Ministry of Education and Science of Ukraine

Odessa National A.S. Popov Academy of Telecommunications

Sub-faculty of information technologies

ALGORITHMIZATION AND PROGRAMMING

Part 2
STRUCTURED DATA PROGRAMMING

Methodical instructions for laboratory training and exercises

Odessa 2020

УДК 004.43 План НМВ 2020 р.

Algorithmization and programming. Methodical instructions for laboratory training and exercises. Part 2. Structured data programming. / Y. V. Prokop, L. N. Bukata, O. G. Trofymenko - Odesa: A.S. Popov ONAT, 2020. - 57p.

Compilers: **Y. V. Prokop**, **L. N. Bukata, O. G. Trofymenko**

English text editor Bukata I.V.

These instructions for laboratory training and exercises contain theoretical information with examples of programs in C++ and variants of individual problems. Methodical instructions will be useful for students of the Academy of Telecommunications who are studying in English, fixing theoretical material, preparing to laboratory training and exercises in the discipline of "Algorithmization and programming".

It is intended for the acquisition of skills in operation on a personal computer and programming by students of the academy studying in English, with the purpose of further usage of these skills in daily professional work. Also, it will be useful for users of personal computers wishing to learn programming in Visual C++ environment.

APPROVED
by the sub-faculty IT meeting and
recommended for publication.

Minutes № 7 from 21.01.2020

APPROVED
by the Methodology Council
the Academy of Communications.

Minutes № 3 from 28.01.2020

Introduction

The Visual C++ language and development tools help you to develop native Universal Windows apps, native desktop and server applications, cross-platform libraries that run on Android and iOS as well as Windows, and managed apps that run on the .NET Framework.

Visual C++ supports two distinct, but closely related flavors of the C++ language, ISO/IEC standard C++ (native C++) and C++/CLI. We'll cover both of them in this course.

There are two principal types of applications, which we are going to create: console application (it executes in "black screen" in character mode) and application with graphical interface with Windows forms.

There's a lot of code even in a simple Windows program, and it's very important not to be distracted by the complexities of Windows while learning the ins and outs of C++.

These instructions present a brief theoretical information, examples of how to create software projects using Visual C++ for the calculation of linear, branched and cyclic structures, control questions and variants of individual problems to the nine laboratory exercises. Each of the proposed laboratory exercises contains problems of different complexity levels. The student himself or under the teacher's instructions selects problem of this or that complexity level according to a student's log number. Problems of basic level are obvious.

Before realization of the laboratory training the student should do the following:

- to select individual tasks with the teacher;
- to study relevant sections of the theoretical course in accordance with lecture notes and academic literature;
- to develop flowcharts for solving the individual problems;
- to write code in C++;
- to prepare a report of the laboratory exercise and submit it to the teacher for checking.

The students may perform Laboratory exercise only after they prepared a report.

The content of the report of the laboratory exercise:

- the title of the topic and the purpose of the laboratory exercise;
- answers to control questions;
- flowcharts for solving of the individual problems;
- programming code of his problems in C++;
- results of calculations on the computer.

The correctness of execution of the program and obtained results should be checked by the teacher.

Lab № 6

One-dimensional array. Operating with arrays in functions

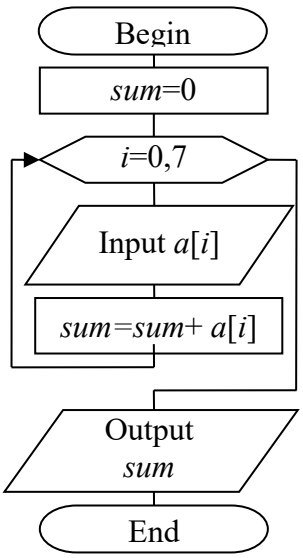
Goal: to get practical skills of operating with 1-dimensional arrays in Visual C++.

Examples of programs

Example 6.1. Enter array of 8 real numbers and calculate the sum of elements.

The program code:

```
#include <iostream>
using namespace std;
int main()
{ double a[8];              // Declaration of array
  double sum = 0;
  cout << "Enter 8 numbers" << endl;
  for (int i = 0; i < 8; i++)
  { cin >> a[i];            // Input element of array
    sum += a[i];            // Add element to the sum
  }
  cout << "Array sum = "<< sum << endl;
  system("pause");
  return 0;
}
```



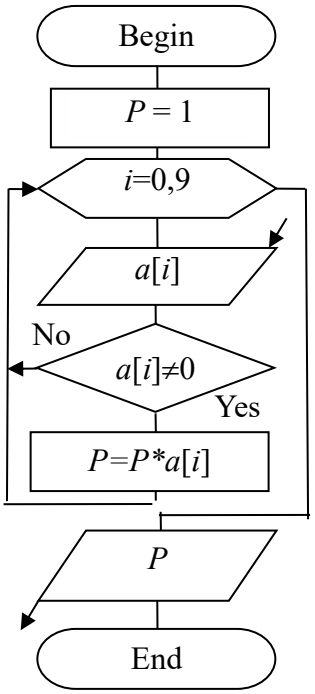
Results:

```
Enter 10 numbers
5  -2  8  1.7  -1  -4  -4.2  7
Array sum = 10.5
```

Example 6.2. Enter array of 10 integer numbers and calculate the product of non-zero elements.

The program code:

```
#include <iostream>
using namespace std;
int main()
{ const int n = 10;     // The number of elements
  int a[n], p = 1;
  cout << "Enter " << n << " integer numbers\n";
  for (int i = 0; i < n; i++)
  { cin >> a[i];        // Input element of array
    if (a[i] != 0)      // If element is non-zero,
       p *= a[i];       // to calculate the product
  }
  cout << "Product p = " << p << endl;
  system("pause");
  return 0;
}
```



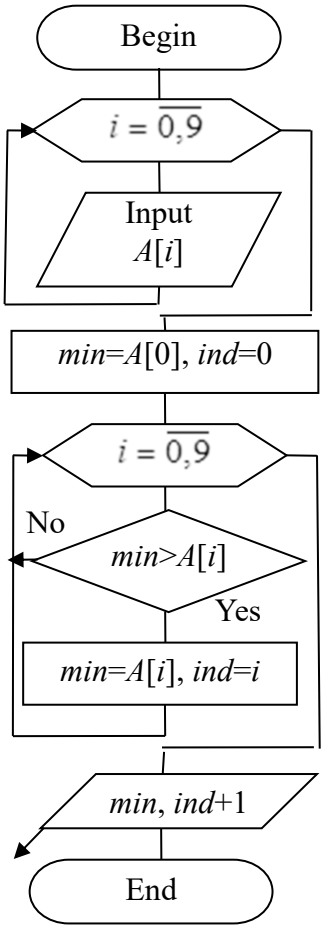
Results:

```
Enter 10 integer numbers
2  8  -4  0  1  0  3  -2  0  1
Product p = 384
```

Example 6.3. Enter array of 10 real numbers and calculate the minimal element and its index.

The program code:

```
#include <iostream>
using namespace std;
int main()
{
  const int n = 10;
  int i, ind;
  double a[n], min;
  cout << "Enter " << n << " real numbers:\n";
  for (i = 0; i < n; i++)
    cin >> a[i];
  min = a[0];
  ind = 0;
  for (i = 1; i < n; i++)
    if (a[i] < min)
    { min = a[i];
      ind = i;
    }
  cout << "min = " << min;
  cout << "  ind = " << ind << endl;
  system("pause");
  return 0;
}
```



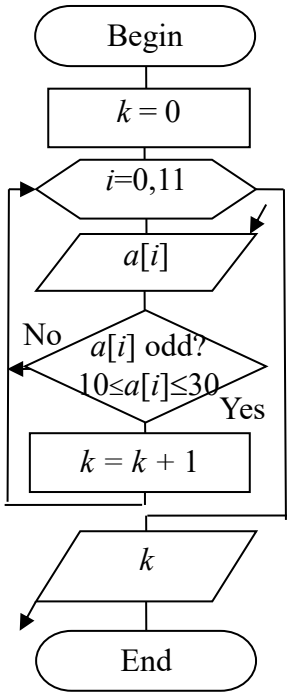
Results:

```
Enter 9 real numbers:
13  25.6  -8  3  1  -14  5  -1  -4
min = -14  ind = 5
```

Example 6.4. Enter array of 12 integer numbers and calculate the quantity of odd elements from the interval (–10, 30).

The program code:

```
#include <iostream>
using namespace std;
int main()
{
  int a[12], i, k = 0;
  cout << "Enter 12 integer numbers:\n";
  for (i = 0; i < 12; i++)
  {
    cin >> a[i];
    if (a[i] % 2 != 0 && a[i] > -10 && a[i] < 30)
      k++;
  }
  cout << "k = " << k << endl;
  system("pause");
  return 0;
}
```



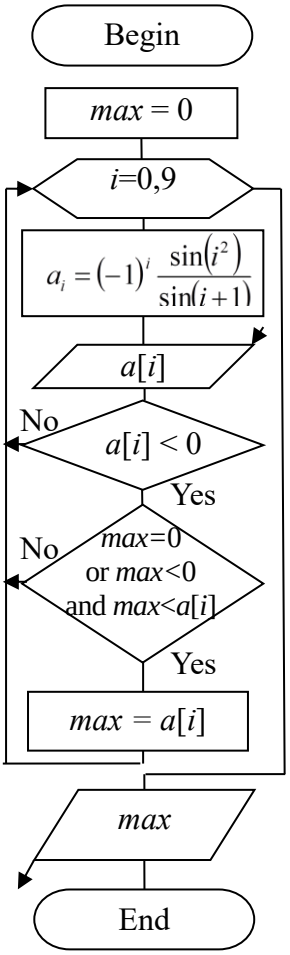
Results:

```
Enter 12 integer numbers:
29  -817  44  16  -75  1  230  -83  19  -34  8  5
k = 4
```

Example 6.5. Fill array of 10 elements under the formula:

$$a_i = (-1)^i \frac{\sin(i^2)}{\sin(i+1)}$$

and define the biggest negative element of array.



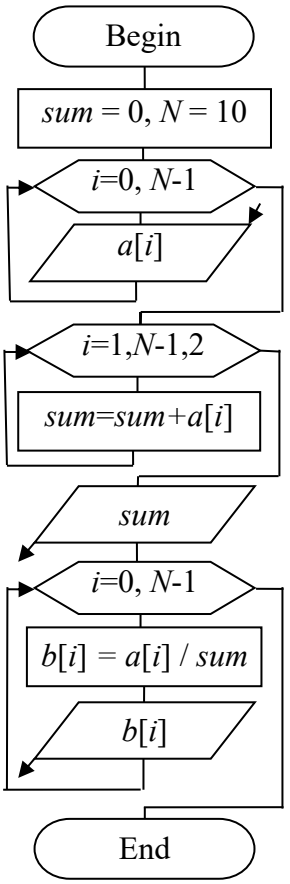
The program code:

```
#include <math.h>
int main()
{
  double a[10], max = 0;  int i;
  printf("Array: \n");
  for (i = 0; i < 10; i++)
  { a[i] = pow(-1,i) * sin(i*i) / sin(i+1);
    printf("%5.2f ", a[i]);
    if (a[i] < 0)
      if (max == 0 || max < 0 && max < a[i])
        max = a[i];
  }
  printf("\nmax = %5.2f\n", max);
  system("pause");
  return 0;
}
```

Results:

```
Array:
0.00  -0.93  -5.36   0.54  0.30  -0.47  -1.51  0.96  2.23  -1.16
max = -0.47
```

Example 6.6. Enter array of 10 integers. Create a new array with elements computed by dividing each element of the original array by the sum of its elements with odd indices.



The program code:

```
#include <iostream>
using namespace std;
int main()
{
  const int N = 10;
  int a[N], i, sum = 0;
  double b[N];
  cout << "Enter 10 integer numbers:\n";
  for (i = 0; i < N; i++)
    cin >> a[i];
  for (i = 1; i < N; i += 2)
    sum += a[i];
  cout << "Sum = " << sum << endl;
  cout << "New array: \n";
  for (i = 0; i < N; i++)
  { b[i] = 1.*a[i] / sum;
    cout << b[i] << " ";
  }
  cout << endl;
  system("pause");
  return 0;
}
```

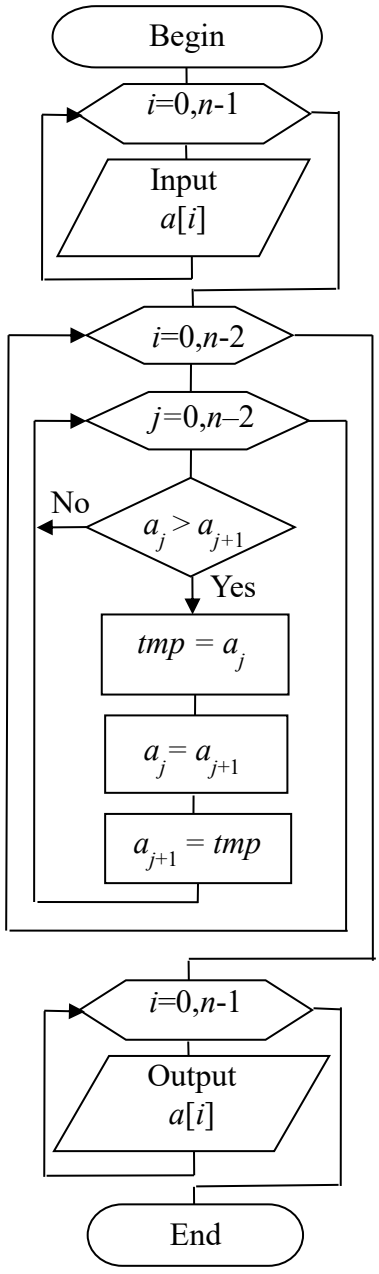
Results:

```
Enter 10 integer numbers:
-3  9  12  6  -5  -2  4  7  0  1
Sum = 21
New array:
-0.1428  0.4285  0.5714  0.2857  -0.2380  -0.09523  0.1904  0.3333  0  0.0476
```

Example 6.7. Enter array of 12 real numbers. Sort it on ascending order.

