Масиви

Базово програмиране
Лекция 6
Милена Карова
кат. КНТ ТУ-Варна

Въведение в масивите (array)

- Определение за масив
- Използване на [ ], име на масив
- Деклариране на масива

```
int score[5];
```

- Елементи на масива

```
score[0], score[1], score[2], score[3], score[4]
```

- Индексирана променлива
- Индекс
- Размер на масива
- Тип на масива

```
int next, score[5], max;
int n = 2;
score[n + 1] = 99;
```

- Използването на цикъла for е удобно
- Използвайте именувана константа за размер на масива

```
const int NUMBER_OF_STUDENTS = 5;
int i, score[NUMBER_OF_STUDENTS], max;
```

Примерна програма за използване на масив

```
#include <iostream>

int main( )
{
    using namespace std;
    int i, score[5], max;

    cout << "Enter 5 scores:\n";
    cin >> score[0];
    max = score[0];
    for (i = 1; i < 5; i++)
    {
        cin >> score[i];
        if (score[i] > max)
            max = score[i];
        //max is the largest of the values score[0],..., score[i].
    }

    cout << "The highest score is " << max << endl
         << "The scores and their\n"
         << "differences from the highest are:\n";
    for (i = 0; i < 5; i++)
        cout << score[i] << " off by "
             << (max - score[i]) << endl;

    return 0;
}
```

```
Enter 5 scores:
5 9 2 10 6
The highest score is 10
The scores and their
differences from the highest are:
5 off by 5
9 off by 1
2 off by 8
10 off by 0
6 off by 4
```

```
cout << "Enter number of students:\n";
cin >> number;
int score[number]; //ILLEGAL ON MANY COMPILERS!
```

Това не може да се използва

Масивите в паметта

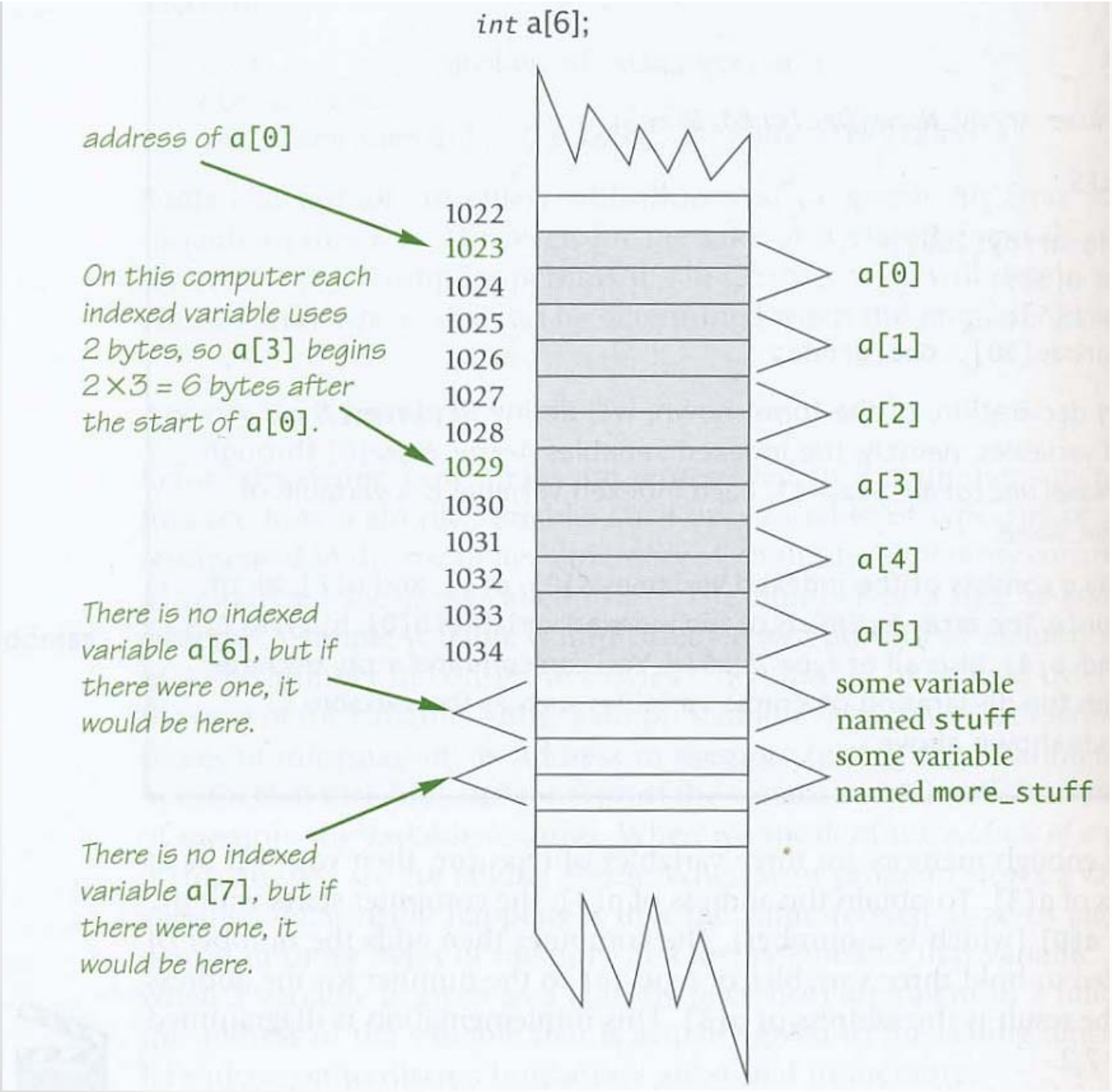
- Получаване адреса на всеки елемент на масива, ако се знае адреса на първия елемент на масива
- Съобщение за грешка, ако се използва елемент на масива с неоторизиран индекс
- Начална инициализация на масивите

```
int children[3] = {2, 12, 1};
```

```
int children[3];
children[0] = 2;
children[1] = 12;
children[2] = 1;
```

```
int b[] = {5, 12, 11};
```

```
int b[3] = {5, 12, 11};
```


