**Курсова работа**

Вариант № **25**

по дисциплината

**Увод в програмирането**

разработена от студента

**Нилгюн Кязим Изет**

**Факултетен № 136011**

**Група 51A**

**Курс: 1**

**Специалност: КН**

Тип обучение: **редовно**

Съдържание

[1. Задание. 3](#_Toc375008457)

[2. Описание на програмата. 3](#_Toc375008458)

[3. Текст на програмата. 6](#_Toc375008459)

[4. Описание на техниките за доказване на коректност на програмата и резултати от тестването. 16](#_Toc375008460)

# Задание.

Да се разработят на програмен език **C++** функции за:

1. Създаване на двумерен масив с елементи от: „**а – цял; b – низ; c – реален; d – низ**“.
2. Разпечатване на двумерния масив.
3. Прехвърляне на елементите на двумерния масив в едномерен масив, като прехвърлянето става по: „**На елементите, които съдържат поне едно целочислено поле от цял тип с нечетна стойност**“.
4. Разпечатване на едномерния масив.
5. Разпечатване на всички елементи на едномерния масив, които отговарят на: „**се отпечата средно аритметичното на сумата от** *полета а и с* **за тези елементи, в които на трета позиция на низа от** *поле b* **стои символа ‘&’**“.
6. Вмъкване на посочен от потребителя брой елементи, с посочени от потребителя стойности от първия елемент, който е: „**след дванадесети елемент, ако в 22 елемент има низ с името Петров**“.
7. Изтриване на посочен от потребителя брой елементи от първия елемент, който е: „**след дванадесети елемент, ако в 22 елемент има низ с името Петров**“.
8. Сортиране на едномерния масив в „**низходяща посока**“.

Да се разработи главна програма, викаща горните функции, която представлява меню с имената на функциите и дава възможност на потребителя да избира на всяка стъпка коя да е следващата изпълнявана функция.