The program code:

```
#include <iostream>
using namespace std;
int main()
{
 const int n = 12;
 double a[n];  int i;
 cout << "Enter " << n << " real numbers:\n";
 for (i = 0; i < n; i++)
    cin >> a[i];
 for (int i = 0; i < n - 1; i++)
   for (int j = 0; j < n - 1; j++)
     if (a[j] > a[j + 1])
     {
       double temp = a[j];
       a[j] = a[j + 1];
       a[j + 1] = temp;
     }
 cout << "Sorted array:\n";
 for (i = 0; i < n; i++)
   cout << a[i] << " ";
 cout << endl;
 system("pause");
 return 0;
}
```



Results:

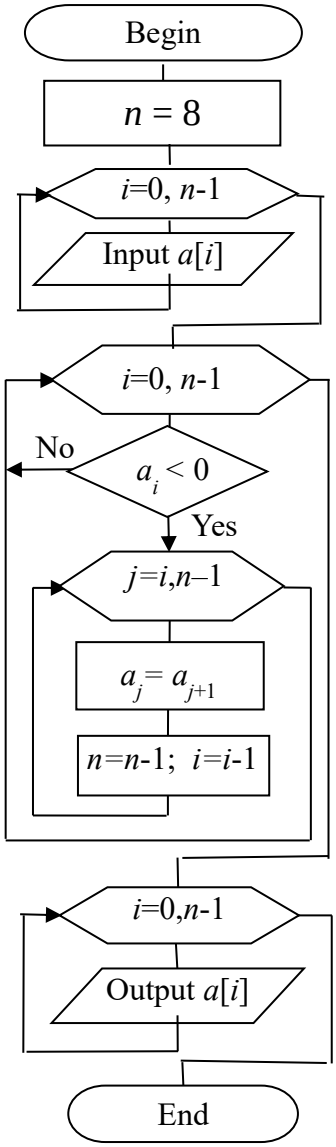
```
Enter 12 real numbers:
5  0  -2  -6  1  -3  8  9  23  5  -1  3
Sorted array:
-6  -3  -2  -1  0  1  3  5  5  8  9  23

Step-by-step array changing:
Inputted array:
5  0  -2  -6  1  -3  8  9  23  5  -1  3

Pass #: 0
0  -2  -6  1  -3  5  8  9  5  -1  3  23

Pass #:  1
-2  -6  0  -3  1  5  8  5  -1  3  9  23

Pass #:  2
-6  -2  -3  0  1  5  5  -1  3  8  9  23

Pass #:  3
-6  -3  -2  0  1  5  -1  3  5  8  9  23

Pass #:  4
-6  -3  -2  0  1  -1  3  5  5  8  9  23
```

```
Pass #:  5
-6  -3  -2  0  -1  1  3  5  5  8  9  23

Pass #:  6
-6  -3  -2  -1  0  1  3  5  5  8  9  23

Pass #:  7
-6  -3  -2  -1  0  1  3  5  5  8  9  23

Pass #:  8
-6  -3  -2  -1  0  1  3  5  5  8  9  23

Pass #:  9
-6  -3  -2  -1  0  1  3  5  5  8  9  23

Pass #:  10
-6  -3  -2  -1  0  1  3  5  5  8  9  23
```

Example 6.8. Enter array of 8 real numbers and delete all negative elements from it.

The program code:

```
#include <iostream>
using namespace std;
int main()
{
  double a[8];  int i, j, n = 8;
  cout << "Enter array of 8 real numbers: \n ";
  for (i = 0; i < n; i++)
    cin >> a[i];
  for (i = 0; i < n; i++)
    if (a[i] < 0)
    {
      for (j = i; j < n; j++)
        a[j] = a[j + 1];
      n--;    i--;
    }
  cout << "\nChanged array consists of " << n << "
   elements:\n ";
  for (i = 0; i < n; i++)
    cout << a[i] << "\t";
  cout << endl;
  system("pause");
  return 0;
}
```

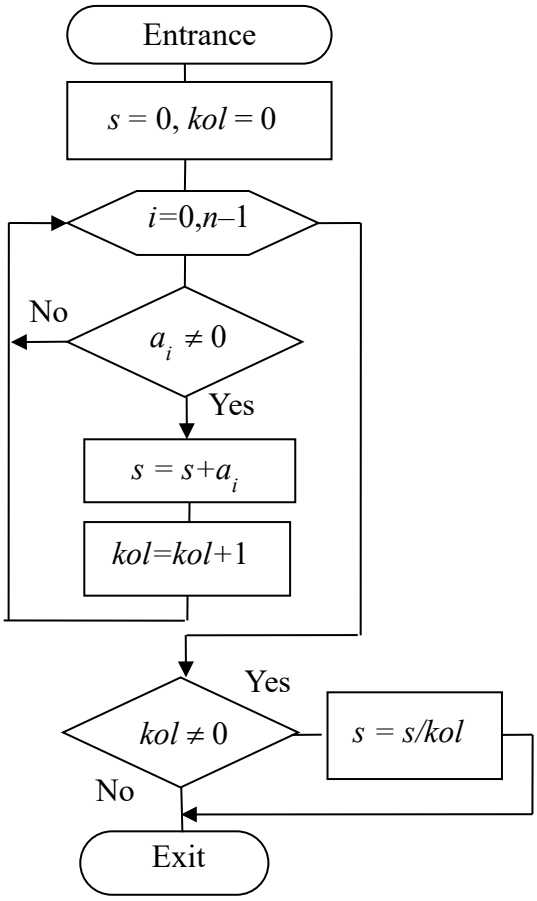


Results:

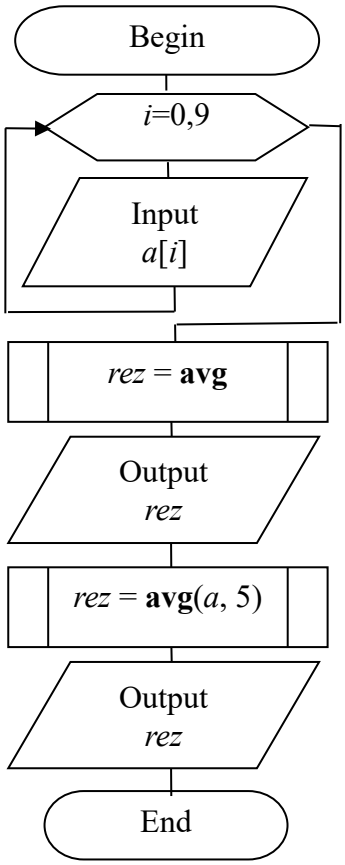
```
Enter array of 8 real numbers:
6  2.5  -3  7  1  0.1  -2  5

Changed array consists of 6 elements:
6       2.5       7       1       0.1       5
```

Example 6.9. Create a function for calculating an average of non-zero elements of an integer array. Use this function to find an average of non-zero elements for array of 10 integers: first for all elements, and then for this first half of the array.

The flowcharts of the avg function and of the main function:



Flowchart of function **avg()**



Flowchart of the main function

The program code:

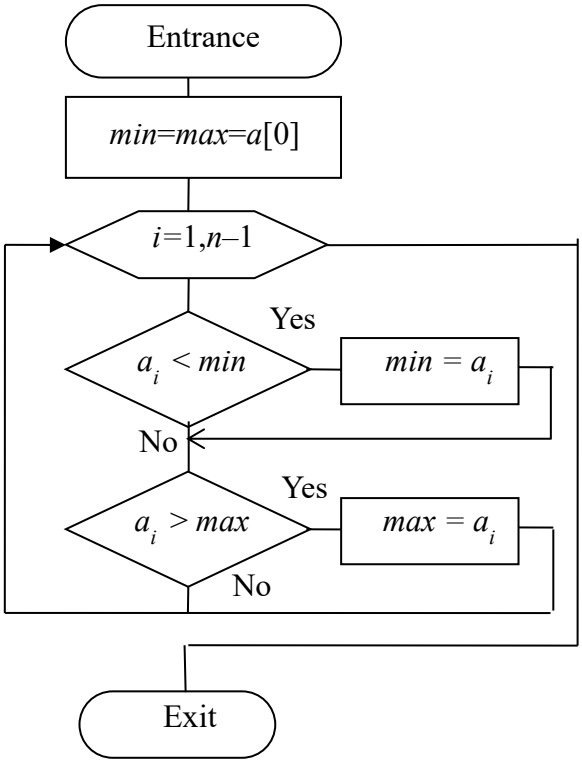
```
#include <iostream>
using namespace std;
double avg(int a[], int n)
{
  double s = 0; int i, kol = 0;
  for (i = 0; i < n; i++)
    if (a[i]!=0)
    {
      s += a[i];
      kol++;
    }
  if (kol) s /= kol;
  return s;
}

int main()
{
  int a[10], i;
  cout << "Enter 10 integer numbers:\n";
  for (i = 0; i < 10; i++)   cin >> a[i];
  cout << "\nAverage of 10 elements = " << avg(a, 10) << endl;
  cout << "\nAverage of the first 5 elements = " << avg(a, 5) << endl;
  system("pause");
  return 0;
}
```

Results:

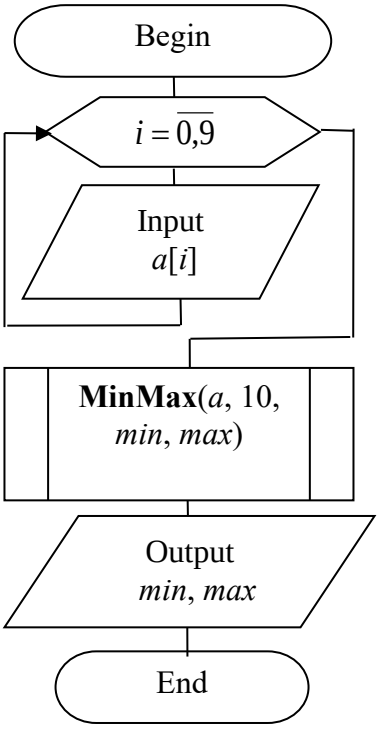
```
Enter 10 integer numbers:
-3  5  0  -8  0  2  3  -1  0  6
Average of 10 elements = 0.571429
Average of the first 5 elements = -2
```

Example 6.10. Create a function to define maximum and minimum elements of an array of 10 real numbers and call the function for the entered array.

The flowcharts of the MinMax function and of the main function:



Flowchart of function **MinMax()**



Flowchart of the main function

The program code:

```
#include <iostream>
using namespace std;

void MinMax(double a[], int n, double& min, double& max)
{
  min = max = a[0];
  for (int i = 1; i < n; i++)
  {
    if (a[i] < min) min = a[i];
    if (a[i] > max) max = a[i];
  }
}
int main()
{
  double a[10], min, max;
  cout << "Enter 10 numbers:\n";
  for (int i = 0; i < 10; i++)
    cin >> a[i];
  MinMax(a, 10, min, max);
  cout << "Minimum: " << min << endl;
```

```
    cout << "Maximum: " << max << endl;
    system("pause");
    return 0;
}
```

Results:

```
Enter 10 numbers:
5.3  2  -8  4  -3  7  0.5  -1.04  2.5  3
Minimum: -8
Maximum: 7
```

Control questions

1) What is array?
2) Write the declaration of array of 25 double numbers.
3) Assign zero to the first element of array A of 25 double numbers.
4) Which variable is the sum of array?

a)
```
s1=0;
for(i=0;i<10;i++)
  s1 += a[i];
```
b)
```
s2=1;
for(i=0;i<10;i++)
  s2 *= a[i];
```
c)
```
s3=0;
for(i=0;i<10;i++)
  if(a[i]>0) s3++;
```

5) Which variable is the number of positive elements of array?

a)
```
S1=0;
for(i=0;i<10;i++)
  S1 += a[i];
```
b)
```
S2=1;
for(i=0;i<10;i++)
  S2 *= a[i];
```
c)
```
S3=0;
for(i=0;i<10;i++)
  if(a[i]>0) S3++;
```

6) Which variable is the biggest element of array?

a)
```
S1=0;
for(i=0;i<10;i++)
  S1 += a[i];
```
b)
```
S2=a[0];
for(i=1;i<10;i++)
if(a[i]>S2)S2=a[i];
```
c)
```
S3=0;
for(i=0;i<10;i++)
if(a[i]>0)S3=S3+1;
```

7) How are arrays passed to functions?
8) What type has a function, which sorts array? Why?
9) What information do the following headers give about the functions:

```
int fun(double x[], int n);
void fun(double x[], int n);
double fun(double x[], int n);
double fun(double x[], int n, double& z);
```

Individual task

1. Create projects with array (Tables 6.1 and 6.2). Draw the flowcharts.

2. Create project with function (Table 6.3). Create function with one or two results and use it in the main function. Draw the flowcharts.