int a[6];
address of a[0]
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
On this computer each indexed variable uses 2 bytes, so a[3] begins 2 × 3 = 6 bytes after the start of a[0].
There is no indexed variable a[6], but if there were one, it would be here.
There is no indexed variable a[7], but if there were one, it would be here.
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
some variable named stuff
some variable named more_stuff

Масиви и функции

- Като параметри на функция могат да се използват както елементи на масива така и целите масиви

```
my_function(n);    my_function(a[3]);    my_function(a[i]);
```

```
#include <iostream>

const int NUMBER_OF_EMPLOYEES = 3;

int adjust_days(int old_days);
//Returns old_days plus 5.

int main( )
{
    using namespace std;
    int vacation[NUMBER_OF_EMPLOYEES], number;

    cout << "Enter allowed vacation days for employees 1"
         << " through " << NUMBER_OF_EMPLOYEES << ":\n";
    for (number = 1; number <= NUMBER_OF_EMPLOYEES; number++)
        cin >> vacation[number-1];

    for (number = 0; number < NUMBER_OF_EMPLOYEES; number++)
        vacation[number] = adjust_days(vacation[number]);

    cout << "The revised number of vacation days are:\n";
    for (number = 1; number <= NUMBER_OF_EMPLOYEES; number++)
        cout << "Employee number " << number
             << " vacation days = " << vacation[number-1] << endl;

    return 0;
}

int adjust_days(int old_days)
{
    return (old_days + 5);
}
```

```
Enter allowed vacation days for employees 1 through 3:
10 20 5
The revised number of vacation days are:
Employee number 1 vacation days = 15
Employee number 2 vacation days = 25
Employee number 3 vacation days = 10
```

Име на масив като аргумент

- Всеки масив се характеризира с 3 основни неща:
- адрес на масива, тип на елементите на масива и размер на масива
- Ако масивът се подава като параметър, то се подава само адреса на първия елемент на масива (което представлява самото име на масива) и размера на масива
- В дадения пример, една и съща функция fill_up може да попълва различни масиви, с различен брой елементи

```
int score[5], time[10];
```

```
fill_up(score, 5);
fill_up(time, 10);
```