# Описание на програмата.

*Програмата е написана на MS Visual Studio Professional 2013.*

* **Някои по-важни променливи:**
  + myArrItem \*oneDimArr; - основният едномерен масив;
  + myArrItem \*\*twoDimArr; - двумерен масив;
  + int rows; - променлива за броя на редовете на двумерния масив;
  + int cols; - променлива за броя на колоните на двумерния масив;
  + int sizeOneDimArr; - променлива за броя на елементите (размера на) в едномерния масив;
  + struct myArrItem{…}; - структура с 4 полета: int a, string b и d, float c;
* **void main()**
  + Главна функция (програма);
  + Служи за извикване на желаните от потребителя функции. Избират се от меню, чрез въвеждане на число съответстващо на желаната функция.
* **void CreateTwoDimArray()**
  + Функция за създаване на двумерен масив;
  + Потребителят избира броя на редовете и колоните на масива. Ако потребителят въведете некоректна стойност за броя на редовете/колоните, т.е 0 или отрицателна стойност – извежда се съобщение за грешка. Ако при заделянето на памет за масива, наличната памет не е достатъчна също се извежда съобщение за това. Не е направено проверка дали въведеното число ще се побере в типа int. Ако всичко е „наред“ – двумерният масив е създаден.
* **void ReadTwoDimArr()**
  + Функция за въвеждане на елементите на двумерния масив;
  + Двумерният масив се обхожда с помощта на 2 вложени for – цикли. Външният е за обхождане на редовете, а вътрешният – на колоните.
  + Потребителят въвежда от конзолата всички полета на съответния елемент. Като е указано каква стойност се очаква да се въведе. Не е направено проверка за коректност на входните данни, т.е трябва да се внимава при въвеждането на данните. Например не трябва да се въвежда нечислов символ в числови променливи като int и float и т.н.
* **void PrintArrat(int choiceDim)** 
  + Функция за разпечатване на едномерен/двумерен масив;
  + Параметър – int choiceDim: ако стойността на този параметър е 1 или ONE\_DIM, функцията служи за отпечатване на едномерен масив. Ако стойността е 2 или TWO\_DIM – служи за отпечатване на двумерен масив. Ако няма стойност 1 или 2 на конзолата се отпечатва съответното съобщение за грешка.
* **void TransferToOneDimArr()** 
  + Функция за прехвърляне на елементите на двумерния масив, които имат поне едно целочислено поле от цял тип с нечетна стойност, в едномерен;
  + Т.к размерът на едномерния масив не е известен, първо се създава един временен (помощен) едномерен масив (bufferArray) с размер - броят на всички елементи в двумерния масив. С 2 вложени цикъла се обхожда двумерният масив и се проверява стойността на целочисленото поле (а) на всеки елемент дали е нечетна. Ако е така този елемент се добавя в помощния масив. След като обхождането завърши ако има поне 1 прехвърлен елемент се създава основният едномерен масив с толкова на брой елементи, колкото са необходими (int index – използва и като брояч). С един цикъл се копират всички елементи на временния масив в основния и след това временният масив се „унищожава“.
* **void PrintAverageOfAC()**
  + Функция за извеждане средно аритметичното на сумата на двете полета a и c, ако на позиция 3 на полето b има символа ‘&’.
  + С помощта на един цикъл for се обхожда едномерният масив и на всеки елемент се проверява дали на позиция 3 на полето b има ‘&’. Важното тук, което трябва да се забележи е, че полето b може да е с по-малка дължина и в такъв случай ако не е предвиден това програмата ще изхвърли изключение. За да се предотврати това е добавено още едно условие, което проверява дължината на низа b. В случай, че няма елементи, които отговарят на горното условие – извежда се съобщение, че няма елементи, които отговарят на това условие.
* **void InsertDeleteItems(int insOrDel)**
  + Функция за вмъкване/изтриване на елементи в/от едномерния масив.
  + Параметър – int insOrDel: ако стойността на insOrDel е 3 или INS функцията служи за вмъкване на елементи в масива. Ако insOrDel е 4 или DEL - служи за изтриване на елементи от масива.
  + Тази функция се изпълнява само ако има поне 22 елемента в масива и 22-ят елемент съдържа низ с име „Петров“, т.е полето b и/или d е равен на „Петров“.
  + Т.к размерите на масивите са статични, т.е не може да се променя – използва временен (помощен) масив.
    - Вмъкване: Размерът на помощния масив е = размерът на основния масив + броя на елементите, които ще се вмъкват. Първо – копират се първите 12 елемента на основния масив в помощния. Второ – добавят се елементите, зададени от потребителя – след 12-ия. Трето – копират се останалите елементи от основния масив в помощния. Изтрива се основният масив. Създава се пак, обаче с размер – размера на помощния масив. Копират се всички елементи от помощния в основния масив и се изтрива временният масив.
    - Изтриване: Размерът на помощния масив е = размерът на основния масив – броя на елементите за изтриване. Първо – копират се първите 12 елемента на основния масив в помощния. Второ – копират се останалите елементи от основния масив като се пропускат тези елементи, които трябва да бъдат изтрити. Изтрива се основният масив. Създава се пак, обаче с размер – размера на помощния масив. Копират се всички елементи от помощния в основния масив и се изтрива временният масив.
  + Броят на елементите за вмъкване/изтриване трябва да е 0 или положително число. При грешна стойност – извежда се съобщение затова.
* **void ShellSort(myArrItem x[], int n)**
  + Функция за сортиране на едномерния масив в низходящ ред по метода на Шел.
  + Параметри – myArrItem x[] – масивът за сортиране; int n – размер на масива.
  + Масивът се сортира по стойността на полето с.
  + Сортираме по отделно групи елементи, които са на разстояние h позиции един от друг. За стъпката h се препоръчва да се използват в намаляващ ред всички членове на редицата 1, 4, 13, 40, 121, 364, . . . , които са по-малки от някаква гранична стойност, зависеща от n – размера на масива. Нужната стойност на h може да бъде определена чрез следния цикъл:

h = 0;

while (2\*(3\*h+1) <= n) h = 3\*h+1;

Този метод се нарича още сортиране чрез вмъкване с намаляваща стъпка, тъй като h постепенно намалява. Той се основава на идеята, че първоначално, когато h е голямо, всяка редица, която сортираме, е малка. По-късно, когато h намалее, редиците стават по-големи, но елементите вече са частично подредени и не се налагат големи размествания.