Table 6.1

№ var.	Size of array	Data type	Problems
1	15	int	Calculate the quantity and the sum of even elements of array
2	10	int	Calculate an average of positive elements
3	8	int	Calculate factorial of the last element of array
4	12	double	Calculate the product of elements which values are smaller than 6

№ var.	Size of array	Data type	Problems
5	14	int	Calculate an average of odd elements
6	18	int	Calculate the sum of elements which are multiple to 3
7	11	double	Calculate the sum of elements which absolute value is not bigger than 10
8	9	int	Calculate a quantity of elements from (0, 7]
9	10	double	Calculate an average of elements with absolute value bigger than 12
10	11	double	Calculate a product of elements from (-4, 5)
11	17	int	Calculate an average of the minimum and the maximum of array elements
12	9	double	Output the quantity of elements which values are greater than the value of the first element
13	15	int	Find the indices of the minimum and the maximum of array
14	10	double	Calculate the sum of array elements, which values belong to the interval [3, 6]
15	8	int	Calculate the product of odd array elements
16	12	double	Find minimum and maximum of array elements
17	9	int	Calculate the product of odd elements with even indices
18	10	double	Find the difference between the first element and the sum of array

Table 6.2

№ var	Size of array	Data type	Problems
1	15	double	Swap the smallest element and the next to the last element
2	10	int	Calculate the sum of positive odd elements and change even elements with this sum
3	12	int	Define, whether array is sorted in ascending order
4	8	double	Calculate factorial of the first element of array with the value smaller than 8
5	14	int	Swap the biggest element and the first element
6	18	double	Calculate the new array as a difference between elements of the inputted array and their average
7	11	int	Change all zero elements with the last element
8	12	double	Calculate the new array as a sum of elements of the inputted array and its minimum
9	9	int	Change all elements with odd indexes to array average
10	11	int	Define, whether array is sorted in descending order
11	13	double	Calculate the new array as a sum between elements of the inputted array and its maximum
12	7	int	Swap the maximal element and the minimal element
13	8	int	Calculate the factorial of the absolute value of maximal element
14	15	double	Swap the first half of the array with the second half
15	13	double	Change the minimal and maximal elements to the average of all

№ var	Size of array	Data type	Problems
			elements
16	11	double	Swap the pairs of elements: 0^{th} and 1^{st}, 2^{nd} and 3^{rd} etc.
17	8	int	Increase all even elements by 3, change all zero elements to array maximum
18	7	int	Swap the first and the last positive elements

Table 6.3

Problems with functions with one or two results

№ var.	Size of array	Data type	Individual problem
1	15	int	Calculate the factorial of the first element of array which is smaller than 10
2	10	int	Calculate the number of array elements which are placed after the first zero element
3	8	int	Calculate the factorial of the first positive element of array
4	12	double	Calculate the smallest negative element of array
5	14	int	Calculate the number and the sum of even elements of array
6	18	double	Calculate the sum of array elements which absolute value is not bigger than 10
7	11	int	Calculate the sum and the number of array elements which are multiple to 3
8	13	double	Calculate the biggest positive element of array
9	15	int	Calculate the factorial of the last even element of array
10	10	double	Calculate the quantity of elements which are bigger than average
11	9	double	Find the indices of the minimal and the maximal elements of the array
12	13	int	Calculate the absolute value of the sum of all negative elements, and the sum of all positive
13	12	double	Calculate the sum of negative elements placed after the maximal element of the array
14	9	int	Calculate the difference between the sum of all positive elements and the sum of absolute values of all negative elements
15	10	double	Calculate the product of single-digit elements of an array
16	8	int	Define, whether array is sorted in ascending order
17	7	double	Define, whether array contains more positive elements than negative ones
18	9	int	Calculate the sum of even elements in the second half of array

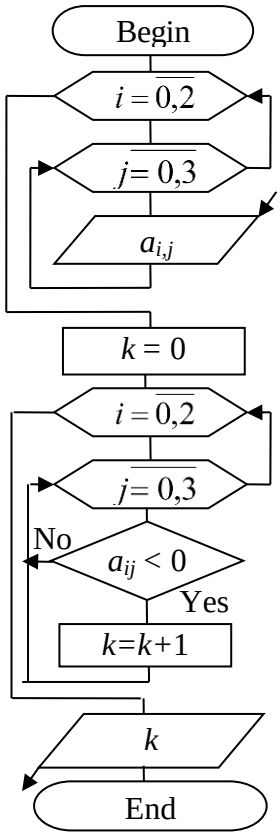
Lab № 7
Two-dimensional arrays

Goal: to get practical skills of operating with 2-dimensional arrays in Visual C++.

Examples of programs

Example 7.1. Enter a matrix of 3×4 integers and compute the quantity of negative elements.

```
#include <iostream>
using namespace std;
int main()
{
  int a[3][4], i, j; int k = 0;
  cout << “Input matrix 3x4: "<<endl;
  for (i = 0; i < 3; i++)
    for (j = 0; j < 4; j++)
      cin >> a[i][j];  // Input element of matrix
  for (i = 0; i < 3; i++)
    for (j = 0; j < 4; j++)
      if (a[i][j] < 0) k++;
     cout<<"\nQuantity of negative elements - "<<k<<endl;
  system("pause");
  return 0;
}
```



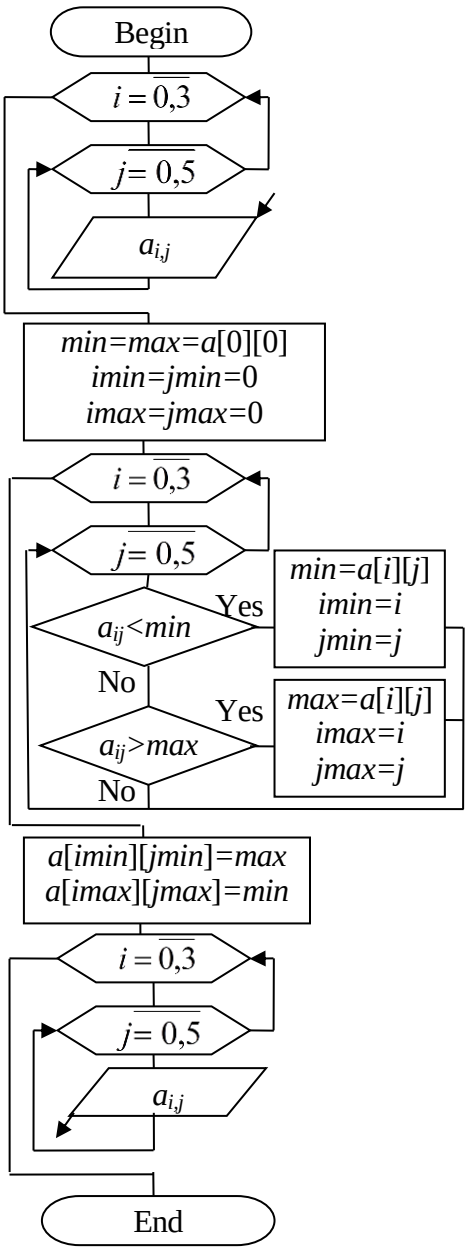
Results:

```
Input matrix 3x4:
1 8 0 -1
3 4 -5 2
0 -1 -7 5

Quantity of negative elements – 4
```

Example 2. Enter the matrix 4×6 of real numbers and swap the maximal and minimal elements.

```
#include <iostream>
using namespace std;
int main()
{
  double a[4][6], min, max;
  int i, j, imin, jmin, imax, jmax;
  cout << "Input matrix 4x6:"<<endl;
  for (i = 0; i < 4; i++)
    for (j = 0; j < 6; j++)
      cin >> a[i][j];
  min = max = a[0][0];
  imin = jmin = imax = jmax = 0;
```

```
  for (i = 0; i < 4; i++)
    for (j = 0; j < 6; j++)
    {
      if (a[i][j] < min)
      {
        min = a[i][j];
        imin = i;
        jmin = j;
      }
      if (a[i][j] > max)
      {
        max = a[i][j];
        imax = i;
        jmax = j;
      }
    }
 a[imin][jmin] = max;
 a[imax][jmax] = min;
 cout<<"\nMatrix with min and max swapped:"<<endl;
 for (i = 0; i < 4; i++)
  {
    for (j = 0; j < 6; j++)
      cout << a[i][j] << "\t";
    cout << endl;
  }
 system("pause");
 return 0;
}
```



Results:

```
Input matrix 4x6:
6 9 -3 0 0 1
1 1 2 -3 0 -1
8 -3 -4 0 0 1
5 -2 -7 0 1 0

Matrix with min and max swapped:
6       -7      -3      0       0       1
1       1       2       -3      0       -1
8       -3      -4      0       0       1
5       -2      9       0       1       0
```

Example 7.3. Enter a matrix 5×7 of integers and create a vector of sums of the negative elements of matrix columns.

```
#include <iostream>
using namespace std;
int main()
{
  int a[5][7], x[7], i, j, s;
  cout << "Input matrix 5x7:"<<endl;
  for (i = 0; i < 5; i++)
    for (j = 0; j < 7; j++)
       cin >> a[i][j];
  cout << "Vector:\n";
```

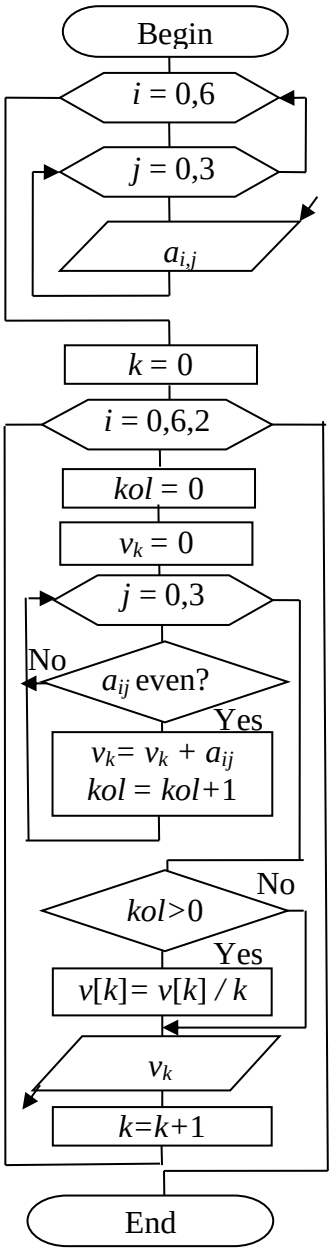
```
  for (j = 0; j < 7; j++)
  {
    s = 0;
    for (i = 0; i < 5; i++)
      if (a[i][j] < 0)
        s += a[i][j];
    x[j] = s;
  }
  for (i = 0; i < 7; i++)
    cout << x[i] << "\t";
  cout << endl;
  system("pause");
  return 0;

}
```

Results:

```
Input matrix 5x7:
 7   1   0   0   2  -4   1
 4   2  -2   0   0   0   0
-1   2   3   4   5  -1  -1
-2  -4  -5   0  -1  -1   6
 0   0   1   1   2   2   3
Vector:
-3  -4  -7   0  -1  -6  -1
```

Example 7.4. Input matrix 7×5 of integers. Form a vector of averages of even elements of odd (1, 3, and 5) matrix rows.

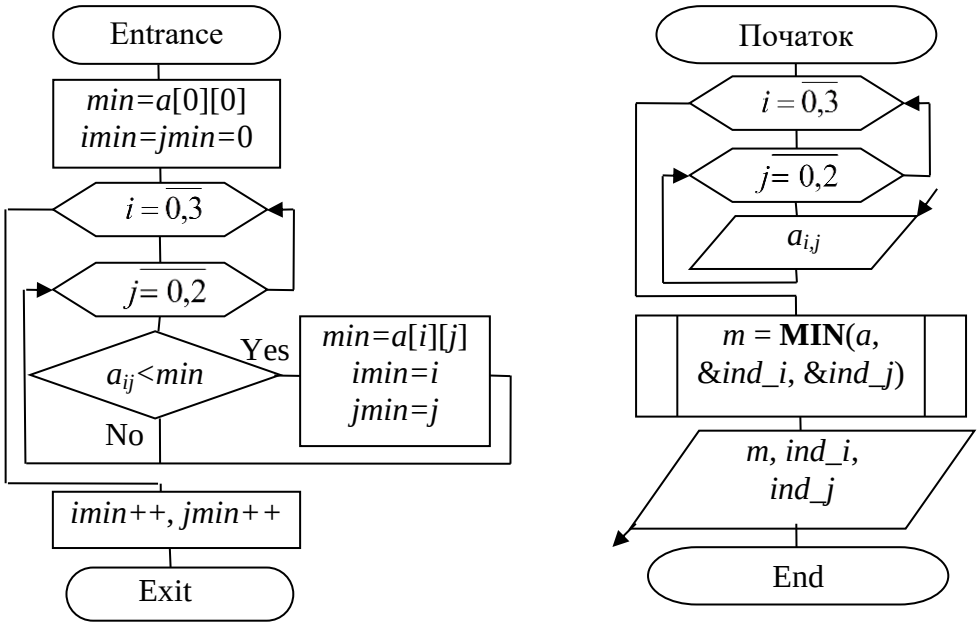
```
#include <iostream>
using namespace std;
int main()
{
  int i, j, a[7][5]; double v[3];
  cout << "Input matrix 7x5:"<<endl;
  for (i = 0; i < 7; i++)
  for (j = 0; j < 5; j++) cin >> a[i][j];
  cout << "\nVector" << endl;
  int kol, k = 0;
  for (i = 1; i < 7; i += 2)
  {
    kol = 0; v[k] = 0;
    for (j = 0; j < 5; j++)
      if (a[i][j] % 2 == 0 && a[i][j])
      { v[k] += a[i][j];
        kol++;
      }
    if (kol>0) v[k] /= kol;
    cout << v[k] << "\t\t";
    k++;
  }
  system("pause");
  return 0;
}
```



Results:

```
Input matrix 7x5:
5   0   0  -2  3
8   7   6  -2  0
1   2   0   0  9
1   0   2   0  3
7  -3  -4  -5  0
1  -4  -8   0  2
5   4   3   1  2
Vector
4     2     -3.33333
```

Example 7.5. Input matrix 4×3 of real numbers and calculate with the function the minimum element and its indexes.