Параметър const (constant array parameter)

```
void show_the_world(int a[], int size_of_a)
{
    cout << "The array contains the following values:\n";
    for (int i = 0; i < size_of_a; i++)
        cout << a[i] << " ";
    cout << endl;
}
```

```
void show_the_world(const int a[], int size_of_a)
{
    cout << "The array contains the following values:\n";
    for (int i = 0; i < size_of_a; a[i]++)
        cout << a[i] << " ";
    cout << endl;
}
```

Mistake, but the compiler will not catch it unless you use the **const** *modifier.*

Програмиране с масиви

```
//Shows the difference between each of a list of golf scores and their average.
#include <iostream>
const int MAX_NUMBER_SCORES = 10;

void fill_array(int a[], int size, int& number_used);

double compute_average(const int a[], int number_used);
void show_difference(const int a[], int number_used);

int main( )
{
    using namespace std;
    int score[MAX_NUMBER_SCORES], number_used;

    cout << "This program reads golf scores and shows\n"
         << "how much each differs from the average.\n";

    cout << "Enter golf scores:\n";
    fill_array(score, MAX_NUMBER_SCORES, number_used);
    show_difference(score, number_used);

    return 0;
}
```

```
void fill_array(int a[], int size, int& number_used)
{
    using namespace std;
    cout << "Enter up to " << size << " nonnegative whole numbers.\n"
         << "Mark the end of the list with a negative number.\n";
    int next, index = 0;
    cin >> next;
    while ((next >= 0) && (index < size))
    {
        a[index] = next;
        index++;
        cin >> next;
    }
    number_used = index;
}

double compute_average(const int a[], int number_used)
{
    double total = 0;
    for (int index = 0; index < number_used; index++)
        total = total + a[index];
    if (number_used > 0)
    {
        return (total/number_used);
    }
    else
    {
        using namespace std;
        cout << "ERROR: number of elements is 0 in compute_average.\n"
             << "compute_average returns 0.\n";
        return 0;
    }
}

void show_difference(const int a[], int number_used)
{
    using namespace std;
    double average = compute_average(a, number_used);
    cout << "Average of the " << number_used
         << " scores = " << average << endl
         << "The scores are:\n";
    for (int index = 0; index < number_used; index++)
    cout << a[index] << " differs from average by "
         << (a[index] - average) << endl;
}
```

```
This program reads golf scores and shows
how much each differs from the average.
Enter golf scores:
Enter up to 10 nonnegative whole numbers.
Mark the end of the list with a negative number.
69 74 68 -1
Average of the 3 scores = 70.3333
The scores are:
69 differs from average by -1.33333
74 differs from average by 3.66667
68 differs from average by -2.33333
```

Търсене в масив

- Търсене в масива на дадена стойност – последователно търсене

```
#include <iostream>
const int DECLARED_SIZE = 20;

void fill_array(int a[], int size, int& number_used);
int search(const int a[], int number_used, int target);
//Precondition: number_used is <= the declared size of a.
//Also, a[0] through a[number_used -1] have values.
//Returns the first index such that a[index] == target,
//provided there is such an index; otherwise, returns -1.

int main( )
{
    using namespace std;
    int arr[DECLARED_SIZE], list_size, target;

    fill_array(arr, DECLARED_SIZE, list_size);

    char ans;
    int result;
    do
    {
        cout << "Enter a number to search for: ";
        cin >> target;

        result = search(arr, list_size, target);
        if (result == -1)
            cout << target << " is not on the list.\n";
        else
            cout << target << " is stored in array position "
                 << result << endl
                 << "(Remember: The first position is 0.)\n";

        cout << "Search again?(y/n followed by Return): ";
        cin >> ans;
    }while ((ans != 'n') && (ans != 'N'));

    cout << "End of program.\n";
    return 0;
}
//Uses iostream:
void fill_array(int a[], int size, int& number_used)
```

```
int search(const int a[], int number_used, int target)
{
    int index = 0;
    bool found = false;
    while ((!found) && (index < number_used))
        if (target == a[index])
            found = true;
        else
            index++;

    if (found)
        return index;
    else
        return -1;
}
```

```
Enter up to 20 nonnegative whole numbers.
Mark the end of the list with a negative number.
10 20 30 40 50 60 70 80 -1
Enter a number to search for: 10
10 is stored in array position 0
(Remember: The first position is 0.)
Search again?(y/n followed by Return): y
Enter a number to search for: 40
40 is stored in array position 3
(Remember: The first position is 0.)
Search again?(y/n followed by Return): y
Enter a number to search for: 42
42 is not on the list.
Search again?(y/n followed by Return): n
End of program.
```

