# Computer Science. Module 3
# Laboratory works
# Table of contents

# Pointers. Dynamic arrays and matrices

## 1.1 Pointers

Pointer is a variable which value is an address of other variable.

There two types of memory: static (stack) and dynamic (heap). When we declare a variable:

int a;

computer automatically allocates 4 bytes of memory in stack for integer value and calls this memory ***a***. Address of this memory is ***&a***. Pointer on this variable:

```
int *p=&a;
```

Declaration of pointer consists of type of pointed variable, symbol ***** and pointer's name:

```
int *x;
float *b;
```

To allocate dynamic memory, we use **new** statement:

```
int *p = new int;        //allocates dynamic memory for integer number
int *q = new int (5);    //allocates dynamic memory for integer number and put 5 in it
```

We can refer to the value which stores in this memory: *p

```
*p = 2;          //puts 2 in memory with address p
(*q) += 2;       //increases by 2 value in memory with address q
```

To free memory, allocated with new command, we use delete:

```
delete p;
```

## 1.2 Dynamic arrays

Dynamic array is array, which:

- ✓ allocates in dynamic memory (heap);
- ✓ size may be a variable (its value must be defined in program before array's declaration)

Difference in operation with static and dynamic arrays is only in declaration of arrays and in memory clearing.

Declarations of double dynamic array of 10 elements:

I
```
double *a = new double [10];
```

or:

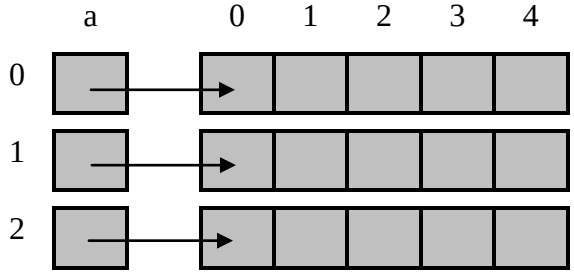
II
```
int N=10;        //N is an integer variable with value 10
double *a=new double [N];
```

To free memory from array, allocated with new command, we use delete []:

```
delete []a;
```

## 1.3 Dynamic matrix

To work conveniently with dynamic matrix, we declare it as several one-dimensional arrays:




Array a is one-dimensional array, which elements are pointers on one-dimensional arrays (matrix rows). Declaration of these arrays is:

```
//At first we declare auxiliary array (left array-column)
float **a= new float *[3];      //we write **: first * means that we declare dynamic array,
//second * shows that elements of this array are pointers on float numbers (pointer on array)
//Then in loop we declare each row separately
//and put addresses of first row elements in corresponded elements of array a:
for (int i=0; i<3; i++)
        a[i] = new float [5];
```

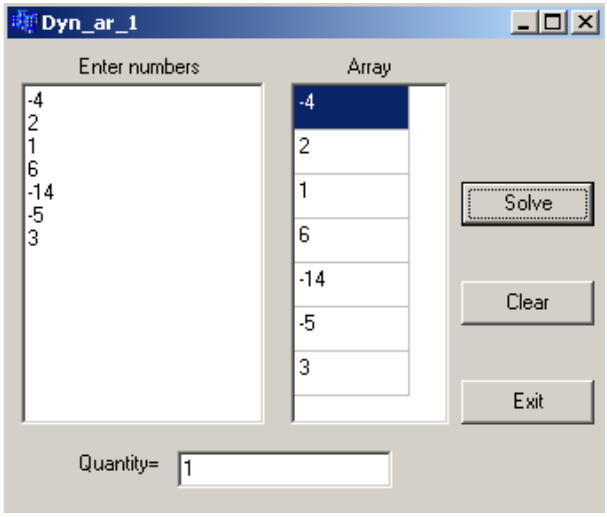
To free memory from dynamic matrix:

```
for (int i=0; i<3; i++)
        delete [] a[i];
delete []a;
```

### 1.4 Examples of programs with dynamic arrays and matrices

**Example 1**

Enter float numbers in Memo. Create array of non zero numbers. Calculate the quantity of negative numbers with odd indexes.



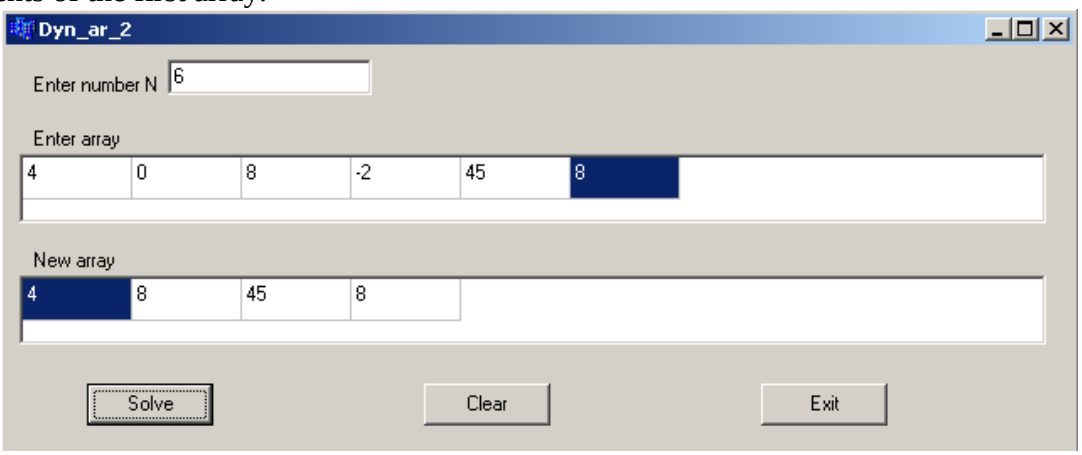

We do not know exactly how many numbers will be in Memo1. Therefore we cannot declare static array with constant number of elements and must use dynamic array. At first we calculate the real number of lines in Memo and after this can declare array.