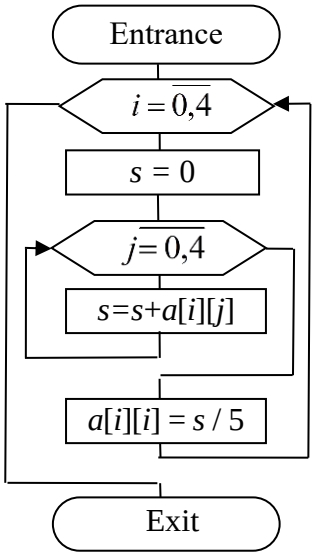

```
#include <iostream>
using namespace std;
double MIN(double a[4][3], int& imin, int& jmin)
{
  double min = a[0][0];  imin = jmin = 0;
  for (int i = 0; i < 4; i++)
    for (int j = 0; j < 3; j++)
      if (a[i][j] < min)
      {
        min = a[i][j]; imin = i;  jmin = j;
      }
  return min;
}

int main()
{
  double a[4][3], m;
  int i, j, ind_i, ind_j;
  cout << " Input matrix 4x3:" << endl;
  for (i = 0; i < 4; i++)
  for (j = 0; j < 3; j++) cin >> a[i][j];
  m = MIN(a, ind_i, ind_j);
```

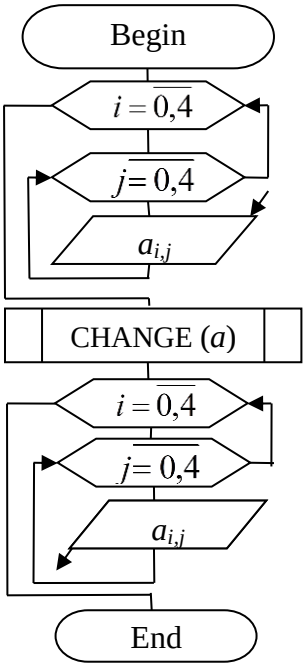
```
    cout << "\n Min = " << m << "\n in " << ind_i+1 << "-th row and " <<
         ind_j+1 << "-th column" << endl;
    system("pause");
    return 0;
}
```

Results:

```
Input matrix 4x3:
 5 -8  0
-1  2  2
-5  9  2
 1  2  3
 Min = -8
 in 1-th row and 2-th column
```

Example 7.6. Input 5×5 matrix of real numbers. Use the function to change the elements of the main diagonal with the arithmetic mean of the corresponding row.






```
#include <iostream>
using namespace std;
void CHANGE(double a[5][5])
{
  for (int i = 0; i < 5; i++)
  {
    double s = 0;
    for (int j = 0; j < 5; j++)
      s += a[i][j];
    a[i][i] = s / 5;
  }
}

int main()
{
  double a[5][5];
  int i, j;
```

```
  cout << " Input matrix 5x5:" << endl;
  for (i = 0; i < 5; i++)
    for (j = 0; j < 5; j++)
      cin >> a[i][j];
  CHANGE(a);
  cout << "\n Matrix with changes" << endl;
  for (i = 0; i < 5; i++)
  {
    for (j = 0; j < 5; j++)
      cout << a[i][j] << "\t";
    cout << endl;
  }
  system("pause");
  return 0;
}
```

Results:

```
Input matrix 5x5:
1   2   3   4   5
1   0   2   0   3
1   1   2   2   3
0   0   1   2   3
1   1   1   1   4

 Matrix with changes
3    2     3    4    5
1    1.2   2    0    3
1    1     1.8  2    3
0    0     1    1.2  3
1    1     1    1    1.6
```

Example 7.7. Fill matrix of 5×10 with random integers and create vector of maximal elements of the columns.

```
#include <iostream>
using namespace std;
void vec_f(int a[5][10], int x[10])
{
  int max;
  for (int j = 0; j < 10; j++)
  {
    max = a[0][j];
    for (int i = 0; i < 5; i++)
      if (max < a[i][j])  max = a[i][j];
    x[j] = max;
  }
}
int main()
{
  int a[5][10], x[10];
  for (int i = 0; i < 5; i++)
    for (int j = 0; j < 10; j++)
      a[i][j] = rand() % 100 - 50;
  cout << "Matrix: " << endl;
  for (int i = 0; i < 5; i++)
  { for (int j = 0; j < 10; j++)  cout << a[i][j] << "\t";
    cout << endl;
  }
```

```
        vec_f(a, x);
        cout << "Vector: " << endl;
        for (int j = 0; j < 10; j++)
          cout << x[j] << "   ";
        system("pause");
        return 0;
      }
```

Results:

```
Matrix:
-9     17    -16    -50    19    -26     28    8     12    14
-45    -5     31    -23    11     41     45   -8    -23   -14
 41    -46   -48      3    42     32    -29   -34   -32    45
-3     -24    21    -12    19    -38     17    49   -15    44
-47    -39   -28    -17    23     14    -9    -39    3     18
Vector:
 41     17     31      3    42     41     45    49    12    45
```

Control questions

1) Which of the following declarations of two-dimensional arrays are incorrect? Why?
   а) `int C[1..5, 1..5];`
   б) `float C[5][5];`
   в) `float C[1..5][1..5];`
   г) `int C: [5][5];`
2) How are elements of two-dimensional array allocated in memory?
3) Declare matrix S of integer numbers with size 7×3.
4) Which of the following fragments calculates the sum of elements of the main diagonal of integer matrix 5×5?
   а) `for(i=0, s=0; i<5; i++) s++;`
   б) `for(i=0, s=0; i<5; i++) s+=A[i][i];`
   в) `for(i=0, s=0; i<5; i++) for(j=0; j<5; j++) s+=A[i][j];`
   г) `for(i=0, s=0; i<5; i++) A[i][i]=0;`
5) How matrices are passed to functions as parameters?
6) What type has a function which for matrix of integer numbers:
   а) finds the biggest element?
   б) calculates the average?
   в) creates vector?
   г) replaces some elements in this matrix?
7) Which declarations are incorrect and why?

```
void fun(int a[][4], int m);
void fun(int a[4][], int m);
void fun(int a[3][4]);
void fun(int a[][], int m, int n);
```

Individual task

Create projects and flowcharts for problems in Tables 7.1-7.5. Process matrices in functions (Tables 7.3-7.5).

Table 7.1

№ var	Individual problem
1	Input real matrix with 5 rows and 4 columns. Calculate the number of positive, negative and zero elements
2	Input integer matrix with 4 rows and 5 columns. Find the smallest element and its indexes
3	Input real matrix with 5 rows and 5 columns. Find the smallest element of the main diagonal and the index of its row
4	Input real matrix with 4 rows and 4 columns. Swap elements of the first row and elements of the secondary diagonal
5	Input integer matrix with 3 rows and 5 columns. Change all negative elements to zero
6	Input real matrix with 6 rows and 6 columns. Find the smallest and the biggest elements
7	Input real matrix with 5 rows and 5 columns. Calculate the sum of elements of the secondary diagonal
8	Input integer matrix with 6 rows and 3 columns. Output indices of positive elements
9	Input double matrix with 3 rows and 6 columns. Find the biggest element of the second row
10	Input real matrix with 4 rows and 4 columns. Find the biggest element of the main diagonal and the index of its column
11	Input integer matrix with 5 rows and 5 columns. Swap elements of the first column and elements of the secondary diagonal
12	Input integer matrix with 3 rows and 4 columns. Change all positive elements to the average of all elements
13	Input real matrix with 6 rows and 6 columns. Find indexes of the smallest and the biggest elements in the last column
14	Change the zero elements of integer matrix 5×5 to its maximum element
15	Swap the first and the last negative elements in the real matrix 7×3
16	Input integer matrix 6×5. Swap the first and the last even elements
17	Input double matrix 5×4. Define the index of the row with minimal element
18	Input double matrix 5×4. Define the index of the column with maximal element

Table 7.2

№ var	Individual problem
1	Input integer matrix with 5 rows and 6 columns. Calculate vector's element as the sums of odd columns
2	Input real matrix with 4 rows and 4 columns. Calculate vector's element as the scalar products of matrix rows and the last column
3	Input integer matrix with 6 rows and 4 columns. Calculate vector's element as the products of odd elements of even rows
4	Input integer matrix with 4 rows and 4 columns. Calculate vector's element as the scalar products of elements of the first row and columns of the matrix
5	Input real matrix with 6 rows and 4 columns. Calculate vector's element as the matrix column with the smallest sum of elements
6	Input integer matrix with 4 rows and 5 columns. Calculate vector's element as the products of even elements of odd columns
7	Input integer matrix with 4 rows and 5 columns. Calculate vector's element as the matrix row with the biggest sum of elements
8	Input integer matrix with 5 rows and 4 columns. Calculate vector's element as the averages of even columns
9	Input real matrix with 4 rows and 4 columns. Calculate vector's element as the scalar products of matrix rows and the last row
10	Input integer matrix with 5 rows and 3 columns. Calculate vector's element as the products of negative elements of even rows
11	Input integer matrix with 4 rows and 4 columns. Calculate vector's element as the scalar products of elements of the first column and rows of the matrix
12	Input real matrix with 6 rows and 5 columns. Calculate vector's element as the matrix column with the biggest sum of elements
13	Input integer matrix with 4 rows and 5 columns. Calculate vector's element as the averages of even elements of odd columns
14	Input integer matrix 3×5 with elements from 0 to 9. Calculate vector's element as the percentage of each of these numbers in the matrix
15	Input double matrix 6×7. Define the row with the minimum sum and swap this row with the first row
16	Input two matrices of real numbers 4×5. Swap the rows of the matrices containing maximum elements
17	Input double matrix 4×5. Calculate vector's element as the average of the columns containing the maximum and minimum elements.
18	Input two matrices of real numbers 4×5. Swap the columns of the matrices containing maximum elements

Table 7.3

Function with one or two results

№ var.	Size of array	Data type	Individual problem
1	5×5	Integer	Calculate the quantity of negative matrix elements
2	4×4	Double	Calculate the sum of elements on the matrix main diagonal
3	6×4	Integer	Find the smallest matrix element
4	3×3	Double	Calculate the product of non-zero matrix elements
5	4×5	Integer	Calculate the average of the smallest and the biggest matrix elements
6	3×5	Double	Calculate the quantity of elements, which are bigger than the first element of the matrix
7	5×3	Integer	Calculate the average of matrix elements
8	5×5	Integer	Calculate the quantity of positive elements of matrix
9	4×4	Double	Calculate the product of elements of the matrix main diagonal
10	6×4	Integer	Find the matrix maximum
11	5×5	Integer	Find the smallest matrix element
12	3×5	Integer	Define the row indices of the minimum and maximum matrix elements
13	4×3	Double	Calculate the average of negative elements
14	6×4	Integer	Calculate the quantity of nonzero matrix elements
15	5×5	Double	Find the matrix elements with maximal and minimal absolute values
16	4×5	Double	Calculate the average of the matrix elements with values from the interval [10, 20]
17	3×5	Integer	Find the smallest element of the matrix and its column index
18	4×6	Double	Calculate the product of maximums in the primary and secondary diagonals

Table 7.4

Functions for changing of matrix elements

№ var.	Size of array	Data type	Individual problem
1	4×3	Integer	Change even elements with 0
2	6×4	Double	Swap the smallest and the biggest matrix elements
3	4×4	Integer	Swap elements of the main and secondary diagonals
4	4×5	Double	Change all negative elements to the value of the smallest element
5	3×5	Integer	Calculate the sum of positive odd elements and change the elements in the corners of matrix to this sum
6	5×3	Integer	Change all non-zero elements to the value of the smallest element
7	5×3	Integer	Transpose matrix
8	4×4	Integer	Change odd elements to the sum of elements above the main diagonal
9	6×4	Double	Swap the first and the biggest matrix elements
10	4×4	Integer	Swap elements of the main diagonal and the last column
11	5×3	Integer	Change all odd elements to the value of the smallest element
12	5×5	Double	Arrange the elements of the main diagonal in reverse order
13	3×4	Integer	Change elements that are multiples of 5 to the maximal matrix element
14	3×6	Double	Swap elements of the first and last columns
15	5×5	Integer	Change all zero matrix elements to the maximum
16	4×6	Integer	Change all even elements of the matrix to the value of the last element
17	5×4	Double	Swap elements of the first and last rows
18	5×5	Integer	Change all negative elements under the main diagonal to 0

Table 7.5

Function for creation of vector

№ var.	Size of array	Data type	Individual problem
1	3×5	Double	Calculate vector's element as the squares of the smallest elements in columns
2	5×3	Integer	Calculate vector's element as the averages of elements in matrix rows
3	3×4	Double	Calculate vector's element as the sums of elements with values not bigger than 10, in columns
4	6×4	Integer	Calculate vector's element as the averages of positive elements in the matrix rows
5	4×3	Double	Calculate vector's element as the sums of positive elements in the matrix rows
6	5×4	Integer	Calculate vector's element as the averages of two-digit numbers in matrix columns
7	4×6	Double	Calculate vector's element as the sums of elements in the odd matrix columns
8	3×5	Double	Calculate vector's element as the maximums of matrix columns
9	5×3	Integer	Calculate vector's element as the sums of positive elements un the matrix rows
10	3×4	Double	Calculate vector's element as the products of columns elements with values from (-3, 4)
11	4×4	Double	Calculate vector's element as the squares of the main diagonal elements
12	5×5	Double	Calculate vector's element as the elements of the secondary diagonal
13	3×6	Integer	Calculate vector's element as the products of nonzero elements in matrix columns
14	5×5	Double	Calculate vector's element as the sums of the absolute values of the negative elements in the matrix rows
15	4×6	Integer	Calculate vector's element as the average of the first and the last elements in the matrix rows
16	3×4	Double	Calculate vector's element as the smallest elements in the matrix columns
17	5×5	Integer	Calculate vector's element as the sum of elements of the main and the secondary diagonal
18	5×4	Integer	Calculate vector's element as the average of the even elements in the matrix rows

Lab № 8
Pointers and dynamic memory

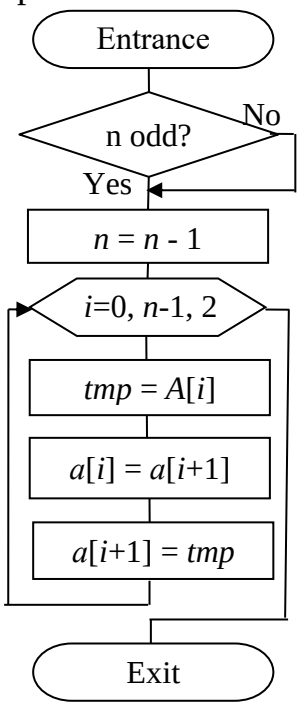
Goal: to get practical skills of programming use of pointers and dynamic memory with one-dimensional arrays.

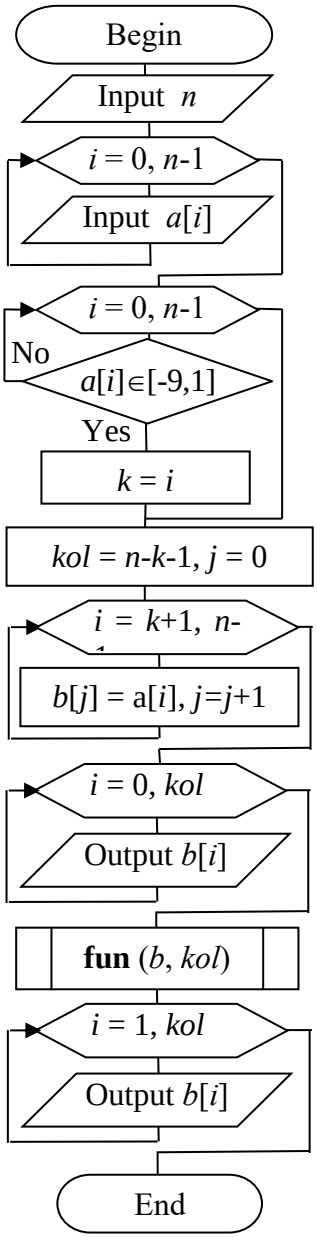
Examples of programs

Example 8.1. Enter a sequence of integers and create a dynamic array of numbers arranged after the first one-digit negative number (if there are no such number, select all). Use the function to swap elements: 1 and 2, 3 and 4, and so on.