Сортиране на масиви

- Сортиране във възходящ и низходящ ред.
- Различни алгоритми за сортиране: метод на мехурчето, quick sort, сортиране чрез пряка селекция и др.
- Интересен сайт : http://www.sorting-algorithms.com/insertion-sort
- Пример с метод на мехурчето; използване на флаг
- Пример с намиране на най-голямата стойност

```
//Tests the procedure sort.
#include <iostream>

void fill_array(int a[], int size, int& number_used);
//Precondition: size is the declared size of the array a.
//Postcondition: number_used is the number of values stored in a.
//a[0] through a[number_used - 1] have been filled with
//nonnegative integers read from the keyboard.

void sort(int a[], int number_used);
//Precondition: number_used <= declared size of the array a.
//The array elements a[0] through a[number_used - 1] have values.
//Postcondition: The values of a[0] through a[number_used - 1] have
//been rearranged so that a[0] <= a[1] <= ... <= a[number_used - 1].

void swap_values(int& v1, int& v2);
//Interchanges the values of v1 and v2.

int index_of_smallest(const int a[], int start_index, int number_used);
//Precondition: 0 <= start_index < number_used. Referenced array elements have
//values.
//Returns the index i such that a[i] is the smallest of the values
//a[start_index], a[start_index + 1], ..., a[number_used - 1].

int main( )
{
    using namespace std;
    cout << "This program sorts numbers from lowest to highest.\n";

    int sample_array[10], number_used;
    fill_array(sample_array, 10, number_used);
    sort(sample_array, number_used);

    cout << "In sorted order the numbers are:\n";
    for (int index = 0; index < number_used; index++)
        cout << sample_array[index] << " ";
    cout << endl;

    return 0;
}

//Uses iostream:
void fill_array(int a[], int size, int& number_used)
```

<The rest of the definition of fill_array is given in Display 7.9.>

```
void sort(int a[], int number_used)
{
    int index_of_next_smallest;
    for (int index = 0; index < number_used - 1; index++)
    {//Place the correct value in a[index]:
        index_of_next_smallest =
                    index_of_smallest(a, index, number_used);
        swap_values(a[index], a[index_of_next_smallest]);
        //a[0] <= a[1] <=...<= a[index] are the smallest of the original arra
        //elements. The rest of the elements are in the remaining positions.
    }
}

void swap_values(int& v1, int& v2)
{
    int temp;
    temp = v1;
    v1 = v2;
    v2 = temp;
}

int index_of_smallest(const int a[], int start_index, int number_used)
{
    int min = a[start_index],
        index_of_min = start_index;
    for (int index = start_index + 1; index < number_used; index++)
        if (a[index] < min)
        {
            min = a[index];
            index_of_min = index;
            //min is the smallest of a[start_index] through a[index]
        }

    return index_of_min;
}
```

```
This program sorts numbers from lowest to highest.
Enter up to 10 nonnegative whole numbers.
Mark the end of the list with a negative number.
80 30 50 70 60 90 20 30 40 -1
In sorted order the numbers are:
20 30 30 40 50 60 70 80 90
```

Многомерни масиви

- С позволява декларирането на многомерни масиви

```
char page[30][100];
```

- Първата колона е от 0до29, втората от 0 до 99

```
page[0][0], page[0][1], ..., page[0][99]
page[1][0], page[1][1], ..., page[1][99]
page[2][0], page[2][1], ..., page[2][99]
          .
          .
          .
page[29][0], page[29][1], ..., page[29][99]
```