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
int i, j=0;
int n=Memo1->Lines->Count;          //calculate the number of lines in Memo1
int N=0;
for (i=0; i<n; i++)                 //calculate the real number of elements in future array
   {x= StrToFloat(Memo1->Lines->Strings[i]);
    if (x!=0) N++;}
SG1->RowCount=N;                    //change StrinGrid's RowCount property to output future array
float *a=new float[N];              //declare dynamic array a
for (i=0; i<N; i++)
  { x= StrToFloat(Memo1->Lines->Strings[i]);
    if (x!=0) {a[j]=StrToFloat(Memo1->Lines->Strings[i]);        //read array elements
               SG1->Cells[0][i]=a[i];                          //and output them in StringGrid
               j++;
              }
   }
int k=0;                            //k is a quantity of negative elements with odd indexes
for (i=0; i<N; i++)
  if (a[i]<0 && i%2!=0) k++;
Edit1->Text=IntToStr(k);
delete []a;                         //free memory from dynamic array
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Edit1->Clear();
Memo1->Clear();
```

```
for (int i=0; i<SG1->RowCount; i++)
   SG1->Cells[0][i]="";
}
```

**Example 2**

Enter integer number N. Enter array of N double elements in StringGrid. Create the second array of the positive elements of the first array.




The number of elements must be entered from the form and we do not know its value writing the program. Therefore we cannot use static array. At first we must enter N and only then we can declare array ***a***. When we enter N, ColCount must be changed. Therefore we write commands to enter N and change ColCount in function of Edit1Change event.

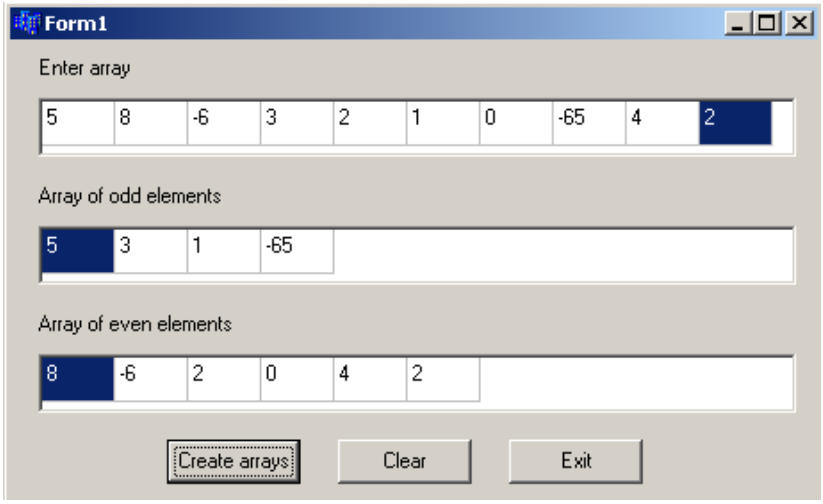
```
//when something is written in Edit1
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
    int N=StrToInt(Edit1->Text);      //input N
    SG1->ColCount=N;                  //change ColCount
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
int N=StrToInt(Edit1->Text);//read N again; N is a local variable and we can't use its value from Edit1Change
float *a=new float[N];          //declare a dynamic array, we cannot declare array b here, because
                                //in this moment we do not know how many elements will be in array b
int i;
for (i=0; i<N; i++)
  {a[i]=StrToFloat(SG1->Cells[i][0]);}
int k=0,j=0;
for (i=0; i<N; i++)
  if (a[i]>0) k++;
float *b=new float[k];                  //now we know k and can declare array b
SG2->ColCount =k;
for (i=0; i<N; i++)
  if (a[i]>0){b[j]=a[i]; j++;}          //calculate array b elements
for (i=0; i<k; i++)
  SG2->Cells[i][0]=FloatToStr(b[i]);
delete []a;                             //free memory from arrays a and b
delete []b;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Edit1->Clear();
for (int i=0; i<SG1->RowCount; i++)
```