The program code:

```
#include <iostream>
using namespace std;
void fun(int a[], int n)
{
  if (n % 2) n--;
  for (int i = 0; i < n; i += 2)
  {
    int tmp = a[i];
    a[i] = a[i + 1];
    a[i + 1] = tmp;
  }
}
int main()
{
  int n = 0, i, j, kol, k = -1;
  cout << "Input n="; cin >> n;
  int * a = new int[n];
  cout << "Input " << n
       << " integers\n";
  for (i = 0; i < n; i++) cin >> a[i];
  for (i = 0; i < n; i++)
    if (a[i]>-10 && a[i]<0)
    {
      k = i; break;
    }
  kol = n - k - 1;
  int * b = new int[kol];
  for (j = 0, i = k + 1; i < n; i++, j++)
    b[j] = a[i];
  cout << "Dynamic array: \n";
  for (i = 0; i < kol; i++) cout << b[i] << " ";
  fun(b, kol);
  cout<< "\n Dynamic array with swapped elements: \n";
  for (i = 0; i < kol; i++) cout << b[i] << " ";
  delete[]a;
  delete[]b;
  system("pause");
  return 0;
}
```





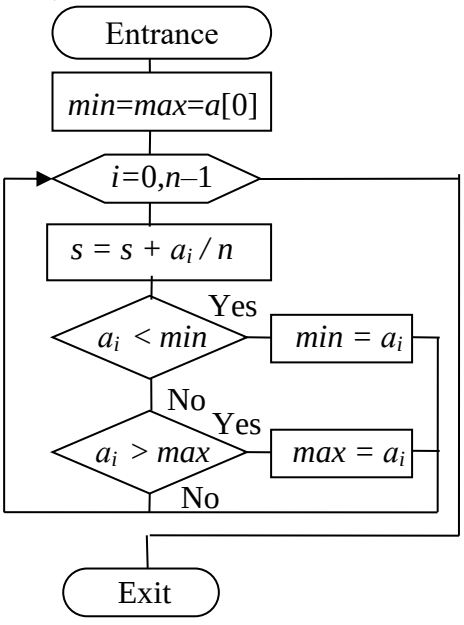
Results:

```
Input n=10
Input 10 integers
5
-13
7
3
2
10
-9
6
-10
8
Dynamic array:
6 -10 8
Dynamic array with swapped elements:
-10 6 8
```

Example 8.2. Enter some sequence of real numbers and create a dynamic array only from the numbers with values from the interval [60, 100]. Use the function to define the minimum and maximum elements and calculate the arithmetic mean of all elements.

The program code:

```
#include <iostream>
using namespace std;

double fun(double a[], int n, double &min, double &max)
{
  double s = 0;
  min = max = a[0];
  for (int i = 0; i < n; i++)
  {
    s += a[i] / n;
    if (min > a[i]) min = a[i];
    if (max < a[i]) max = a[i];
  }
  return s;
}

int main()
{
  int n = 0, i, j, kol = 0;
  double min, max, avg;
  cout << "Input n=";
  cin >> n;
  double * a = new double[n];
  cout << "Input " << n << " real numbers\n";
  for (i = 0; i < n; i++)
    cin >> a[i];
  for (i = 0; i < n; i++)
    if (a[i] >= 60 && a[i] <= 100)
      kol++;
  double * b = new double[kol];
```

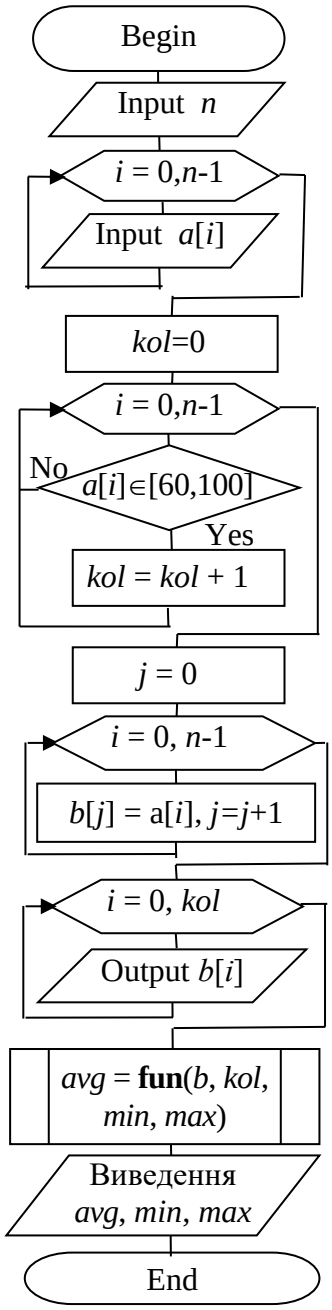


Flowchart of function **fun()**

```
  for (j = 0, i = 0; i < n; i++)
    if (a[i] >= 60 && a[i] <= 100)
    {
      b[j] = a[i];
      j++;
    }
  cout << "Dynamic array: \n";
  for (i = 0; i < kol; i++)
    cout << b[i] << " ";
  cout << endl;
  avg = fun(b, kol, min, max);
  cout << "Average=" << avg << ", min=" << min
       << ", max=" << max << endl;
  delete[]a;
  delete[]b;
  system("pause");
  return 0;
}
```

Results:

```
Input n=9
Input 9 real numbers
90
-123
0
3
216
88
4
56
99
Dynamic array:
90  88  99
Average=92.3333, min=88, max=99
```



Flowchart of the main function

Control questions

1) What is pointer? What is the syntax of its declaration?
2) What does `NULL` pointer mean?
3) How can we get an address of variable? What is `&` operand?
4) Explain the difference between variables `a` and `b`:
   a) `int a; double b;` b) `int *a; double *b;`
5) Explain the difference between usual and dynamic arrays.
6) Which are the correct declarations of dynamic array of 5 integer numbers:
   a) `int a[5];`
   b) `int *a[5];`
   c) `int *a=malloc(5);`
   d) `int *a=(int*) malloc(20);`
   e) `int *a=(int*) malloc(5*sizeof(int));`
   f) `int *a=new int [5];`
   g) `int *a=new [5];`
   h) `int *a=new int (5);`

7) How can we free memory which was allocated with `new`?
8) Select correct commands to free memory from dynamic array `a` of 5 elements:
   a) `delete a[5];`
   b) `delete a[];`
   c) `delete []a;`
   d) `free (a);`
   e) `free a[5];`
9) What is the difference between `malloc()` and `calloc()`?
10) What is the `realloc()` function used for?

Table 8.1

№ var.	Individual problems
1	Input a sequence of integer numbers (array a) and create dynamic array (array b) of numbers with value smaller than 6. Write a function to calculate an average of array b
2	Input a sequence of real numbers (array a) and create dynamic array (array b) of positive numbers. Write a function to find the smallest element of array b
3	Input a sequence of real numbers (array a) and create dynamic array (array b) of numbers before the first negative number (or all numbers in case if there is no negative number). Write a function to sort array b on ascend
4	Input a sequence of integer numbers (array a) and create dynamic array (array b) of even non-zero numbers. Write a function to swap the biggest and the smallest elements of array b
5	Input a sequence of numbers (array a) and create dynamic array (array b) of numbers before the first number with zero value or all numbers (in case if there is no zero). Write a function to calculate the product of elements with absolute value smaller than 10
6	Input a sequence of real numbers (array a) and create dynamic array (array b) of numbers with absolute value not bigger than 8. Write a function to calculate an average of the smallest and the biggest elements
7	Input a sequence of integer numbers (array a) and create dynamic array (array b) of numbers before the first three-digit number or all numbers (in case if there is no three-digit number). Write a function to calculate an average of odd elements
8	Input a sequence of real numbers (array a) and create dynamic array (array b) of numbers from [-4, 7). Write a function to find the sum of array b
9	Input a sequence of real numbers (array a) and create dynamic array (array b) of numbers before the first number=100 (or all numbers in case if there is no negative number). Write a function to find maximum of array b
10	Input a sequence of integer numbers (array a) and create dynamic array (array b) of numbers which are multiple to 3 and 4. Write a function to find the sum of negative elements

11	Input a sequence of numbers (array a) and create dynamic array (array b) of numbers after the first two-digit number or all numbers (in case if there is no zero). Write a function to calculate the quantity of negative elements
12	Input a sequence of real numbers (array a) and create a dynamic array (array b) of numbers with values from the interval [10, 25]. Write a function to find the number of elements with values bigger than the value of the first element of array b
13	Input a sequence of numbers (array a) and create a dynamic array (array b) of odd numbers. Using the function change all negative elements to zeros
14	Input a sequence of real numbers (array a) and create a dynamic array (array b) of numbers with absolute values outside the interval (20, 40]. Using the function, calculate the number of elements less than the average of all elements
15	Input a sequence of numbers (array a) and create a dynamic array (array b) of nonzero numbers. Use the function to find the largest of the even elements
16	Input a sequence of real numbers (array a) and create a dynamic array (array b) of numbers with absolute values from the interval [5, 50). Use a function to find the minimal positive element
17	Input a sequence of numbers (array a) and create a dynamic array (array b) of numbers whose values do not exceed 100. Using a function, calculate the sum of two-digit elements
18	Input a sequence of numbers (array a) and create a dynamic array (array b) of the numbers after the first three-digit number. Using the function, calculate the quantity of elements that are multiples of 5

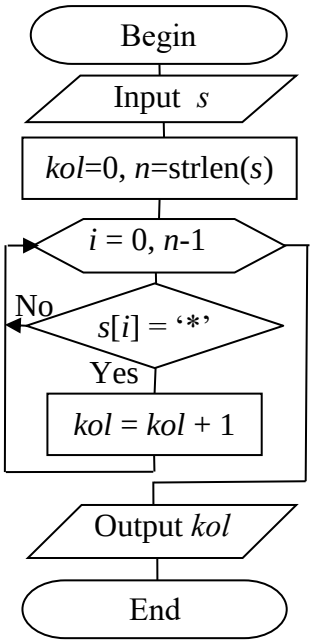
Lab № 9
Characters (char) and C-strings (char*)

Goal: to get practical skills of writing programs with characters and c-strings.

Examples of programs

Example 9.1. Enter a string and calculate a quantity if * in it.

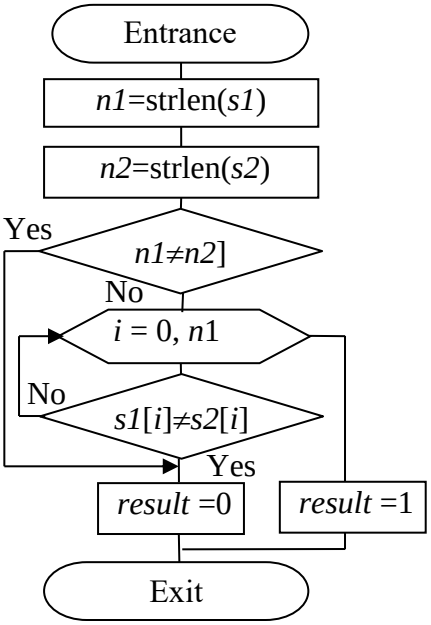
```
#include <iostream>
using namespace std;
int main()
{
  char s[50];
  int i, n, kol = 0;
  puts("Enter a string:  ");  gets_s(s);
  n = strlen(s);      //Length of the string
  for (i = 0; i < n; i++)
    if (s[i] == '*') kol++;
  printf("Quantity of symbols * in the string = %d", kol);
  system("pause");
  return 0;
}
```



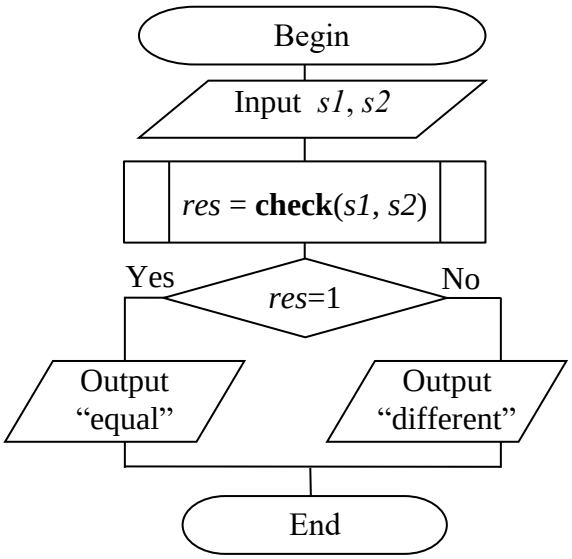
Result:

```
Enter a string:
q***H****f**
Quantity of symbols * in the string = 9
```

Example 9.2. Enter two strings and define whether they are equal or different.



Flowchart of function **check()**



Flowchart of the **main** function

```
#include <iostream>
using namespace std;
int check(char* s1, char* s2)
{
  int i, j, n1, n2;
  n1 = strlen(s1); n2 = strlen(s2);
  if (n1 != n2) return 0;
  for (i = 0; i < n1; i++)
    if (s1[i] != s2[i]) return 0;
  return 1;
}
int main()
{
  char s1[50], s2[50];   int res;
  puts("Enter the first string");
  gets_s(s1);
  puts("Enter the second string");
  gets_s(s2);
  res = check(s1, s2);
  if (res) cout << "The strings are equal" << endl;
  else cout << "The strings are different" << endl;
  system("pause");
  return 0;
}
```

Results:

```
Enter the first string
Hello world
Enter the second string
Hello world!
The strings are different
```

Example 8.3. Enter a string and define whether it contains punctuation marks.

