of elements do not match)" << endl;

return 1;

}

if (!isnumber(line)) {

cerr << "Array F: file contains non-digit character" << endl;

return 1;

}

F[i] = atoi(line.c\_str());

i++;

}

file.close();

// G

// open file

file.open("arrays/G" + test\_version + ".txt", ios::in);

if (!file.is\_open()) {

cerr << "Error opening file('arrays/G" << test\_version << ".txt')" << endl;

exit(1);

}

// read size

getline(file, line);

if (!isnumber(line)) {

cerr << "Array G: file contains non-digit character" << endl;

return 1;

}

if (G\_size != atoi(line.c\_str())) {

cerr << "Array G: invalid data(first line and number of elements do not match)" << endl;

return 1;

}

if (G\_size < 0) {

G\_size = 0;

}

// read array

int G[G\_size];

i = 0;

while (getline(file, line)) {

if (i > G\_size) {

cerr << "Array G: invalid data(first line and number of elements do not match)" << endl;

return 1;

}

if (!isnumber(line)) {

cerr << "Array G: file contains non-digit character" << endl;

return 1;

}

G[i] = atoi(line.c\_str());

i++;

}

file.close();

// H

// open file

file.open("arrays/H" + test\_version + ".txt", ios::in);

if (!file.is\_open()) {

cerr << "Error opening file('arrays/H" << test\_version << ".txt')" << endl;

exit(1);

}

// read size

getline(file, line);

if (!isnumber(line)) {

cerr << "Array H: file contains non-digit character" << endl;

return 1;

}

if (H\_size != atoi(line.c\_str())) {

cerr << "Array H: invalid data(first line and number of elements do not match)" << endl;

return 1;

}

if (H\_size < 0) {

H\_size = 0;

}

// read array

int H[H\_size];

i = 0;

while (getline(file, line)) {

if (i > H\_size) {

cerr << "Array H: invalid data(first line and number of elements do not match)" << endl;

return 1;

}

if (!isnumber(line)) {

cerr << "Array H: file contains non-digit character" << endl;

return 1;

}

H[i] = atoi(line.c\_str());

i++;

}

file.close();

// Q

for(int i = 0; i < F\_size; i++) {

Q[i] = F[i];

}

for(int i = 0; i < G\_size; i++) {

Q[F\_size + i] = G[i];

}

for(int i = 0; i < H\_size; i++) {

Q[F\_size + G\_size + i] = H[i];

}

int Q\_set\_size = 0;

int Q\_set[Q\_size];

for(int i = 0; i < Q\_size; i++) {

bool add = true;

for(int j = 0; j < Q\_set\_size; j++) {

if (Q[i] == Q\_set[j]) {

add = false;

break;

}

}

if (add) {

Q\_set[Q\_set\_size] = Q[i];

Q\_set\_size++;

}

}

int Q\_res\_size = Q\_set\_size;

int\* Q\_res = new int[Q\_res\_size];

for (int i = 0; i < Q\_res\_size; i++) {

Q\_res[i] = Q\_set[i];

}

out(F, "F", F\_size, true);

out(G, "G", G\_size);

out(H, "H", H\_size);

out(Q\_res, "Q", Q\_res\_size);

return 0;

}

Версия 2:

// Автор: Новиков Г.В.

// Группа: 1302

// Дата начала: 11.11.2021

// Дата окончания: 12.11.2021

// Версия: 4.2.01

#include <iostream>

#include <fstream>

using namespace std;

bool isnumber(string str) {

for (char c: str) {

if (!isdigit(c) and c != '-') return false;

}

return true;

}

int count\_els(string arrayname) {

fstream file;

file.open("arrays/" + arrayname + ".txt", ios::in);

if (!file.is\_open()) {

cerr << "Error opening file('arrays/" << arrayname << ".txt')" << endl;

exit(1);

}

string line;

int array\_size = 0;

while (getline(file, line)) {

if (line.length() != 0) {

array\_size++;

}

}

return array\_size;

}

int\* read\_array(string arrayname, int array\_size) {

fstream file;

file.open("arrays/" + arrayname + ".txt", ios::in);

if (!file.is\_open()) {

cerr << "Error opening file('arrays/" << arrayname << ".txt')" << endl;

exit(1);