```
   SG1->Cells[i][0]="";
for (int i=0; i<SG2->RowCount; i++)
   SG2->Cells[i][0]="";
}
```

**Example 3**

Enter array of 10 integer numbers. Create two new arrays: of odd elements of the first array and of even elements of it.

Change for all StringGrids property DefaultColWidth=40 to make cells more narrowly.



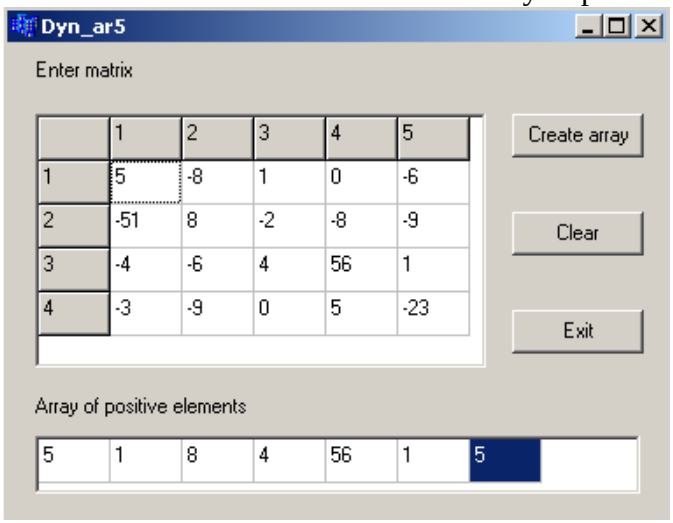

First array is static (it consists of 10 elements), two other arrays (b1 and b2) are dynamic. At first we must calculate the number of their elements (n1 and n2) and only after this can declare them.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
int a[10];
int i, n1=0, n2=0, j1=0, j2=0;
for (i=0; i<10; i++)
  a[i]=StrToInt(SG1->Cells[i][0]);
for (i=0; i<10; i++)              //calculate n1 and n2
  if (a[i]%2!=0) n1++;
  else n2++;
SG2->ColCount=n1;                 //set StringGrid's size for array b1
SG3->ColCount=n2;                 //set StringGrid's size for array b2
int *b1=new int [n1];             //declare arrays b1 and b2
int *b2=new int [n2];
for (i=0; i<10; i++)              //fill arrays b1 and b2
  if(a[i]%2!=0){b1[j1]=a[i]; j1++;}
  else {b2[j2]=a[i]; j2++;}
for (i=0; i<n1; i++)              //output array b1
  SG2->Cells[i][0]=IntToStr(b1[i]);
for (i=0; i<n2; i++)              //output array b2, we cannot output both arrays in one loop, because they have
  SG3->Cells[i][0]=IntToStr(b2[i]);          //different numbers of elements
delete []b1;                      //free memory from b1 and b2
delete []b2;
}

//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
```

```
{ int i;
for (i=0; i<10; i++)
  SG1->Cells[i][0]="";
for (i=0; i<SG2->ColCount; i++)
  SG2->Cells[i][0]="";
for (i=0; i<SG3->ColCount; i++)
  SG3->Cells[i][0]="";
}
//---------------------------------------------------------------------------
```

**Example 4**

Enter matrix 4x5 of double numbers. Create new one-dimensional array of positive matrix elements.




First StringGrid has 1 FixedRow and 1 FixedCol for order numbers of rows and columns. Therefore, matrix elements are in cells from [1][1] (not from [0][0]).

Matrix ***a*** is static, array ***b*** is dynamic.

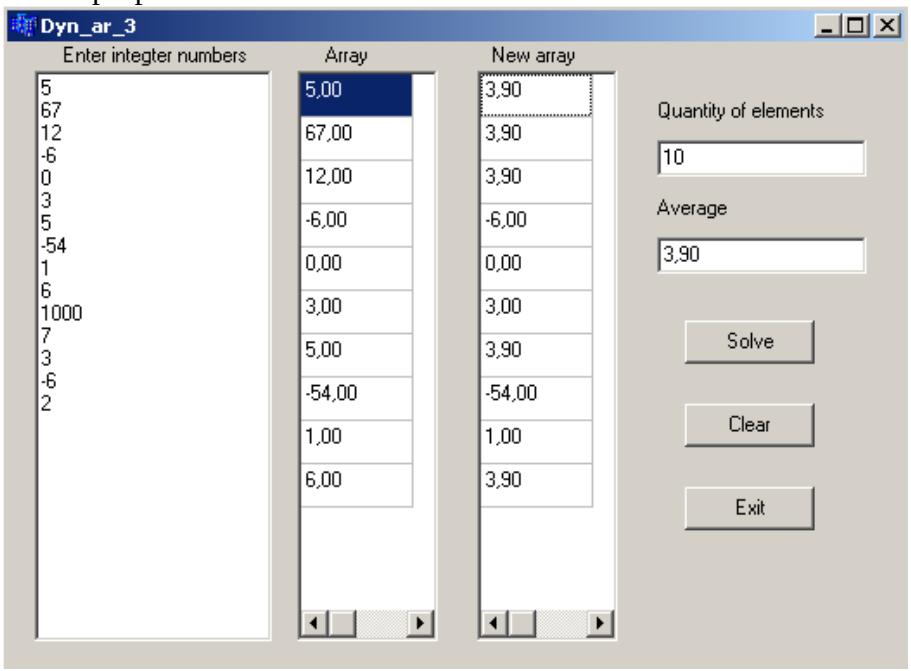
```
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{//when the form appears on the screen, fixed row and column must be filled with values
for (int i=1; i<=4; i++)
  SG1->Cells[0][i]=IntToStr(i);
for (int j=1; j<=5; j++)
  SG1->Cells[j][0]=IntToStr(j);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
double a[4][5];
int i, j, n=0, k=0;
for (i=0; i<4; i++)      //element has indexes [i][j], but cell has indexes [j+1][i+1] (they numerates from 1)
  for (j=0; j<5; j++)
    a[i][j]=StrToFloat(SG1->Cells[j+1][i+1]);
for (i=0; i<4; i++)      //calculate the number of elements of array b
  for (j=0; j<5; j++)
    if (a[i][j]>0) n++;
SG2->ColCount=n;
double *b=new double [n];    //declare array b
for (i=0; i<4; i++)
  for (j=0; j<5; j++)
    if(a[i][j]>0){b[k]=a[i][j]; k++;}
```

```
for (i=0; i<n; i++)
  SG2->Cells[i][0]=FloatToStr(b[i]);
delete []b
}

//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{ int i;
for (i=0; i<4; i++)
    for (j=0; j<5; j++)
       SG1->Cells[j][i]="";
for (i=0; i<SG2->ColCount; i++)
     SG2->Cells[i][0]="";

}
//---------------------------------------------------------------------------
```