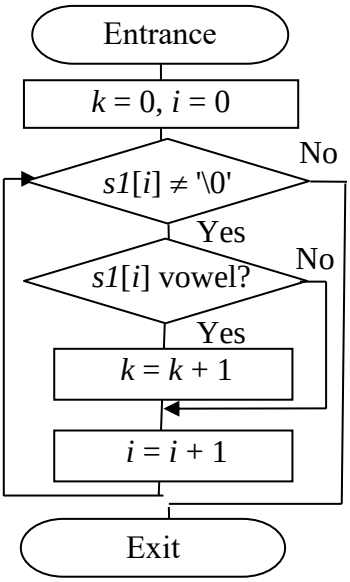
```
#include <iostream>
using namespace std;
bool punkt(char* s)
{
  int i;
  for (i = 0; s[i] != '\0'; i++)
    if (s[i]=='.' || s[i]==',' || s[i]==';' || s[i]=='!' || s[i]=='?' ||
        s[i]=='-')   // if ( ispunct ( s[i] ) )
      return 1;
  return 0;
}

int main()
{
  char s[100];
  puts("Enter a string");  gets_s(s);
  int res = punkt(s);
  if (res)
    cout << "The string contains punctuation marks" << endl;
  else
    cout << "Punctuation marks are absent" << endl;
  system("pause");
```

```
    return 0;
}
```

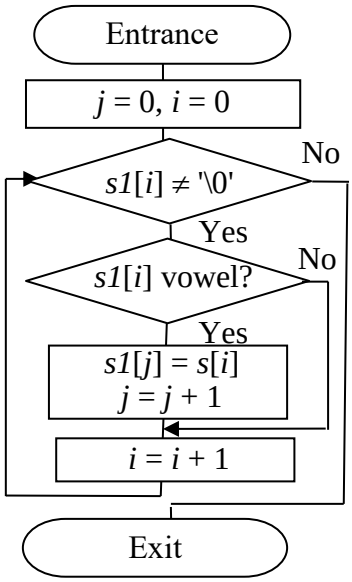
Results:

```
Enter a string
For example, the length of a string can be found with the length() function:
The string contains punctuation marks

Enter a string
You can access the characters in a string by referring to its index number
inside square brackets
Punctuation marks are absent
```

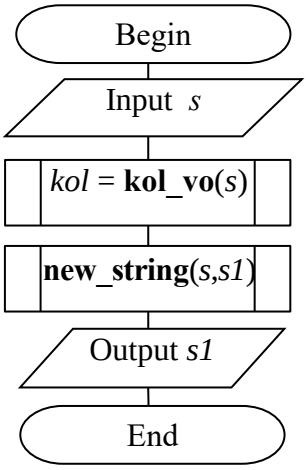
Example 8.4. Enter a string. Create a new string with the vowels.



Flowchart of function **kol_vo()**



Flowchart of function **new_string()**



Flowchart of the **main** function

```
#include <iostream>
using namespace std;
int kol_vo(char *s)
{
  int k = 0, i;
  for (i = 0; s[i] != '\0'; i++)
    if (s[i]=='a' || s[i]=='e' || s[i]=='i' || s[i]=='o' || s[i]=='u')
      k++;
  return k;
}
void new_string(char * s, char *s1)
{
  int i, j = 0;
  for (i = 0; s[i] != '\0'; i++)
    if (s[i]=='a' || s[i]=='e' || s[i]=='i' || s[i]=='o' || s[i]=='u')
    {
      s1[j] = s[i];   j++;
    }
   s1[j] = '\0';
}
```

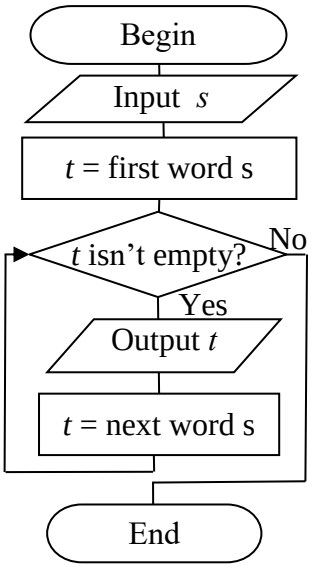
```
int main()
{
  int i, kol = 0;  char * s = new char[100];
  puts(" Enter a string:   ");  gets_s(s, 100);
  kol = kol_vo(s);
  char * s1 = new char[kol + 1];
  new_string(s, s1);
  puts("\n New string from vowels  ");  puts(s1);
  delete[] s; delete[] s1;
  system("pause");
  return 0;
}
```

Results:

```
Enter a string:
You can access the characters in a string by referring to its index number
inside square brackets

 New string from vowels
Ouaaeeaaeiaieeioiieueiieuaeae
```

Example 8.5. Enter a string and output all words in a column.

```
#include <iostream>
using namespace std;
int main()
{
  char s[100], *t;
  puts(" Enter a string:");  gets_s(s);
  cout << "\n Words: \n";
  t = strtok(s, " .,;?!-");
  while (t != NULL)
  { puts(t);
    t = strtok(NULL, " .,;?!-");
  }
  system("pause");
  return 0;
}
```



Results:

```
Enter a string:
You can access the characters in a string by referring to
its index number inside square brackets

 Words:
You
can
access
the
characters
in
a
string
by
referring
to
its
index
number
inside
square
brackets
```

Example 8.8. Enter a string. Delete all short words (with length < 4) in this string.

```
#include <iostream>
using namespace std;
int  main()
{
  char * s = new char[100], *t;
  char * s1 = new char[100];
  strcpy(s1, "\0");
  puts(" Input a string:");   gets_s(s, 100);
  t = strtok(s, " .,;?!-");
  while (t != NULL)
  {
    if (strlen(t) >= 4)
    {
      strcat(s1, t);
      strcat(s1, " \0");
    }
    t = strtok(NULL, " .,;?!-");
  }
  puts("\n String without short words");
  strcpy(s, s1);
  puts(s);
  delete[] s;  delete[] s1;
  system(" pause");
  return 0;
}
```

Results:

```
Input a string:
You can access the characters in a string by referring to its index number inside
square brackets

 String without short words
access characters string referring index number inside square brackets
```

Control questions

1) What is ANSI-table? How many characters are in this table?
2) What is the size of `char` variable?
3) What operations with symbols do you know?
4) Which expressions are true?

а) `'0' < 'y'`	г) `'w' > 'W'`
б) `'5' > 'f'`	д) `' ' > '0'`
в) `'F' > 'f'`	е) `'я' > 'ю'`

5) How can we change the case of Latin and Cyrillic symbols in programs?
6) Write all ways of declaration of c-string.
7) Is it possible to assign a part of c-string to another c-string? Write an example.
8) Which function allows to add one string to another? Write an example.
9) Write all ways of input of c-strings in console mode.

Individual task

1. Create program and flowchart for problems from table 9.1. Do not use functions from string.h, except of strlen.
2. Create program and flowchart for problems from table 9.2. Use functions from string.h.

Table 9.1

№ var.	Individual problems
1	Input a string and define the number of low case letters in this string
2	Input a string and change the capital letters to spaces
3	Input a string and define the number of digits in this string
4	Input a string and change the Latin letters to exclamation mark
5	Input a string and change punctuation marks to symbol '#'
6	Input a string and define the number of vowels in this string
7	Input a string and output ANSI-codes instead of characters of this string
8	Input a string and define the number of punctuation marks in this string
9	Input a string and change the capital letters to spaces
10	Input a string and define the number of commas in this string
11	Input a string and change the Latin letters to '*'
12	Input a string and change punctuation marks to symbol '-'
13	Input a string and define the number of capital letters in this string
14	Input a string and output ANSI-codes of digits in this string
15	Input a string and calculate the quantity of hyphen symbol in the string
16	Input a string and create a new string of Latin letters
17	Input a string and calculate the quantity of “th” combination
18	Input a string and output all vowels

Table 9.2

№ var.	Individual problem
1	Input a string and delete words with digits from the string
2	Input a string and output the longest word of the string in the reverse order
3	Input a string and create a new string of words with length bigger than 6
4	Input a string and output all words, which start and end with the same symbol
5	Input a string and delete words with odd length
6	Input a string and output the shortest word of the string
7	Input a string and a word. Delete this word from the string
8	Input a string and output all words, which length is bigger than 7
9	Input a string and output all words, which start from letter K
10	Input a string and find the longest sequence of letters A and the length of this sequence
11	Input a string with parenthesis and output all symbols inside the parenthesis
12	Input a string and define, how many combinations "Microsoft" are in the string
13	Input a string and count the quantity of words, which start and end with the same letter (ignoring the case)
14	Input a string and define whether the combination "C++" presents in the string, and if yes, output the position of the first combination
15	Input a string and insert length of word after each word. Use function itoa to convert a number to C-string
16	Input a string and change symbols of arithmetic operations to their names (‘+’ – “plus” etc.)
17	Input a string and create a new string of words with hyphen
18	Input a string and insert symbol * before and after each word

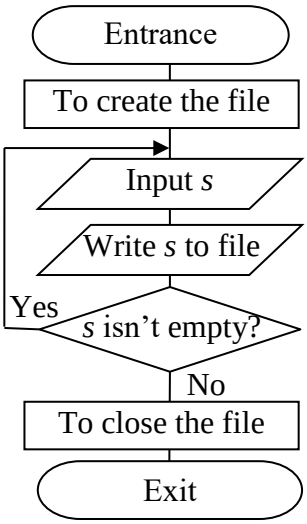
Lab 10

Text files

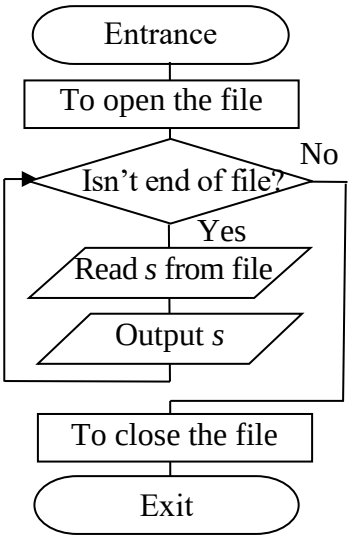
Goal: to get practical skills of creating and editing of text files in Visual C++.

Examples of programs

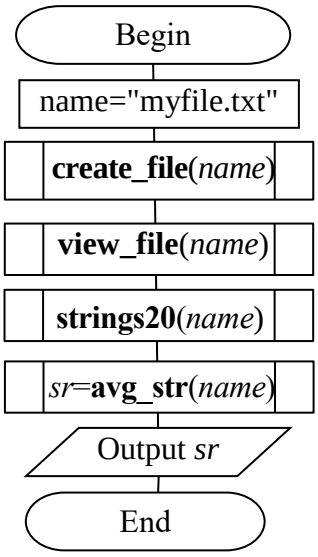
Example 1. Create a file with strings (the last string is an empty string). Output the file content. Output strings longer than 20 characters. Calculate average length of the strings.



Flowchart of function **create_file()**



Flowchart of function **view_file()**



Flowchart of the **main** function

Function Description `view_file` ()

Function `view_file()` displays the contents of the file.

Function parameter – string variable `name` – the name of the physical file on the disk.

The function is type of void, that is, it does not return a value of `main()`.

Local variables:

– `s` – the string up to 100 characters in length for reading lines from a file;

– `f` – the file variable with which has got the access to the file.

The algorithm of the function:

1. Open file:

```
f = fopen(name, "rt");
```

The `fopen` command opens the file. The first parameter (`name`) is the name of the file that is being opened, the second parameter (`"rt"`) is the opening mode: the file is opened as a text file for reading data.

2. Make sure the file is open. If not, display a message and interrupt the function:

```
if (f == NULL) { cout << "Cannot open file to veiw\n"; return; }
```

3. Read lines from a file until they are finished:

```
while (fgets(s, 100, f)>0)
```

The next line is read from the file using the command:

```
fgets(s, 100, f)
```

It has three arguments:

– `s` is a string variable that writes a string read from a file;

– `100` – maximum number of characters that can be read at a time;

– `f` – the file from which the data is read.

The function `fgets()` returns the actual length (number of characters) of the read string. That is, the loop will exit when `fgets()` returns a value of 0, which means that there are no more lines in the file.

4. To delete `\n` from the end of the string:

```
s[strlen(s) - 1] = '\0';
```

5. Display each scanned line **`s`**.

```
puts(s);
```

6. At the end of the cycle, close the file:

```
fclose(f);
```

The code:

```
#include <iostream>
using namespace std;

// Create file
void create_file(char* name)
{
  char s[100]; FILE* f;
  f = fopen(name, "wt");  // to open the file as text for creation
  // to check if file is opened
  if (f == NULL) { cout << "Cannot create file\n"; return; }
  cout << "Input strings" << endl;
  do {                    // while an empty string is not entered
    gets_s(s, 100);       // to enter a string from the keyboard
    fputs(s, f);          // to write the string to file
    fputs("\n", f);       // to move a cursor to the beginning of the next line
  } while (strcmp(s, ""));
  fclose(f);              // to close (and save) the file
}

// View file
void view_file(char* name)
{
  char s[100]; FILE* f;
  f = fopen(name, "rt");  // to open file as text for reading
  if (f == NULL) { cout << "Cannot open file to veiw\n"; return; }
  cout << "\nView file" << endl;
  while (fgets(s, 100, f))
  { // to read strings from file while not reached the end of file
    s[strlen(s) - 1] = '\0'; // to delete \n from the end of the string
    puts(s);                 //output a string
  }
  fclose(f);
}
```

```
// Output strings longer than 20 characters
void strings20(char* name)
{
  char s[100]; FILE* f;
  f = fopen(name, "rt");
  if (f == NULL)
    { cout << "Cannot open file\n"; return; }
  cout << "\nStrings with length bigger than 20 characters:" << endl;
  while (fgets(s, 100, f)>0)
  {
    s[strlen(s) - 1] = '\0';
    if (strlen(s) > 20)  puts(s);
  }
  fclose(f);
}

// Calculate average of the strings
double avg_str(char* name)
{
  char s[100]; FILE* f;
  int sum = 0, kol = 0; // kol is for total sum of strings' length
  f = fopen(name, "rt");
  if (f == NULL)
    { cout << "Cannot open file\n"; return 0; }
  while (fgets(s, 100, f))
  {
    sum += strlen(s)-1; // -1 is to delete \n from the end of the string
    kol++;
  }
  fclose(f);
  if (kol - 1)
    return (double)sum / (kol - 1); // average
  else
    return 0;
}

// Main function
int main()
{
  FILE * f = NULL;              //declaration of the file variable
  char name [] = "myfile.txt";  //name of the file on the HDD
  create_file(name);
  view_file(name);
  strings20(name);
  double sr = avg_str(name);
  cout << "\nAverage strings length=" << sr << endl;
  system("pause");
  return 0;
}
```

Results:

```
C-strings
Therefore, this array has a capacity to store sequences of up to 20 characters.
But this capacity does not need to be fully exhausted:
the array can also accommodate shorter sequences.
For example,
at some point in a program, either the sequence "Hello"
or the sequence "Merry Christmas" can be stored in foo,
since both would fit in a sequence with a capacity for 20 characters

View file
C-strings
Therefore, this array has a capacity to store sequences of up to 20 characters.
But this capacity does not need to be fully exhausted:
the array can also accommodate shorter sequences.
For example,
at some point in a program, either the sequence "Hello"
or the sequence "Merry Christmas" can be stored in foo,
since both would fit in a sequence with a capacity for 20 characters.

Strings with length bigger than 20 characters:
Therefore, this array has a capacity to store sequences of up to 20 characters.
But this capacity does not need to be fully exhausted:
the array can also accommodate shorter sequences.
at some point in a program, either the sequence "Hello"
or the sequence "Merry Christmas" can be stored in foo,
since both would fit in a sequence with a capacity for 20 characters.

Average strings length=48.5
```

Control questions

1) What file is called text file?
2) What is the difference between text and binary files?
3) Write methods of reading and writing data from/to text file.
4) What is the difference between methods `Write` and `WriteLine`?
5) Write command to create a text file.
6) Write command to remove a text file.