- В C++ многомерният масив може да се разглежда като масив от масиви

Предаване на многомерни масиви като параметри

```
void display_page(const char p[][100], int size_dimension_1)
{
    for (int index1 = 0; index1 < size_dimension_1; index1++)
    {//Printing one line:
        for (int index2 = 0; index2 < 100; index2++)
            cout << p[index1][index2];
        cout << endl;
    }
}
```

- Размерността на първия индекс на масива не се показва директно , а се предава като размерност - цяло число

Пример

```
eads quiz scores for each student into the two-dimensional array grade (but
he input code is not shown in this display). Computes the average score
or each student and the average score for each quiz. Displays the quiz scores
nd the averages.
clude <iostream>
clude <iomanip>
st int NUMBER_STUDENTS = 4, NUMBER_QUIZZES = 3;

d compute_st_ave(const int grade[][NUMBER_QUIZZES], double st_ave[]);
recondition: Global constants NUMBER_STUDENTS and NUMBER_QUIZZES
re the dimensions of the array grade. Each of the indexed variables
rade[st_num-1, quiz_num-1] contains the score for student st_num on quiz
uiz_num.
ostcondition: Each st_ave[st_num-1] contains the average for student
umber stu_num.

d compute_quiz_ave(const int grade[][NUMBER_QUIZZES], double quiz_ave[]);
recondition: Global constants NUMBER_STUDENTS and NUMBER_QUIZZES
re the dimensions of the array grade. Each of the indexed variables
rade[st_num-1, quiz_num-1] contains the score for student st_num on quiz
uiz_num.
ostcondition: Each quiz_ave[quiz_num-1] contains the average for quiz number
uiz_num.

d display(const int grade[][NUMBER_QUIZZES],
                        const double st_ave[], const double quiz_ave[]);
recondition: Global constants NUMBER_STUDENTS and NUMBER_QUIZZES are the
imensions of the array grade. Each of the indexed variables grade[st_num-1,
uiz_num-1] contains the score for student st_num on quiz quiz_num. Each
t_ave[st_num-1] contains the average for student stu_num. Each
uiz_ave[quiz_num-1] contains the average for quiz number quiz_num.
ostcondition: All the data in grade, st_ave, and quiz_ave has been output.

main( )

using namespace std;
int grade[NUMBER_STUDENTS][NUMBER_QUIZZES];
double st_ave[NUMBER_STUDENTS];
double quiz_ave[NUMBER_QUIZZES];

code for filling the array grade goes here, but is not shown.>
```

```
    compute_st_ave(grade, st_ave);
    compute_quiz_ave(grade, quiz_ave);
    display(grade, st_ave, quiz_ave);
    return 0;
}

void compute_st_ave(const int grade[][NUMBER_QUIZZES], double st_ave[])
{
    for (int st_num = 1; st_num <= NUMBER_STUDENTS; st_num++)
    {//Process one st_num:
        double sum = 0;
        for (int quiz_num = 1; quiz_num <= NUMBER_QUIZZES; quiz_num++)
            sum = sum + grade[st_num-1][quiz_num-1];
        //sum contains the sum of the quiz scores for student number st
        st_ave[st_num-1] = sum/NUMBER_QUIZZES;
        //Average for student st_num is the value of st_ave[st_num-1]
    }
}


void compute_quiz_ave(const int grade[][NUMBER_QUIZZES], double quiz_ave[
{
    for (int quiz_num = 1; quiz_num <= NUMBER_QUIZZES; quiz_num++)
    {//Process one quiz (for all students):
        double sum = 0;
        for (int st_num = 1; st_num <= NUMBER_STUDENTS; st_num++)
            sum = sum + grade[st_num-1][quiz_num-1];
        //sum contains the sum of all student scores on quiz number qui
        quiz_ave[quiz_num-1] = sum/NUMBER_STUDENTS;
        //Average for quiz quiz_num is the value of quiz_ave[quiz_num-1
    }
}


//Uses iostream and iomanip:
void display(const int grade[][NUMBER_QUIZZES],
                            const double st_ave[], const double quiz_ave[]
{
    using namespace std;
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(1);
    cout << setw(10) << "Student"
         << setw(5) << "Ave"
         << setw(15) << "Quizzes\n";
```