**Example 5**

Enter float numbers in Memo. Create an array of these numbers, which go before the first number 1000. If there is no 1000 in Memo, create array of all numbers. Write **subroutines** to:

1. Calculate the quantity of array elements.
2. Fill in array from Memo (parameters are array and quantity of elements).
3. Calculate the average of array elements (array is a parameter). Calculate the average of created array using this function.
4. Output array elements in the StringGrid (parameters are: pointer to the StringGrid object (the StringGrid, which is filling), array and quantity of elements.
5. Write subroutine, which receives two parameters (float numbers), and if the first of parameters is bigger than the second one, changes the second parameter with the first one. Use this function to change in array elements, which are greater than average, with the value of average. New array output in the new StringGrid. For this purpose use the function from item 4.




```
//---------------------------------------------------------------------------
```

***//function for calculation of the number of elements before 1000***

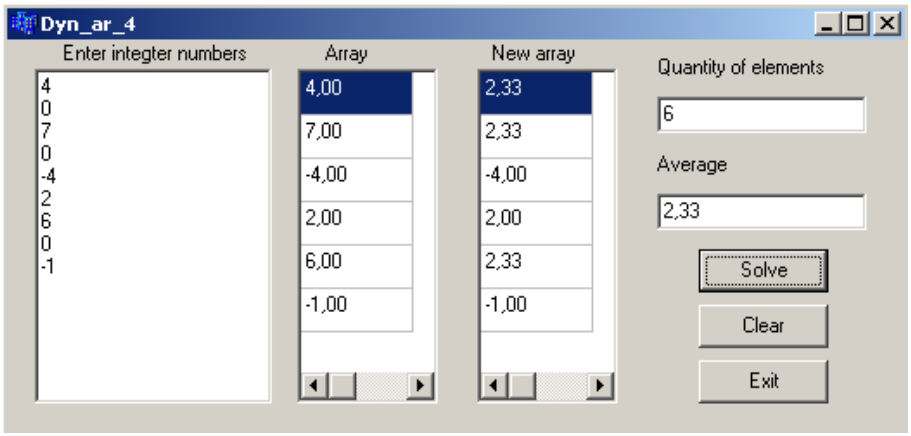
```
int koli4estvo(){
   int i;
   for(i=0; i<Form1->M1->Lines->Count; i++)       //look sequentially all numbers in Memo
```

```
      if (StrToFloat(Form1->M1->Lines->Strings[i])==1000)       //if the number with index
                                    //i is equal to 1000, then i is a quantity of numbers before 1000
         return i;                  //return i and interrupt the function execution
         //if the loop worked until its end and function execution was not interrupted, it means that
         //there is no 1000 in Memo and we must return the quantity of Memo lines
   return Form1->M1->Lines->Count;
}
//function for filling the array
void massiv(float arr [], int N){
  int i;
  for (i=0; i<N;i++){
    arr[i]=StrToFloat(Form1->M1->Lines->Strings[i]);
    }
}
//function for outputting array in StringGrid
void setka(TStringGrid *sg, float arr[], int N){
  for (int i=0; i<N; i++)
    sg->Cells[0][i]=FormatFloat("0.00",arr[i]);
}
//function for calculation the average of array elements
float sr_ar(float arr[], int N){
  float sum=0;
  int i;
  for (i=0; i<N; i++)
    sum+=arr[i];
  return (sum/N);
}
//function for changing two numbers, x – pointer, y - value
void zamena(float*x, float y){
  if(*x>y) *x=y;         //*x - dereference
}

/*
//function for changing two numbers, x – reference, y - value
void zamena1(float &x, float y){
  if(x>y) x=y;           //without dereference
}

*/
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
//Calculate the array elements
 int kol_el=koli4estvo();
//Output the quantity of array elements in Edit1:
Form1->Edit1->Text=IntToStr(kol_el);
//Change quantity of rows in StringGrids
  SG1->RowCount=kol_el;
  SG2->RowCount=kol_el;
//Memory allocation for array of kol_el float elements
  float *a=new float[kol_el];
//Fill the array
  massiv(a,kol_el);
//Output array a in SG1
```

```
  setka(SG1, a, kol_el);
//Calculate the average af array
  float srednee=sr_ar(a,kol_el);
  Edit2->Text=FormatFloat("0.00",srednee);
//Change the big elements with average value
  for(int i=0; i<kol_el; i++)
    zamena(&a[i],srednee);              //or the same is: zamena1(a[i],srednee);
//Output the changed array a in SG2
  setka(SG2, a, kol_el);
//free memory from array a
  delete []a;
}
//---------------------------------------------------------------------------
```

**Example 6**

Enter integer numbers in Memo. Create an array of non-zero numbers from Memo. Write **subroutines** 1-5 from the previous task.




This program differs from the previous in calculating the number of elements and filling the array.