* **void FreeArray()**
  + Функция за освобождаване на паметта, заета от масива/ите oneDimArr и/или twoDimArr.

# Текст на програмата.

#include <iostream>

#include <string>

using namespace std;

#define ONE\_DIM 1

#define TWO\_DIM 2

#define INS 3

#define DEL 4

struct myArrItem

{

int a; // поле 1 с име а

string b; // поле 2 с име b

float c; // поле 3 с име c

string d; // поле 4 с име d

};

myArrItem \*oneDimArr; // едномерене масив

myArrItem \*\*twoDimArr; // двумерен масив

int rows; // брой редове на двумерния масив

int cols; // брой колони на двумерния масив

int sizeOneDimArr; // размер на едномерния масив

// функция за извеждане на масив (двумерен / едномерен)

void PrintArray(int choiceDim)

{

// Ако е избран ЕДНОмерният масив за печат

if (choiceDim == ONE\_DIM)

{

for (int index = 0; index < sizeOneDimArr; index++)

{

cout << endl << "Item [" << index + 1 << "]" << endl;

cout << oneDimArr[index].a << endl;

cout << oneDimArr[index].b << endl;

cout << oneDimArr[index].c << endl;

cout << oneDimArr[index].d << endl;

}

}

// Ако е избран ДВУмерният масив за печат

else if (choiceDim == TWO\_DIM)

{

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

{

for (int col = 0; col < cols; col++)

{

cout << endl << "Item [" << row + 1 << ", " << col + 1 << "]" << endl;

cout << twoDimArr[row][col].a << endl;

cout << twoDimArr[row][col].b << endl;

cout << twoDimArr[row][col].c << endl;

cout << twoDimArr[row][col].d << endl;

}

}

}

// Грешен избор

else

{

cout << endl << "Wrong dimension of the array!" << endl;

}

}

// функция за въвеждане на елементите на матрицата

void ReadTwoDimArr()

{

// обхождане по редове

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

{

// обхождане по колони

for (int col = 0; col < cols; col++)

{

cout << endl << "Item [" << row + 1 << ", " << col + 1 << "]" << endl;

cout << "Please enter an integer number (a): ";

cin >> twoDimArr[row][col].a;

cin.sync();

cout << "Please enter a string (b): ";

getline(cin, twoDimArr[row][col].b);

cout << "Please enter a floating point number (c): ";

cin >> twoDimArr[row][col].c;

cin.sync();

cout << "Please enter a string (d): ";

getline(cin, twoDimArr[row][col].d);

}

}

}

// функция за създаване на двумерен масив

void CreateТwoDimArray()

{

// въвеждане размера на матрицата (двумерния масив)

cout << "Please enter the number of the array's rows: ";

cin >> rows;

// Ако числото е коректно, т.е положително

if (rows > 0)

{

// заделяне на памет за редовете

twoDimArr = new myArrItem \*[rows];

// проверка дали има достатъчно памет

if (twoDimArr == nullptr)

{

cout << endl << "Error: memory could not be allocated!" << endl;

system("pause");

exit(1);

}

else

{

cout << "Please enter the number of the array's columns: ";

cin >> cols;

if (cols > 0)

{

// заделяне на памет за колоните

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

{

twoDimArr[index] = new myArrItem[cols];

if (twoDimArr[index] == nullptr)

{

cout << endl << "Error: memory could not be allocated!" << endl;

system("pause");

exit(1);

}

}

}

else

{

cout << endl << "Error: the number of the array's columns can't"

<< " be negative or zero!" << endl;

}

}

}

else

{

cout << endl << "Error: the number of the array's rows can't"

<< " be negative or zero!" << endl;

}

}

// функция за осовобождаване на паметта, заета

// от двата основни масива:

// двумерния и едномерния

void FreeArrays()

{

// освобождаване на паметта, заета от колоните на двумерния масив

if (cols > 0)

{

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

{

delete[] twoDimArr[index];

}

cols = 0;

}

// освобождаване на паметта, заета от редовете на двумерния масив

if (rows > 0)