Individual task

Create files, which name is your surname and .rtf extension.

Table 10.1

№ var.	Individual problem
1	Calculate a quantity of strings with length bigger than 20
2	Calculate a quantity of strings, which end with punctuation marks
3	Output all strings, which do not contain parenthesis
4	Output all words with length bigger than 5, and calculate their quantity
5	Calculate the quantity of strings, which begin and end with the same letter
6	Output the longest word of the file and the length of this word
7	Output all strings, which do not contain punctuation marks
8	Output the shortest word of the file and the order number of its row
9	Calculate a quantity of strings, which contain digits
10	Output all strings, which start from the low-case letter
11	Calculate the quantity of punctuation marks in the file
12	Output all strings, which do not contain capital letters
13	Output all words, which contain letter ‘z’, and calculate their quantity
14	Calculate the quantity of strings which contain punctuation marks
15	Output the shortest words from each string of the file
16	Output all strings which do not contain digits
17	Calculate the quantity of strings, which start from the capital letter
18	Output all words with length smaller than 6 and calculate their quantity

Lab 11
Structures

Goal: to get practical skills of operation with structures and their fields.

Examples of programs

Example 11.1. Create a program for processing data with student session results: student name, group, and the results of the two exams. The program provides the possibility of inputting and displaying data on some number of students and the selection of data on students who have successfully passed the session and got the scholarship (have no unsatisfactory marks and an average score is more than 75 points), and also determine the name of the student with the highest average score. Use array of structures to store data.

The program code:

```
#include <iostream>
using namespace std;
struct student
{
  char surname[22], gr[8];
  int ex1, ex2;
};
int main()
{
  int kol = 0;                  // Total number of students
  cout << "Enter a number of students - "; cin>> kol;
  student *z = new student[kol]; //array of students
  cout << "Enter rows with information about " << kol
       << " students and their marks : \nSurname Group Mark1 Mark2" << endl;
  for (int i = 0; i < kol; i++)
    scanf("%s %s %i %i", z[i].surname, z[i].gr, &z[i].ex1, &z[i].ex2);
  cout << "\nMarks of "<< kol << " students: \nSurname \tGroup \tMark1 \tMark2\n";
  for (int i = 0; i < kol; i++)
    printf("%s\t%s\t%i\t%i\n", z[i].surname, z[i].gr, z[i].ex1, z[i].ex2);
  cout << "\nSuccessfully passed the session and got the scholarship: " << endl;
  int n = 0;                    // Quantity of students
  double sr, max(0);            // Average score and maximal mark
  char maxname[22];             // Surname of the student with maximal mark
  for (int i = 0; i < kol; i++)
  {
    if (z[i].ex1 >= 60 && z[i].ex2 >= 60 && (z[i].ex1 + z[i].ex2) / 2 >= 75)
    {
      n++;
      printf("%s\t%s\t%i\t%i\n", z[i].surname, z[i].gr, z[i].ex1, z[i].ex2);
    }
    sr = (z[i].ex1 + z[i].ex2) / 2.0;
    if (sr > max)
    { max = sr;
      strcpy(maxname, z[i].surname);
    }
  }
```

```
  cout << endl << "Quantity of students: " << n << endl;
  cout << maxname << " has maximal average mark " << max << endl;
  system("pause");
  return 0;
}
```

Results:

```
Enter a number of students - 4
Enter rows with information about 4 students and their marks :
Surname      Group        Mark1     Mark2
Shevchenko   IPZ-1.1      65        90
Ivanov       IK-1.1       50        60
Chumak       IPZ-1.2      75        85
Grib         IPZ-1.1      20        65

Marks of 4 students:
Surname      Group        Mark1     Mark2
Shevchenko   PZ-1.1       65        90
Ivanov       IK-1.1       50        60
Chumak       IPZ-1.2      75        85
Grib         IPZ-1.1      20        65

Successfully passed the session and got the scholarship:
Shevchenko   IPZ-1.1      65        90
Chumak       IPZ-1.2      75        85

Quantity of students: 2
Chumak has maximal average mark 80
```

Example 11.2. Write information about 3 books to the file. View this file. Find the newest book.

The code:

```
#include <iostream>
using namespace std;
int main()
{
 struct book
{
  char author[51];
  char bname[51];
  int year;
  float price;
};
  FILE * f;
  book z, maxyear;          //z is a book, maxyear is a book with maximal year
  //File creation
  f = fopen("myfile.txt", "wt");  // To open file for creation
  if (f == NULL)            // Test if file is opened successfully
    { cout << "Can't open the file\n"; return 0; }
  puts("Input information about 3 books");
  printf("Author\tBook\tYear\tPrice\n");
  for (int n = 0; n < 3; n++)
  {
    // To read an information about a book from the keyboard
    scanf("%s %s %d %f", z.author, z.bname, &z.year, &z.price);
```

```
    //To write inputted information to file
    fprintf(f, "%s\t|%s\t|%d\t|%5.2f\n", z.author, z.bname, z.year, z.price);
  }
  fclose(f);

  //Viwe file
  puts("\nView file");
  f = fopen("myfile.txt", "rt"); //To open file for reading
  if (f == NULL)
    {  cout << "Can't open the file\n"; return 0; }
  while (!feof(f)) //While the end of file is not reached
  { //To read an information about a boo from the file to variable z
    fscanf(f, "%s\t|%s\t|%d\t|%f\n", z.author, z.bname, &z.year, &z.price);
    //To output the z variable to the screen
    printf("%s\t|%s\t|%d\t|%5.2f\n", z.author, z.bname, z.year, z.price);
  }
  fclose(f);

  // Search for a book with maximal year
  maxyear.year = 0;//Initial value for maximal year
  f = fopen("myfile.txt", "rt");
  if (f == NULL)
    { cout << "Can't open the file\n"; return 0; }
  while (!feof(f))
  {
    fscanf(f, "%s\t|%s\t|%d\t|%f", z.author, z.bname, &z.year, &z.price);
    // If a year of the current book is bigger than maximal year
    if (z.year > maxyear.year)
      maxyear = z;      // Assign the current book to maxbook
  }
  fclose(f);
  if (maxyear.year > 0) // If maximal year is found
  { // Output the information about it
    puts("\nThe newest book");
    printf("%s\t|%s\t|%d\t|%5.2f\n", maxyear.author, maxyear.bname,
            maxyear.year, maxyear.price);
  }
  system("pause");
  return 0;
}
```

Results:

```
Input information about 3 books
Author  Book    Year    Price
Shevchenko Kobzar 2010 150
Rowling Harry_Potter 2016 340
Straustrup The_C++_Programming_Language 2012 275

View file
Shevchenko    |Kobzar                          |2010     |150.00
Rowling       |Harry_Potter                    |2016     |340.00
Straustrup    |The_C++_Programming_Language    |2012     |275.00

The newest book
Rowling       |Harry_Potter                    |2016     |340.00
```

Control questions

1) Give a definition of the structure as a data type.
2) How can we access to structure fields?
3) How can we define the size of memory, which is necessary to store the structure?
4) Is it correct to give the same names for variables and structure fields?
5) Write declaration of the structure, which describes for some electric device the following characteristics: device name, power consumption and rated voltage.

Individual task

1. Enter array of five structures and solve individual problem (Table 11.1).
2. Create text file with information about several structures and solve individual problem (Table 11.2).

Table 11.1

№ var.	Structure fields	Individual problem
1	*Information about students' exams:* – surname, – group, – physics – informatics – history	Define the average mark of each student and select students with average marks bigger than 75
2		Select students with at least one unsatisfactory mark and define their quantity
3		Output all students who passed the session and their average marks. Sort them in descending order
4	*Information about employees:* – surname, – position, – education – birth year, – salary	Select employees younger than 30 years and calculate their quantity
5		Select employees, without high education and calculate their percentage in all employees
6		Define all the oldest and the youngest employees
7		Calculate the average salary and select all employees with salary higher than average
8	*Information about products*: – name, – manufacturer, – price, – quantity	Define the most expensive product, output full information about it and calculate its total cost
9		Calculate the total quantity and the total cost of all goods
10		Calculate the average price and select all goods with price lower than average
11		Input a manufacturer and select all goods of this manufacturer and their quantity and average price

№ var.	Structure fields	Individual problem
12	*Information about TV-programs*: – name, – frequency (times a week), – rating	Select all programs with frequency 1 time a week and define the most popular of them
13		Select all programs with frequency more than 3 times a week and sort them by rating
14		Calculate the average rating and select all programs with rating higher than average
15		Define the most popular TV-program and sort programs by name AZ
16	*Information about books in a library*: – author, – name, – publishing year, – number of pages	Define the oldest book and output all books of the given author
17		Calculate an average number of pages and find the most new book
18		Calculate a quantity of books, which were published after 2000. Sort books by number of pages on descending order.

Table 11.2

№ var.	Content of the file	Problem
1	List for automobile inspection: information about the stolen cars: state number, brand of car, color, date of application (3 fields: day, month, year)	Output information about stolen cars “BMW”, which were stolen in the current year
2	Information about the payment for building services: street, house number, surname of habitant, date of payment (3 fields: day, month, year), debt	Output information about debtors from Pushkinskaya street, with a debt over 500 ₴
3	Summer curriculum of trains: number of train, place of departure, place of destination, departure time (3 fields: hours, minutes, seconds)	Output information about the trains of direction Odessa - Kyiv, that leave from 10:00 to 17:00
4	Log of events of the operating system: the name of a start program; level of event (error, warning and others like that); date of event (3 fields: day, month, year); time of event (3 fields: hours, minutes, seconds)	Output information about errors and define how many days passed from each of errors to current moment of time
5	Information about medications in a pharmacy: the name of medications, expiry of their term (3 fields: day, month, year), price	Output data about medications, which expiry term is in a current year
6	List of employees of an enterprise: table number, surname, sex (m/f), birth year, position	Output information about all employees-women older than 55 years and men older 60 years

№ var.	Content of the file	Problem
7	Information about products: name, date of producing (3 fields: day, month, year), term of realization, price	Output information about products which were produced today
8	The list of employees: the log number, surname, position, year of birth, year of recruitment	Output information about employees who have worked for more than ten years at the enterprise
9	List of holidays in the calendar: holiday name, date (3 fields: day, month, year)	Display information about winter holidays
10	Schedule of airplanes: flight number, destination, departure time (3 fields: hours, minutes, seconds)	Output information about all flights before 10 am.
11	Student group list: surname, name, year of birth, average mark	Find a student with the highest average mark
12	Repertoire of the opera house: name of the play, genre, date (3 fields: day, month, year), beginning	Display the information about the ballets, which have not yet taken place, as well as children's plays (beginning up to 15 hours)
13	Student group list: surname, name, date of birth (3 fields: day, month, year), average mark	Output information about students who have birthday in the current month
14	The list of employees: the log number, surname, position, year of birth, year of recruitment	Output information about engineers
15	Information about products: name, date of producing (3 fields: day, month, year), term of realization, price	Output information about products with the term of realization today
16	Summer curriculum of trains: number of train, place of departure, place of destination, departure time (3 fields: hours, minutes, seconds)	Output information about the trains, which are departing now
17	Information about products: name, date of producing (3 fields: day, month, year), term of realization, price	Output information about products with word “chocolate” in the name
18	Information about products: name, date of producing (3 fields: day, month, year), term of realization, price	Output information about products which were produced last year

Lab 12
Binary files

Goal: to get practical skills of creation and edition of binary files in Visual C++.

Examples of programs

Example 12.1. To create the binary file with at least 10 records. A structure consists of the fields: *Surname*, *Name*, *Patronymic*, *Position*, *Year of employment*, *Salary*. View the file. Select data about engineers who worked more than 5 years. Create a text file with information about employees with salary smaller than average.