***//function for calculation of the number of elements before 1000***

```
int koli4estvo(){
    int i, k=0;
    for(i=0; i<Form1->M1->Lines->Count; i++)
        if (StrToFloat(Form1->M1->Lines->Strings[i])!=0) k++;
    return k;
}
```

***//function for filling the array***

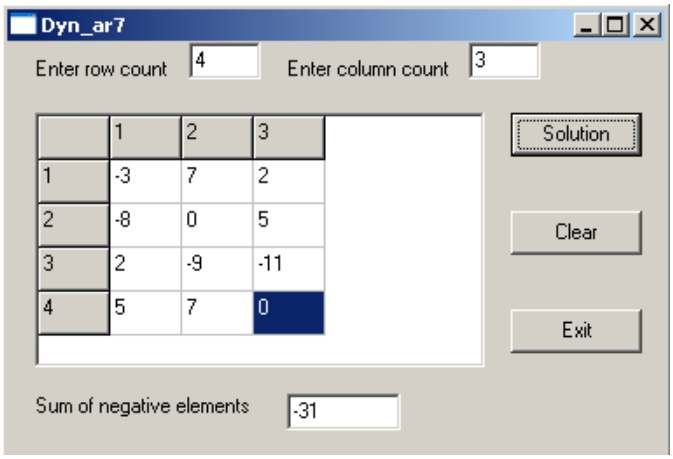
```
void massiv(float arr [], int N){
  int i, j=0, x;
  for (i=0; i<Form1->M1->Lines->Count;i++){
    x= StrToFloat(Form1->M1->Lines->Strings[i]);
    if (x!=0)
      {arr[j]=x; j++;}
    }
}
```

Other functions are the same as in previous program.

**Example 7**

Enter matrix dimension – number of rows and columns. Enter double matrix elements. Calculate the sum of negative elements.

We don't know the number of rows and columns of matrix. Therefore, matrix must be dynamic.

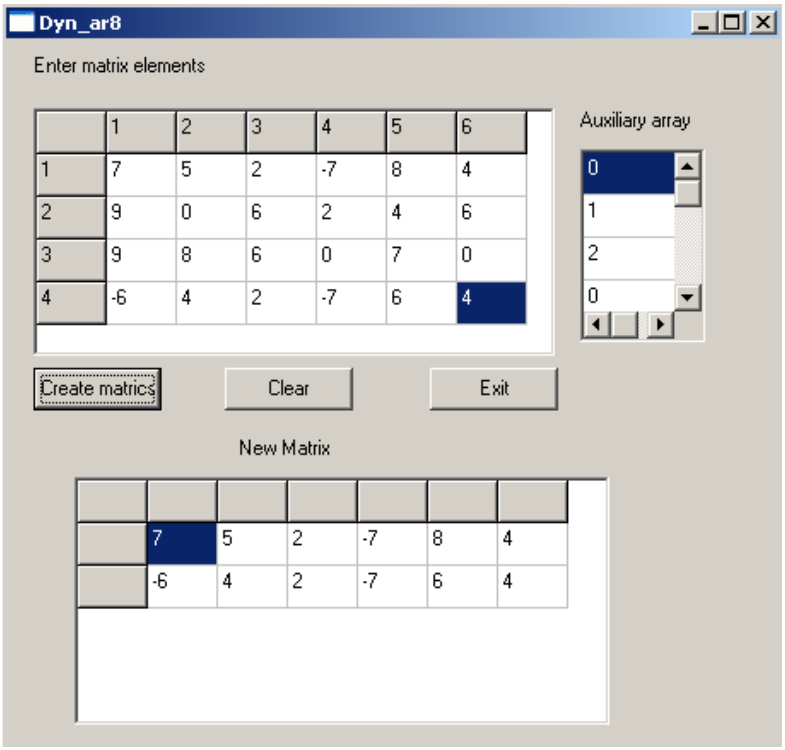



```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{                                   //we don't write order numbers of rows and columns at first
}
//---------------------------------------------------------------------------

int rowc, colc;    //global variables - matrix dimension
//---------------------------------------------------------------------------
//when we enter number of rows in Edit1
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
rowc=StrToInt(Edit1->Text);
SG1->RowCount=rowc+1;   //set RowCount
for (int i=1; i<=rowc; i++)     //and fill fixed column with numbers of rows
   SG1->Cells[0][i]=IntToStr(i);
}
//---------------------------------------------------------------------------
//when we enter number of columns in Edit2
void __fastcall TForm1::Edit2Change(TObject *Sender)
{
colc=StrToInt(Edit2->Text);
SG1->ColCount=colc+1;       //set ColCount
for (int j=1; j<=colc; j++)      //and fill fixed row with numbers of columns
  SG1->Cells[j][0]=IntToStr(j);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
int i, j;
double s=0;
double **a=new double *[rowc];      //declare auxiliary array of pointers on matrix rows
for(i=0; i<rowc; i++)
  a[i]=new double [colc];                //allocate memory for each row a[i] separately
for (i=0; i<rowc; i++)
  for (j=0; j<colc; j++)
    a[i][j]=StrToFloat(SG1->Cells[j+1][i+1]);
for (i=0; i<rowc; i++)
  for (j=0; j<colc; j++)
    if (a[i][j]<0) s+=a[i][j];
Edit3->Text=FloatToStr(s);
for (int i=0; i<rowc; i++)
```

```
    delete [] a[i];          //at first free memory from each matrix row a[i] separately
delete []a;                  //then free memory from auxiliary array a
}

//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{ int i,j;
for (i=1; i<=rowc; i++)
  for (j=1; j<=colc; j++)
    SG1->Cells[j][i]="";
Edit1->Clear();
Edit2->Clear();
Edit3->Clear();
}
```

**Example 8**

Enter integer matrix 4x6. Create new matrix with those rows of the first matrix, which do not contain zero elements.