{

delete[] twoDimArr;

rows = 0;

}

// освобождаване на паметта, заета от одномерния масив

if (sizeOneDimArr > 0)

{

delete[] oneDimArr;

sizeOneDimArr = 0;

}

}

// функция за прехвърляне на елементите с нечетна стойност

// на полето 'а' на двумерния масив в едномерен масив

void TransferToOneDimArr()

{

// размерът на едномерния масива в "най-лошия" случай

// може да е rows \* cols, в случай че стойността на полето

// 'а' на всички елементи на двумерния масив е нечетна.

sizeOneDimArr = rows \* cols;

// създаване на временен едномерен масив с максимално възможен размер

myArrItem \*bufferArray = new myArrItem[sizeOneDimArr];

int index = 0; // индекс на едномерния масив (брояч)

// прехвърляне на елементите, отговарящи на зададеното условие

// във временния масив

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

{

for (int col = 0; col < cols; col++)

{

if (twoDimArr[row][col].a % 2 != 0)

{

bufferArray[index].a = twoDimArr[row][col].a;

bufferArray[index].b = twoDimArr[row][col].b;

bufferArray[index].c = twoDimArr[row][col].c;

bufferArray[index].d = twoDimArr[row][col].d;

index++;

}

}

}

// Ако има поне 1 прехвърлен елемент

if (index > 0)

{

sizeOneDimArr = index;

// създаване на едномерен масив с нужния брой елементи

oneDimArr = new myArrItem[sizeOneDimArr];

// копиране на елементите в масива

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

{

oneDimArr[i].a = bufferArray[i].a;

oneDimArr[i].b = bufferArray[i].b;

oneDimArr[i].c = bufferArray[i].c;

oneDimArr[i].d = bufferArray[i].d;

}

cout << endl << "Ready!" << endl;

}

else

{

cout << endl << "One-dimensional array is empty!" << endl;

sizeOneDimArr = 0;

}

// освобождаване на паметта, заета от временния масив

delete[] bufferArray;

}

// функция за печатане средно-аритметичното на а и с

// ако на 3 позиция на b има символа '&'

void PrintAverageOfAC()

{

// средно-аритметично

double average = 0;

// сума

double sum = 0;

// Булева променлива за проверка дали има поне 1

// елемент, отговарящ на зададеното условие

bool flag = false;

for (int index = 0; index < sizeOneDimArr; index++)

{

// Ако низът е с дължина >= 4 и на 3-та позиция има символа '&'

// низът може да е с дължина < 4 - затова се прави тази проверка!

if (oneDimArr[index].b.length() >= 4 && oneDimArr[index].b[3] == '&')

{

flag = true;

sum += oneDimArr[index].a;

sum += oneDimArr[index].c;

// изчисляване на средно-аритметичното

average = sum / 2.0;

printf("\n\rItem [%d] Average: %.2f", index + 1, average);

}

sum = 0;

}

if (!flag)

{

cout << endl << "No items matching the condition!" << endl;

}

}

void InsertDeleteItems(int insOrDel)

{

int numElements;

string msg = "How many elements you want to insert? ";

if (insOrDel == DEL)

msg = "How many elements you want to delete? ";

cout << endl << msg;

cin >> numElements;

// ако е въведено коректно число, т.е 0 или по-голямо

if (numElements > -1)

{

// ако има поне 22 елемента и 22-ят елемент има низ

// с име "Петров" / "Petrov"

if (sizeOneDimArr >= 22 &&

(oneDimArr[21].b == "Petrov" ||

oneDimArr[21].d == "Petrov" ||

oneDimArr[21].b == "Петров" ||

oneDimArr[21].d == "Петров"))

{

// заделяне на памет за новия масив

int size;

// ако е избрана операцията вмъкване

if (insOrDel == INS)

{

size = sizeOneDimArr + numElements;

}

// ако е избрана операцията изтриване и има достатъчно

// на брой елементи за изтриване

else if (insOrDel == DEL && (sizeOneDimArr - 12 > numElements))

{

size = sizeOneDimArr - numElements;