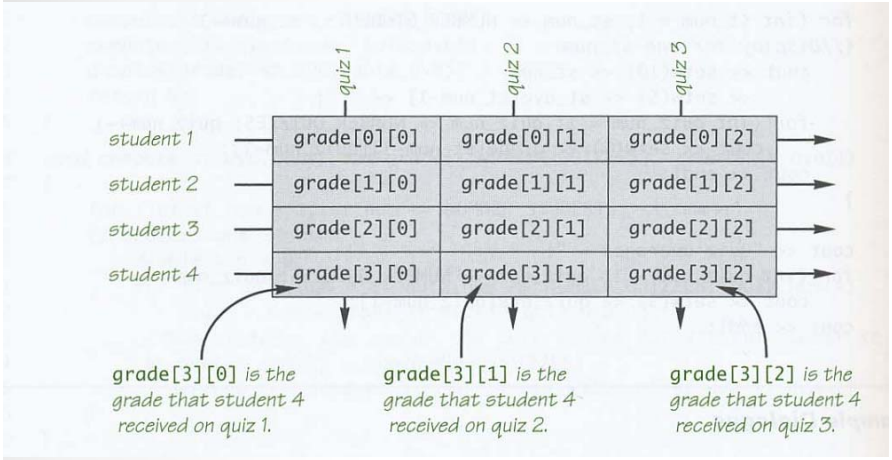
Пример

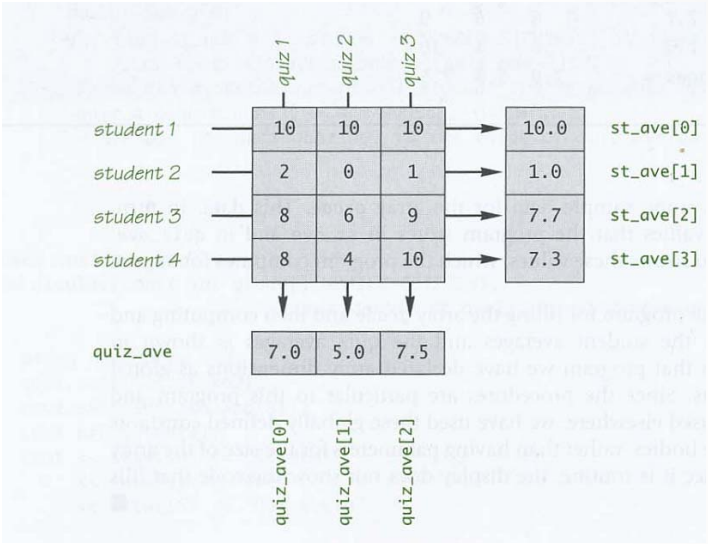
```
    for (int st_num = 1; st_num <= NUMBER_STUDENTS; st_num++)
    {//Display for one st_num:
        cout << setw(10) << st_num
             << setw(5) << st_ave[st_num-1] << " ";
        for (int quiz_num = 1; quiz_num <= NUMBER_QUIZZES; quiz_num++)
            cout << setw(5) << grade[st_num-1][quiz_num-1];
        cout << endl;
    }

    cout << "Quiz averages = ";
    for (int quiz_num = 1; quiz_num <= NUMBER_QUIZZES; quiz_num++)
        cout << setw(5) << quiz_ave[quiz_num-1];
    cout << endl;
}
```

```
Student     Ave          Quizzes
      1    10.0           10    10    10
      2     1.0            2     0     1
      3     7.7            8     6     9
      4     7.3            8     4    10
 Quiz averages =         7.0   5.0   7.5
```

Представяне





- Реално задачата се свежда до намиране на средно-аритметични стойности по редове и колони.