First matrix is static. We do not know the number of rows of the second matrix. Therefore the second matrix is dynamic.

```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
for(int i=1; i<=4; i++)
  SG1->Cells[0][i]=IntToStr(i);
for(int j=1; j<=6; j++)
  SG1->Cells[j][0]=IntToStr(j);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
int i, j, rowc=0,k=0;
```

```
double s=0;
double a[4][6];
int r[4]={0};              //r is array of quantities of zero elements in matrix rows
for (i=0; i<4; i++)
  for (j=0; j<6; j++)
    a[i][j]=StrToFloat(SG1->Cells[j+1][i+1]);

for (i=0; i<4; i++)
  for (j=0; j<6; j++)
    if(a[i][j]==0) {r[i]++;}       //r[i] is a quantity of zero elements in i-th row
//output array r to control ourselves
for(i=0; i<4; i++)
  SG3->Cells[0][i]=IntToStr(r[i]);
//calculate the quantity of rows without zeroes
for(i=0; i<4; i++)
   if (r[i]==0) rowc++;           //r[i]=0 means that there is no zeroes in i-th row
SG2->RowCount=rowc+1;
//declare matrix b
double **b=new double *[rowc];     //declare auxiliary array
for(i=0; i<rowc; i++)
  b[i]=new double [6];                    //declare rows of new matrix (6 columns)
for (i=0; i<4; i++)
  if (r[i]==0)                    //if r[i] copy the i-th row to new matrix
     {for (j=0; j<6; j++)
       b[k][j]=a[i][j];
      k++;
     }
for (i=0; i<rowc; i++)
  for (j=0; j<6; j++)
     SG2->Cells[j+1][i+1]=FloatToStr(b[i][j]);
for (int i=0; i<rowc; i++)       //free memory from dynamic matrix
    delete [] b[i];
delete []b;
}
//---------------------------------------------------------------------------


void __fastcall TForm1::Button2Click(TObject *Sender)
{ int i,j;
for (i=1; i<=4; i++)
  for (j=1; j<=6; j++)
     SG1->Cells[j][i]="";
for (i=1; i<=SG2->RowCount; i++)
  for (j=1; j<=6; j++)
     SG2->Cells[j][i]="";
for (i=0; i<4; i++)
     SG3->Cells[0][i]="";
}
```

# Laboratory work 3.1
# Pointers. Dynamic arrays.

### Laboratory task

***Individual variants with the low level of complexity***

**Table L1.1**

| Var | Individual task |
|---|---|
| 1 | Enter integer numbers in Memo. Create array of odd numbers. Calculate the maximum of negative array elements. |
| 2 | Enter integer numbers in StringGrid. Create array of even numbers. Calculate the quantity of array elements, which are multiple to 4. |
| 3 | Enter float numbers in Memo. Create array of negative numbers. Calculate the sum of minimum and maximum. |
| 4 | Enter float numbers in StringGrid. Create array of numbers from interval [-5, 7). Calculate the product of array elements with odd indexes. |
| 5 | Enter array of 12 integer numbers. Create new array of the first array elements, which are multiple to 3, but not multiple to 4. Sort array in descending order. |
| 6 | Enter integer matrix 4x4. Create one-dimensional array of even matrix elements, which are less than 10. Sort array in ascending order. |
| 7 | Enter double numbers in Memo. Create array of numbers from interval (0, 7]. Sort array on decrease. |
| 8 | Enter double numbers in StringGrid. Create array of numbers with absolute value is greater than 5. Calculate the sum of positive array elements which even indexes. |
| 9 | Enter integer numbers in Memo. Create array of positive even numbers. Replace minimum and maximum. |
| 10 | Enter integer numbers in StringGrid. Create array of odd numbers from interval [-6, 8). Change all negative elements with average of array. |
| 11 | Enter array of 9 float numbers. Create new array of the first array elements, which are less than average of the first array. Calculate the sum of new array. |
| 12 | Enter float matrix 4x5. Create one-dimensional array of non-zero matrix elements, which are greater than -3. Find minimum of new array. |
| 13 | Enter integer numbers in Memo. Create array of positive numbers in odd lines. Calculate the product of elements, which are not multiple to 3. |
| 14 | Enter float numbers in StringGrid. Create array of negative numbers with even indexes. Calculate the difference between the last element and maximum. |
| 15 | Enter double numbers in Memo. Create array of numbers from interval [-5, 1)U[3, 8). Calculate the quantity of elements, which are less, than average. |
| 16 | Enter double numbers in StringGrid. Create array of positive numbers with absolute value from interval [-6, 8). Define, is array sorted on increase. |
| 17 | Enter array of 11 integer numbers. Create new array of the first array elements, which are not greater than given number (enter it from the form). Calculate the quantity of even elements. |
| 18 | Enter double matrix 6x4. Create one-dimensional array of matrix elements from the interval [-4, 4]. Calculate the quantity of elements, which are equal to minimum. |
| 19 | Enter integer numbers in Memo. Create array of numbers from interval [-2, 10) which are not multiple to 4. Calculate the average of elements, which go after maximum. |
| 20 | Enter float numbers in StringGrid. Create array of one-digit numbers with even indexes. Calculate the product of non-zero elements. |
| 21 | Enter array of 11 double numbers. Create new array of the first array elements, which go before minimum. Define, is array sorted on decrease. |
| 22 | Enter integer matrix 6x4. Create one-dimensional array of matrix elements which are multiple to one of its indexes. Calculate the sum of array elements, which are bigger than their neighbors. |

***Individual variants with the medium level of complexity***

In all variants you must *write subroutines* to calculate the quantity of array elements and fill in array from Memo (parameters are array and quantity of elements). Type of numbers is float or you must select it in correspondence to the content of the individual task.