}

// грешен избор или няма достатъчно на брой

// елементи за изтриване

else

{

cout << "\n\rNot as many items for deletion "

<< "as you set!" << endl;

system("pause");

return;

}

// заделяне на памет

myArrItem \*bufferArray = new myArrItem[size];

// копиране на първите 12 елемента от основния

// масив във временния масив

for (int index = 0; index < 12; index++)

{

bufferArray[index].a = oneDimArr[index].a;

bufferArray[index].b = oneDimArr[index].b;

bufferArray[index].c = oneDimArr[index].c;

bufferArray[index].d = oneDimArr[index].d;

}

if (insOrDel == INS)

{

// въвеждане на numElements на брой елемента

// от клавиатурата във временния масив

for (int index = 12; index < 12 + numElements; index++)

{

cout << endl << "Item [" << index - 11 << "]" << endl;

cout << "Please enter an integer number (a): ";

cin >> bufferArray[index].a;

cin.sync();

cout << "Please enter a string (b): ";

getline(cin, bufferArray[index].b);

cout << "Please enter a floating point number (c): ";

cin >> bufferArray[index].c;

cin.sync();

cout << "Please enter a string (d): ";

getline(cin, bufferArray[index].d);

}

// копиране на всички елементи след 12-ия

// от основния във временния масив

for (int index = 12; index < sizeOneDimArr; index++)

{

bufferArray[index + numElements].a = oneDimArr[index].a;

bufferArray[index + numElements].b = oneDimArr[index].b;

bufferArray[index + numElements].c = oneDimArr[index].c;

bufferArray[index + numElements].d = oneDimArr[index].d;

}

cout << endl << "Inserted!" << endl;

}

else if (insOrDel == DEL)

{

// пропускане numElements на брой елемента

// след 12-ия и копиране на следващите

for (int index = 12 + numElements; index < sizeOneDimArr; index++)

{

bufferArray[index - numElements].a = oneDimArr[index].a;

bufferArray[index - numElements].b = oneDimArr[index].b;

bufferArray[index - numElements].c = oneDimArr[index].c;

bufferArray[index - numElements].d = oneDimArr[index].d;

}

cout << endl << "Deleted!" << endl;

}

sizeOneDimArr = size;

// освобождаване на паметта, заета от основния масив

delete[] oneDimArr;

// заделяне на памет за основния масив

// (колкото е необходимо)

oneDimArr = new myArrItem[sizeOneDimArr];

// копиране на елементите от временния масив

// в основния

for (int index = 0; index < sizeOneDimArr; index++)

{

oneDimArr[index].a = bufferArray[index].a;

oneDimArr[index].b = bufferArray[index].b;

oneDimArr[index].c = bufferArray[index].c;

oneDimArr[index].d = bufferArray[index].d;

}

// освобождаване на паметта, заета от

// временния масив

delete[] bufferArray;

}

else

{

// ако няма 22 или повече елемента

if (sizeOneDimArr < 22)

{

cout << endl << "Not a sufficient number (22) of "

<< "elements in the array!" << endl;

}

// ако има 22 или повече елемента, но 22-ят елемент

// няма низ с име "Петров" / "Petrov"

else

{

cout << endl << "b and d of the 22nd element is "

<< "not equal to \"Petrov\"" << endl;

}

}

}

// ако е въведено отрицателно число

else

{

cout << endl << "The number of elements must be 0 "

<< "or greater!" << endl;

}

}

// функция за сортиране на едномерен масив

// в низходящ ред по метода на Шел.

// Масивът се сортира по стойността на

// полето с

void ShellSort(myArrItem x[], int n)

{

int h = 0, i, j, next;

myArrItem y;

//Намиране начална (най-голяма) стъпка

while (2 \* (next = 3 \* h + 1) <= n)

h = next;

for (; h > 0; h /= 3)

{

for (i = h; i < n; i++) {

y.a = x[i].a;

y.b = x[i].b;

y.c = x[i].c;

y.d = x[i].d;

j = i - h;

while (j >= 0 && y.c > x[j].c)

{

x[j + h].a = x[j].a;

x[j + h].b = x[j].b;

x[j + h].c = x[j].c;

x[j + h].d = x[j].d;

j -= h;