The program code:

```
#include <iostream>
using namespace std;

struct employee
{
  char surname[30];
  char name[30];
  char patronymic[30];
  char position[30];
  int start_year;
  float salary;
};

//To add one record to a binary file
void add_record(char* name)
{
  employee z;
  FILE* f;
  f = fopen(name, "ab");
  if (f == NULL)
    {  cout << "Cannot create file to add record\n"; return; }
  scanf("%s\t%s\t%s\t%s\t%i\t%f", z.surname, z.name, z.patronymic, z.position,
        &z.start_year, &z.salary);
  fwrite(&z, sizeof(employee), 1, f);
  fclose(f);
}
//View the binary file
void view_file(char* name)
{
  employee z;
  FILE* f;
  f = fopen(name, "rb");
  if (f == NULL) { cout << "Cannot open file to veiw\n"; return; }
  cout << "\nView binary file" << endl;
  cout << "Surname\tName\tPatronymic\tPosition\tStart date\tSalary\n" << endl;
  while (fread(&z, sizeof(employee), 1, f))
    printf("%s\t%s\t%s\t%s\t%i\t%6.2f\n",   z.surname,   z.name,   z.patronymic,
           z.position, z.start_year, z.salary);
  fclose(f);
}
```

```
//To select data about the engineers who worked for more than 5 years.
void select_data(char* name)
{
  employee z;
  FILE* f;
  int curr_year = 2020;
  f=fopen(name, "rb");
  if(f==NULL)
  {  cout<<"Cannot open file to veiw\n"; return;}
  cout<<"\nInformation about engineers who worked for more than 5 years"<<endl;
  cout<<"Surname\tName\tPatronymic\tPosition\tStart date\tSalary\n"<<endl;
  while (fread(&z, sizeof(employee), 1, f))
  {
    if (strcmp(z.position, "engineer")==0 && curr_year - z.start_year > 5)
      printf("%s\t%s\t%s\t%s\t%i\t%6.2f\n",  z.surname,  z.name,  z.patronymic,
             z.position, z.start_year, z.salary);
  }
  fclose(f);
}

//Create the text file with information about employees with salary smaller than average
void create_text_file(char* bname, char* tname)
{
  employee z;
  FILE* fb, *ft;
  float sum = 0, avg;
  int k = 0;
  fb = fopen(bname, "rb");
  if (fb == NULL)
  {  cout << "Cannot open file\n";
     return;
   }
  while (fread(&z, sizeof(employee), 1, fb))
  {
    sum += z.salary;
    k++;
  }
  if (k) avg = sum / k;
  else avg = 0;
  cout << "\nAverage salary = " << avg << endl;
  fseek(fb, 0, 0);           //To return to the beginning of the file
  ft = fopen(tname, "wt");   //To open ft file for creation
  if (ft == NULL)
  {  cout << "Cannot create text file\n"; return; }
  // To read data from the file fb to variable z while not end of file
  while (fread(&z, sizeof(employee), 1, fb))
  { // If a salary of the current employee is less than average,
    if (z.salary < avg)
    { // To write z variable to file ft
      fprintf(ft, "%s\t%s\t%s\t%s\t%i\t%g\n", z.surname, z.name, z.patronymic,
              z.position, z.start_year, z.salary);
    }
  }
  fclose(ft);
  fclose(fb);
}
```

```
//View text file
void view_text_file(char* name)
{
  employee z;
  FILE* f;
  f = fopen(name, "rt");
  if (f == NULL) {  cout << "Cannot open text file to veiw\n"; return; }
  cout  <<  "\nView  text  file  with  information  about  employees  with  salary
          smaller than average" << endl;
  cout << "Surname\tName\tPatronymic\tPosition\tStart date\tSalary\n" << endl;
  while (!feof(f))
  {
    fscanf(f, "%s\t%s\t%s\t%s\t%i\t%f\n", z.surname, z.name, z.patronymic,
           z.position, &z.start_year, &z.salary);
    printf("%s\t%s\t%s\t%s\t%i\t%6.2f\n", z.surname, z.name, z.patronymic,
           z.position, z.start_year, z.salary);
  }
  fclose(f);
}
//Main file
int main()
{
  char name[] = "binaryfile.txt", tname[]="textfile.txt";
  cout<<"Input information about employees:"<<endl;
  cout<<"Surname\tName\tPatronymic\tPosition\tStart date\tSalary\n"<<endl;
  for(int i=0; i<6; i++) add_record(name);
  view_file(name);
  select_data(name);
  create_text_file(name, tname);
  view_text_file(tname);
  system("pause");
  return 0;
}
```

Test data:

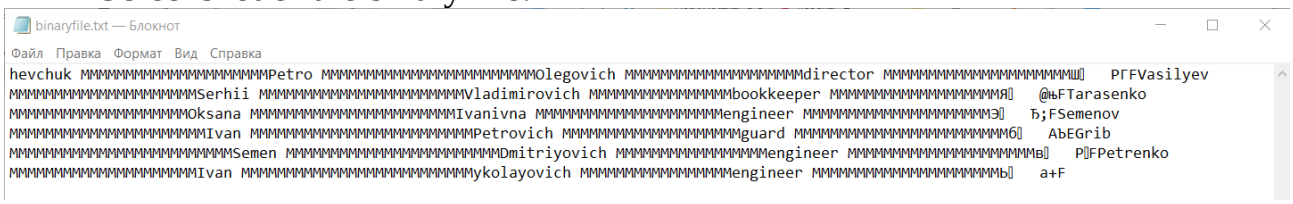
```
Shevchuk    Petro   Olegovich       director    2008  25000
Vasilyev    Serhii  Vladimirovich   bookkeeper  2015  20000
Tarasenko   Oksana  Ivanivna        engineer    2013  12000
Semenov     Ivan    Petrovich       guard       2017  7000
Grib        Semen   Dmitriyovich    engineer    2018  8500
Petrenko    Ivan    Mykolayovich    engineer    2012  11000
```

Results:

```
View binary file
Surname     Name    Patronymic      Position        Start date      Salary

Shevchuk    Petro   Olegovich       director    2008  25000.00
Vasilyev    Serhii  Vladimirovich   bookkeeper  2015  20000.00
Tarasenko   Oksana  Ivanivna        engineer    2013  12000.00
Semenov     Ivan    Petrovich       guard       2017  7000.00
Grib        Semen   Dmitriyovich    engineer    2018  8500.00
Petrenko    Ivan    Mykolayovich    engineer    2012  11000.00

Information about engineers who worked for more than 5 years
Surname Name    Patronymic      Position        Start date      Salary

Tarasenko   Oksana  Ivanivna        engineer        2013    12000.00
Petrenko    Ivan    Mykolayovich    engineer        2012    11000.00
```

```
Average salary = 13916.7
View text file with information about employees with salary smaller than average
Surname      Name    Patronymic      Position      Start date   Salary

Tarasenko    Oksana  Ivanivna        engineer        2013     12000.00
Semenov      Ivan    Petrovich       guard           2017     7000.00
Grib         Semen   Dmitriyovich    engineer        2018     8500.00
Petrenko     Ivan    Mykolayovich    engineer        2012     11000.00
```

Screenshot of the binary file:



Control questions

1) What file is called binary?
2) What is the difference between opening modes `OpenOrCreate` and `Open`?
3) Write instruction to create binary file.
4) Write instruction to move 1) to the beginning of binary file; 2) to the end of binary file. In which situations we use these commands?

Individual task

Create binary file, view it and select (output) records according to your individual variant (table 12.1). Write selected records to the text file.

Table 12.1

№ var.	Content of the binary file	Problem
1	List of students in a group: log number, last and first names, marks in physics, math and philosophy	Select students, who has no “3” marks, and average ball of each student
2	Information about cars for sale: type of car, type of engine, mileage of run, year of producing, starting price	Select the Ford cars with mileage of run less than 50000 km, which were produced more than two years ago
3	List of goods in the shop of electronics: code, name, producer, country-producer, year of producing, price	Select all refrigerators and calculate an average price of China TV
4	List of workers of enterprise: table number, last name and initials, position, date of employment, salary	Output information about a programmer with the biggest salary, and select all managers
5	List of books in a library: inventory number, name of book, author, year of publishing, price	Select books with names, which begin with a word "Programming", and also books that were published more than 10 years ago
6	List of subscribers of telephone station: the last name of subscriber, telephone	Output information about subscribers that have a debt of more than 200 ₴.

№ var.	Content of the binary file	Problem
	number, license fee for a month, debt	Define the biggest debt
7	List of printers for a sale: type of printer, firm-producer, speed of work (an amount of pages is in a minute), cost of printer	Output information about printers of firm “HP”, that print more than 10 pages in a minute
8	Results of competitions in athletics (100 m): name, gender, country, result (time)	Display information about Ukrainian athletes. Determine the leader in the distance of 100 m.
9	Data on employees: log number, name, position, gender, year of birth, marital status, number of children	Output information about single men. Calculate the number of employees who have more than 2 children
10	Results of the football tournament: team, country, city, number of victories, drawbacks and defeats	Output data about teams from Ukraine and calculate the total number of victories, drawbacks and defeats for them.
11	Results of athletics competitions: country, number of gold, silver and bronze medals	Calculate the total number of medals for each country. Identify the top three countries
12	List of students of the course: order number, surname and name, group, average point	Identify the best and worst students, and select students of inputted group
13	Employee data: log number, surname and name, position, gender, year of birth, marital status, number of children	Output information about single men. Calculate the number of employees with more than 3 children
14	Athletics Results (100m): surname, gender, country, result (time)	Output information about Ukrainian athletes. Define a leaders among men and women
15	Bus ticket price information: destination, price, bus model, departure time	Output information about bus to Kiev, and find the most expensive ticket
16	List of books in the bookstore: book title, author, year of publication, price	Print information about Taras Shevchenko's books published after 2000. Identify the most expensive book
17	Product list: serial number, name, manufacturer, price, quantity	Determine the total cost of each product and the total cost of all goods. Output the goods with the price over 1000 UAH.
18	Mobile phone store price list: firm, model, year, price	Output information first about all Nokia phones and then information on phones with a price of less than 2000 UAH

Recommended literature list

1. C++ // University of Cambridge: Department of Engineering: Computing Help. [Electronic source]. – Access mode: http://www-h.eng.cam.ac.uk/help/tpl/languages/C++.html.
2. C++ for C programmers // University of Cambridge: Department of Engineering. [Electronic source]. – Access mode: http://www-h.eng.cam.ac.uk/help/tpl/talks/CtoC++.html.
3. Basics of programming. Basic algorithms : methodical instructions for laboratory training and exercises / Prokop Y. V., Trofimenko E. G., Bukata L. N. – Part 1. – Odessa : ONAT named after A. S. Popov, 2016. – 76 p.
4. С++. Основи програмування. Теорія та практика: підручник / [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : Фенікс, 2010. – 544 с.
5. С++. Теорія та практика: навч. посіб. з грифом МОНУ / [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : ВЦ ОНАЗ, 2011. – 587 с.
6. С++. Алгоритмізація та програмування : підручник / О.Г. Трофименко, Ю.В. Прокоп, Н.І. Логінова, О.В. Задерейко. 2-ге вид. перероб. і доповн. – Одеса: Фенікс, 2019. – 477 с.
7. C++. Loops. // Wikiversity. [Electronic source]. – Access mode: https://en.wikiversity.org/wiki/C%2B%2B/Loops.
8. Control Statements // University of Cambridge: Department of Engineering: Computing Help: Languages: C++. [Electronic source]. – Access mode: http://www-h.eng.cam.ac.uk/help/languages/C++/c++_tutorial/control.html.
9. For statement (C++). [Electronic source]. – Access mode: https://msdn.microsoft.com/en-us/library/b80153d8.aspx.
10. Iteration Statements (C++). [Electronic source]. – Access mode: https://msdn.microsoft.com/ en-us/library/7ftwh93a.aspx.
11. Programing C++: Методическая система "Web-технологии". [Электронный ресурс]. – Режим доступа: http://metodweb.yomu.ru/en/oop-on-c.
12. The C++ Programming Language. [Electronic source]. – Access mode: http://www.stroustrup.com/C++.html#learning
13. Visual C++ .NET: пособие для разработчиков C++ / [А. Корера, С. Фрейзер, С. Джентайл и др.]. – М. : ЛОРИ, 2003. – 398 с.
14. Дейтел Х. М. Как программировать на C++; пер с англ. / Х. М. Дейтел, П. Дж. Дейтел. – М. : ООО "Бином-Пресс", 2008. – 1456 с.
15. Довбуш Г. Ф. Visual C++ на примерах / Г.Ф. Довбуш, А.Д. Хомоненко ; под ред. проф. А.Д. Хомоненко. – СПб. : БХВ-Петербург, 2007. – 528 с.
16. Зиборов В. В. MS Visual C++ 2010 в среде .NET. Библиотека программиста / Зиборов В. В. – СПб. : Питер, 2012. – 320 с.

17. Основи програмування. Базові алгоритми : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката. – Ч. 1. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2014. – 108 с.

18. Основи програмування. Опрацювання структурованих типів : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката. – Ч. 2. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2014. – 130 с.

19. Основи програмування. Програмне опрацювання файлів : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката. – Ч. 3. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2015. – 80 с.

20. Стивен Прата. Язык программирования C++. Лекции и упражнения : учебник ; пер. с англ. / Стивен Прата. – СПб.: ООО "ДиаСофтЮП", 2005. – 1104 с.

21. Страуструп Б. Язык программирования C++. Специальное издание; пер. с англ. / Страуструп Б. – М. : ООО "Бином-Пресс", 2006. – 1104 с.

22. Трофименко О. Г. Комп’ютерні технології та програмування. Базові алгоритми : метод. вказівки для лаб. і практ. робіт / Трофименко О. Г., Прокоп Ю. В., Буката Л. М. – Ч. 1. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2016. – 108 с.

23. Трофименко О. Г. Комп’ютерні технології та програмування. Програмування структурованих даних : метод. вказівки для лаб. і практ. робіт / Трофименко О. Г., Прокоп Ю. В., Буката Л. М. – Ч. 2. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2016. – 134 с.

24. Трофименко О. Г. Комп’ютерні технології та програмування. Програмне опрацювання файлів : метод. вказівки для лаб. і практ. робіт / Трофименко О. Г., Прокоп Ю. В., Буката Л. М. – Ч. 3. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2016. – 80 с.

25. Трофименко О. Г. Створення багатомодульних програмних проектів для опрацювання даних у файлах засобами Visual C++: метод. вказівки для виконання курсової роботи з дисципліни "Основи програмування" / Трофименко О. Г., Прокоп Ю. В. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2015. – 44 с.

26. Хортон А. Visual C++ 2010: полный курс.; пер. с англ. / Хортон А. – М. : ООО "И. Д. Вильямс", 2011. – 1216 с.

Content

Y. V. Prokop
L. N. Bukata
O. G. Trofymenko

ALGORITHMIZATION AND PROGRAMMING

Part 2

STRUCTURED DATA PROGRAMMING

Methodical instructions for laboratory training and exercises

Volume 3,56 printed sheets
ONAT named after A.S. Popov, 2020