**Table L1.2**

| Var | Individual task |
|---|---|
| 1 | Enter numbers in Memo. Create an array of numbers, which go before -123. If there is no number -123 in Memo, create array of all numbers. Write subroutine to calculate the average of array elements (array is a parameter). |
| 2 | Enter numbers in Memo. Create an array of numbers, which go after first number equal to 5. If there is no number 5 in Memo, create array of all numbers. Write subroutine to calculate the maximum of array elements (array is a parameter). |
| 3 | Enter numbers in Memo. Create an array of positive numbers. If there is no any positive number in Memo, output the message. Write subroutine to calculate the product of array elements (array is a parameter). |
| 4 | Enter numbers in Memo. Create an array of negative numbers. If there is no any negative number in Memo, output the message. Write subroutine to calculate the sum of array elements (array is a parameter). |
| 5 | Enter numbers in Memo. Create an array of numbers, which absolute value is not bigger than 8. If there are no such numbers in Memo, output the message. Write subroutine to calculate the quantity of positive array elements (array is a parameter). |
| 6 | Enter numbers in Memo. Create an array of numbers, which go before 0. If there are no such numbers in Memo, create array of all numbers. Write subroutine to calculate the maximum of array elements' absolute values (array is a parameter). |
| 7 | Enter numbers in Memo. Create an array of numbers from the interval (-2,8]. If there are no such numbers in Memo, output the message. Write subroutine to calculate the average of positive array elements (array is a parameter). |
| 8 | Enter numbers in Memo. Create an array of positive numbers, which are not bigger than 100. If there are no such numbers in Memo, output the message. Write subroutine to calculate the quantity of array elements from the interval [5, 15] (array is a parameter). |
| 9 | Enter numbers in Memo. Create an array of numbers, which difference from 5 is not bigger than 10. If there are no such numbers in Memo, output the message. Write subroutine to calculate the average of array elements (array is a parameter). |
| 10 | Enter numbers in Memo. Create an array of numbers, which go before the first number, which absolute value is less than 10. If there are no such numbers in Memo, create array of all numbers. Write subroutine to calculate the quantity of negative array elements (array is a parameter). |
| 11 | Enter numbers in Memo. Create an array of numbers, which go after the fist positive number. If there is no positive numbers in Memo, create array of all numbers. Write subroutine to calculate the minimum of non-zero array elements (array is a parameter). |
| 12 | Enter numbers in Memo. Create an array of numbers, which go before the first negative number. If there is no negative number in Memo, create array of all numbers. Write subroutine to calculate the average of non-zero array elements (array is a parameter). |
| 13 | Enter numbers in Memo. Create an array of numbers, which go before the first three-digit number. If there is no three-digit number in Memo, create array of all numbers. Write subroutine to calculate the quantity of non-negative array elements (array is a parameter). |
| 14 | Enter numbers in Memo. Create an array of one-digit numbers. If there are no one-digit numbers in Memo, output the message. Write subroutine to calculate the product of positive array elements (array is a parameter). |
| 15 | Enter numbers in Memo. Create an array of two-digit numbers. If there are no two-digit numbers in Memo, output the message. Write subroutine to calculate the sum of negative array elements (array is a parameter). |
| 16 | Enter numbers in Memo. Create an array of numbers. If there are no three-digit numbers in Memo, output the message. Write subroutine to calculate the average of negative array elements (array is a parameter). |

Continuation of the table L1.2

| Var | Individual task |
|---|---|
| 17 | Enter numbers in Memo. Create an array of numbers, which go before the first negative two-digit number. If there is no negative two-digit number in Memo, create array of all numbers. Write subroutine to calculate the product of non-zero array elements from the interval (-5, 5] (array is a parameter). |
| 18 | Enter numbers in Memo. Create an array of numbers, which go before the first positive number. If there is no positive number in Memo, create array of all numbers. Write subroutine to calculate the sum of array elements with even indexes (array is a parameter). |
| 19 | Enter numbers in Memo. Create an array of numbers, which go after the first negative number. If there is no negative number in Memo, output the message. Write subroutine to calculate the product of array elements with odd indexes (array is a parameter). |
| 20 | Enter numbers in Memo. Create an array of odd numbers. If there is no odd numbers in Memo, output the message. Write subroutine to calculate the sum of minimum and maximum (array is a parameter). |
| 21 | Enter numbers in Memo. Create an array of numbers, which are multiple to 3. If there are no such numbers in Memo, output the message. Write subroutine to calculate the difference between the product of array elements and their sum (array is a parameter). |
| 22 | Enter numbers in Memo. Create an array of numbers, which absolute value is from the interval [6, 18]. If there are no such numbers in Memo, output the message. Write subroutine to calculate the difference between the sum of array elements and maximum (array is a parameter). |

### *Individual variants with the high level of complexity*

For correspondent task in table L1.2 write the auxiliary task:

1) Output array elements in the StringGrid (parameters are: pointer to the StringGrid object (the StringGrid, which is filling), array and quantity of elements.
2) Write subroutine with task from table L1.3. Output changed array in other StringGrid. For this purpose use the function from item 1).