}

x[j + h].a = y.a;

x[j + h].b = y.b;

x[j + h].c = y.c;

x[j + h].d = y.d;

}

}

}

void main()

{

int choice;

bool isSelectedCreate = false;

bool isSelectedTransfer = false;

do

{

printf("\n\r");

printf("%18s--------------------MENU------------------\n\r", "");

printf("%25s1. Create Two-Dimensional Array\n\r", "");

printf("%25s2. Print Two-Dimensional Array\n\r", "");

printf("%25s3. Transfer\n\r", "");

printf("%25s4. Print One-Dimensional Array\n\r", "");

printf("%25s5. Print - Condition 1\n\r", "");

printf("%25s6. Insert items\n\r", "");

printf("%25s7. Delete items\n\r", "");

printf("%25s8. Sort Array\n\r", "");

printf("%25s9. Delete Arrays\n\r", "");

printf("%25s10. Clear - Console\n\r", "");

printf("%21sChoice operation or enter 0 for exit\n\r", "");

printf("%18s------------------------------------------\n\r", "");

printf("%18s", "");

cin >> choice;

getchar();

switch (choice)

{

/\*\*\*\*\*/ case 1:

// ако вече е избрана тази операция, преди да се

// изпълни отново е необходимо да се изтрият

// масивите (да се освободи паметта)

if (!isSelectedCreate)

{

isSelectedCreate = true;

CreateТwoDimArray();

ReadTwoDimArr();

}

else

{

cout << endl << "First you must to delete "

"the arrays! (choice 9)" << endl;

}

break;

/\*\*\*\*\*/ case 2:

PrintArray(TWO\_DIM);

break;

/\*\*\*\*\*/ case 3:

// ако вече е избрана тази операция, преди да се

// изпълни отново е необходимо да се изтрият

// масивите (да се освободи паметта)

if (!isSelectedTransfer)

{

TransferToOneDimArr();

isSelectedTransfer = true;

}

else

{

cout << endl << "First you must to delete "

"the arrays! (choice 9)" << endl;

}

break;

/\*\*\*\*\*/ case 4:

PrintArray(ONE\_DIM);

break;

/\*\*\*\*\*/ case 5:

PrintAverageOfAC();

break;

/\*\*\*\*\*/ case 6:

InsertDeleteItems(INS);

break;

/\*\*\*\*\*/ case 7:

InsertDeleteItems(DEL);

break;

/\*\*\*\*\*/ case 8:

ShellSort(oneDimArr, sizeOneDimArr);

break;

/\*\*\*\*\*/ case 9:

FreeArrays();

isSelectedCreate = false;

isSelectedTransfer = false;

break;

/\*\*\*\*\*/ case 10:

// изчистване на конзолата

system("cls");

break;

} // end swich

} // end do-while

while (choice != 0);

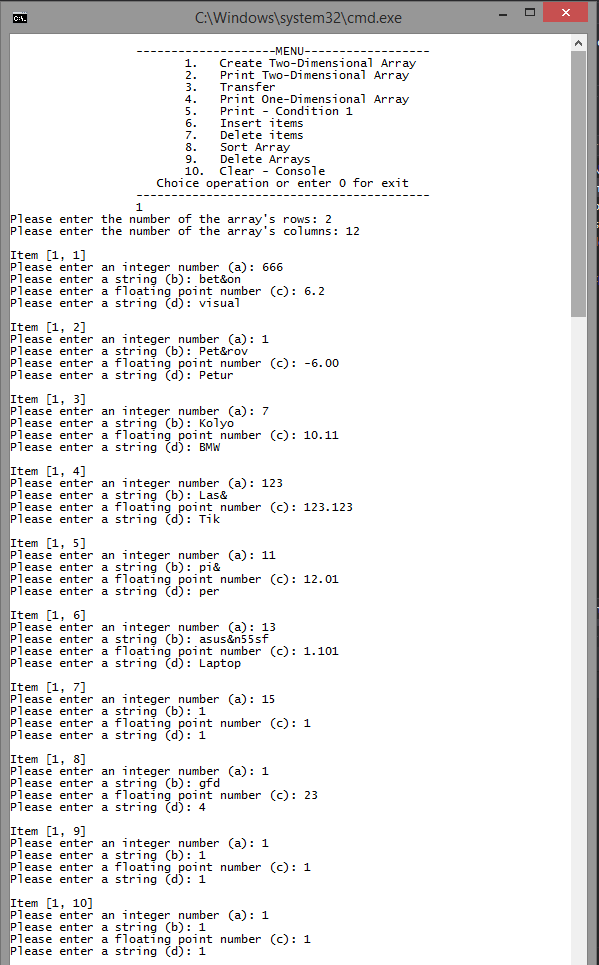
FreeArrays();