}

int\* array = new int[array\_size];

string el;

for (int i = 0; i < array\_size; i++) {

if (getline(file, el)) {

if (isnumber(el)) {

\*(array + i) = atoi(el.c\_str());

} else {

cerr << "Array " + arrayname + ": file contains non-digit character" << endl;

exit(1);

}

} else {

cerr << "Array " + arrayname + ": wrong size of array(" << to\_string(array\_size) << ")" << endl;

exit(1);

}

}

file.close();

return array;

}

void out(int\* array, string arrayname, int arraysize, bool clear\_file = false) {

fstream file;

if (clear\_file) {

file.open("out.txt", ios::out);

if (!file.is\_open()) {

cerr << "Error opening file" << endl;

exit(1);

}

} else {

file.open("out.txt", ios::app);

file << endl;

if (!file.is\_open()) {

cerr << "Error opening file" << endl;

exit(1);

}

}

file << arrayname << ":" << endl;

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

file << \*(array + i) << endl;

}

file.close();

}

int main(int argc, char const \*argv[])

{

cout << "Author: Novikov G. \n"

"Group: 1302 \n"

"Start date: 12.11.2021 \n"

"End date: 13.11.2021 \n"

"Version: 4.2.01 \n" << endl;

string test\_version = "2";

int F\_count = count\_els("F" + test\_version);

int G\_count = count\_els("G" + test\_version);

int H\_count = count\_els("H" + test\_version);

int\* F = read\_array("F" + test\_version, F\_count);

int\* G = read\_array("G" + test\_version, G\_count);

int\* H = read\_array("H" + test\_version, H\_count);

int Q\_count = F\_count + G\_count + H\_count;

int\* Q = new int[Q\_count];

for(int i = 0; i < F\_count; i++) {

\*(Q + i) = F[i];

}

for(int i = 0; i < G\_count; i++) {

\*(Q + F\_count + i) = G[i];

}

for(int i = 0; i < H\_count; i++) {

\*(Q + F\_count + G\_count + i) = H[i];

}

int Q\_set\_count = 0;

int Q\_set[Q\_count];

for(int i = 0; i < Q\_count; i++) {

bool add = true;

for(int j = 0; j < Q\_set\_count; j++) {

if (\*(Q + i) == \*(Q\_set + j)) {

add = false;

break;

}

}

if (add) {

\*(Q\_set + Q\_set\_count) = \*(Q + i);

Q\_set\_count++;

}

}

int\* Q\_res = new int[Q\_set\_count];

int Q\_res\_count = Q\_set\_count;

for (int i = 0; i < Q\_set\_count; i++) {

\*(Q\_res + i) = \*(Q\_set + i);

}

out(F, "F", F\_count, true);

out(G, "G", G\_count);

out(H, "H", H\_count);

out(Q\_res, "Q", Q\_res\_count);

delete[] F;

delete[] G;

delete[] H;

delete[] Q;

delete[] Q\_res;

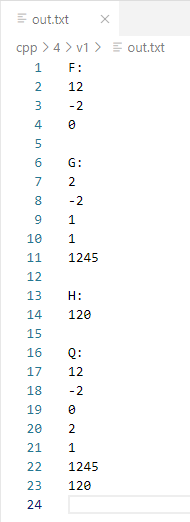
return 0;

}

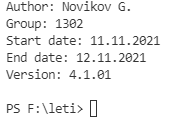
1. **Результаты работы программы**

Результатом является файл out.txt с множествами F, G, H и Q.

*out.txt:*

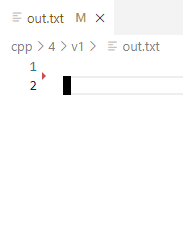


*Console:*

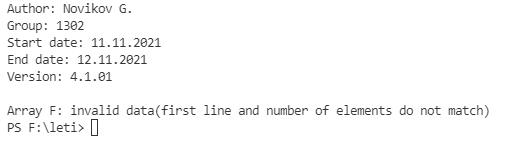
**

В случае ошибки(пример с несовпадающими размером массива и первой строкой файла):

*out.txt:*



*Console:*

**

1. Вывод

В процессе выполнения данной работы я успешно организовал работу с одномерным массивом, ввод из файла и вывод в него. Написанная мной программа выдала правильные результаты, совпавшие с контрольным примером.