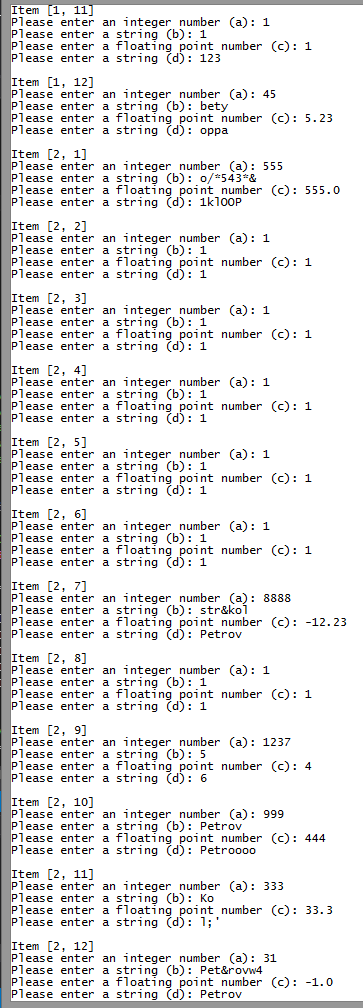
system("pause");

}

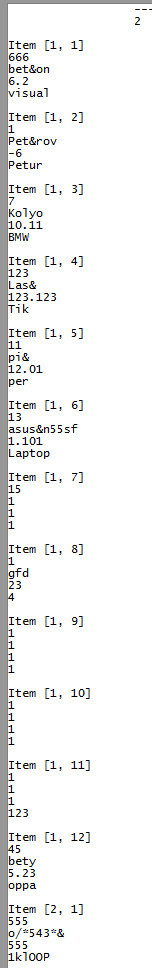
# Описание на техниките за доказване на коректност на програмата и резултати от тестването.

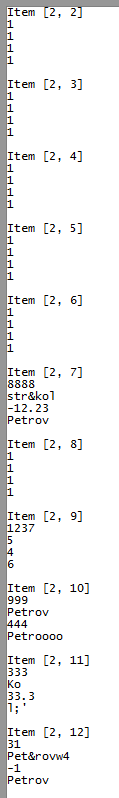
Операция 1 - Създаване на двумерния масив и въвеждане на елементите му!



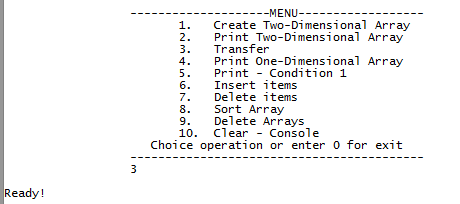


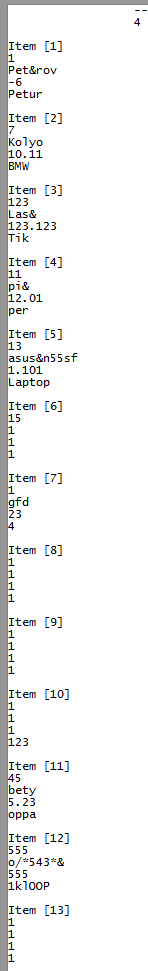
Операция 2 – Разпечатване на двумерния масив

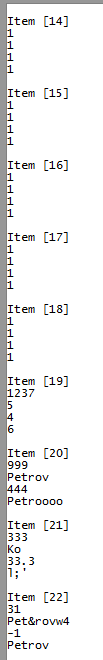




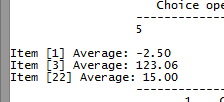
Операция 3 и 4– Прехвърляне на елементите в едномерен масив и извеждане на масива



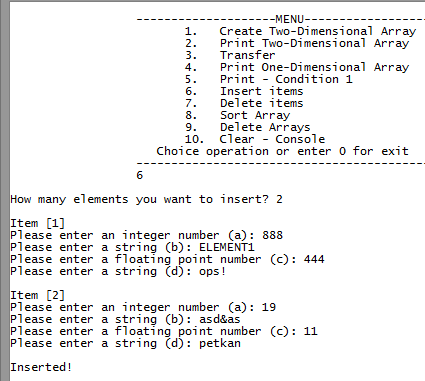


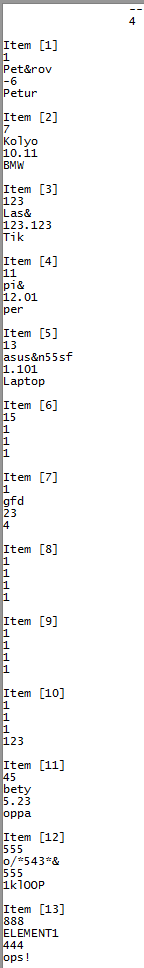
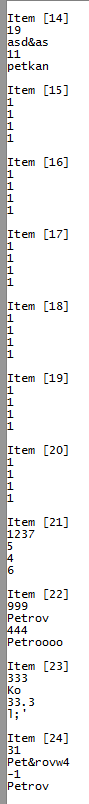


Операция 5 – разпечатване на средно аритметичното на сумата на полетата a и c, ако на 3-та позиция на полета b има символа ‘&’

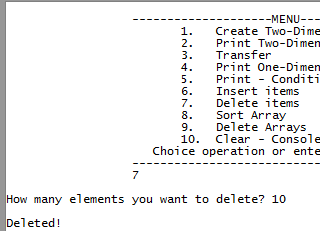


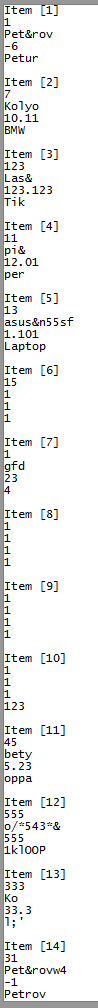
Операция 6 и 4 – Вмъкване на елементи в масива и извеждане на масива



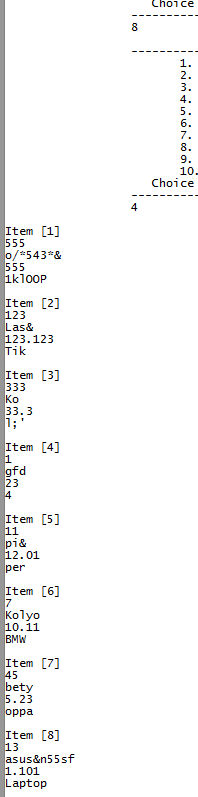
 

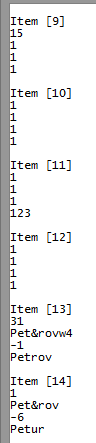
Операция 7 и 4 – Изтриване на елементи и извеждане на масива





Операция 8 и 4 – сортиране и извеждане на масива





Първият проблем се появи при въвеждането на отрицателна/нулева стойност за броя на редовете/колоните на двумерният масив. Решение на проблема бе да се добави проверка за стойностите на тези променливи.

Вторият проблем се появи при въвеждането на полетата на елементите. Т.к за въвеждане и извеждане на данните се използват потоци (cin, cout) при въвеждането на низ след число имаше проблем. Проблемът е решен като се използва cin.sync(), който синхронизира потока от данни.

Третият проблем се появи, когато бе забравено проверката за дължината на низа, където се проверява 4-ият символ (символът на 3-та позиция) от низа b дали е ‘&’. Без тази проверка ако дължината на низа е по-малко от 4 символа – програмата изхвърля изключение OutOfRange.