**Table L1.3**

| Var | Individual task |
|---|---|
| 1 | Subroutine receives two parameters (numbers), and if the first of parameters is bigger than the second one, changes the second parameter with the first one. Use this function to change in array elements, which are greater than average, with the value of average. |
| 2 | Subroutine receives three parameters (numbers), and if the first and the second of parameters are equal, changes the first parameter with the third one. Use this function to change in array elements, which are equal to minimum, with the number -11. |
| 3 | Subroutine receives three parameters (numbers), and if the first and the second of parameters are equal, changes the second parameter with the third one. Use this function to change in array elements, which are equal to maximum, with the number 100. |
| 4 | Subroutine receives two parameters (numbers), and if the first of parameters is bigger than the second one, replaces the values of parameters. Use this function to replace in array those elements, which are greater than their next elements, with next elements. |
| 5 | Subroutine receives two parameters (numbers), and if the first of parameters is less than the second one, changes the first parameter with the product of parameters. Use this function to multiply in array those elements, which are less than the first element, by the first element. |
| 6 | Subroutine receives two parameters (numbers), and if they have opposite signs, increases both parameters by 1. Use this function to increase by 1 in array those pairs of elements, which have one sign. |
| 7 | Subroutine receives three parameters (numbers), and if the second parameter is greater the first and the third parameters, changes the second parameter with the average of the first and the third. Use this function to change in array elements, which are bigger than their neighbors, with the average of their neighbors. |
| 8 | Subroutine receives two parameters (numbers), and if the first of parameters is less than the second one, the first parameter with the second one. Use this function to change in array all positive elements, which are less than the average of positive elements, with this average. |

Continuation of the table L1.3

| Var | Individual task |
|---|---|
| 9 | Subroutine receives two parameters (numbers), and changes the smaller of parameters with the bigger one. Use this function to change the smaller of two elements with the bigger one in all pairs of elements. |
| 10 | Subroutine receives three parameters (numbers), and if the first of parameters is less than other parameters, change the first parameter with the product of other parameters. Use this function to change in array all elements, which are less than their neighbors, with the product of their neighbors. |
| 11 | Subroutine receives two parameters (numbers), and if they have different signs, changes their signs on the opposite. Use this function to change to the opposite signs in all pairs of elements, which have different signs. |
| 12 | Subroutine receives two parameters (numbers), and if the second of parameters is equal to 0, changes it with the first parameter. Use this function to change in array all zero elements, with minimum of non-zero elements. |
| 13 | Subroutine receives two parameters (numbers), and if the difference between parameters is less than 2, changes the first parameter with 0. Use this function to change all elements, which difference from average of non-zero elements is less than 2, with 0. |
| 14 | Subroutine receives three parameters (numbers), and changes the smaller of the first and the second parameters with the third one. Use this function to change with the first element all array elements, which are less than 10. |
| 15 | Subroutine receives three parameters (numbers), and if the absolute value of difference of the first and the second parameters is bigger than 5, decreases both parameters with the third one. Use this function to decrease by 3 all pairs of elements elements, between which difference is bigger than 5. |
| 16 | Subroutine receives two parameters (numbers), and if the first of parameters is less than the second one, replaces their values. Use this function to replace in array all pairs of elements, if the first of them is bigger than the second. |
| 17 | Subroutine receives three parameters (numbers), and if the first of parameters is bigger than the third one, changes the third parameter with the second one. Use this function to change in array all negative elements, with the average of negative elements. |
| 18 | Subroutine receives three parameters (numbers), replaces the first and the second parameters and increases them by the third one. Use this function to replace in array all pairs of elements and increase them by product of non-zero elements from the interval (-5, 5]. |
| 19 | Subroutine receives two parameters (numbers), and if they are equal, changes the second parameter with their product. Use this function to change in array all elements, which absolute value is equal to maximum, with the product of element's absolute value and maximum. |
| 20 | Subroutine receives three parameters (numbers), and if the first parameter is bigger than the third one, changes the second parameter with average of three parameters. Use this function to change array elements, for which previous element is bigger than next element, with average value of this element and its neighbors. |
| 21 | Subroutine receives three parameters (numbers), and if the first parameter is equal to the second one, changes the first parameter with the third one. Use this function to change in each series successively going identical elements all elements, except for last , with 0. |
| 22 | Subroutine receives three parameters (numbers), and if the first parameter is less than the second one, changes the second parameter with the first one. Use this function to find minimum and then increase all elements by minimum. |

# Laboratory work 3.2
# Pointers. Dynamic matrixes

**Laboratory task.**

***Individual variants with the low level of complexity***

Enter matrix dimension – number of rows and columns. Enter matrix elements. Do calculation according the table L2.1

**Table L2.1**

| Var | Elements type | Individual task |
|---|---|---|
| 1 | integer | Calculate the quantity of odd elements. |
| 2 | float | Calculate the sum of negative elements. |
| 3 | double | Calculate the product of elements with absolute value bigger than 5. |
| 4 | integer | Calculate the average of elements, which are multiple to 3. |
| 5 | float | Calculate the geometric middle of elements from the interval [-2, 5). |
| 6 | double | Calculate the minimum of positive elements. |
| 7 | integer | Calculate the maximum of even elements. |
| 8 | float | Output the indexes of elements, which are smaller than 13. |
| 9 | double | Calculate the quantity of elements, which are bigger than sum of their indexes. |
| 10 | integer | Calculate the sum of elements, which lay under the main diagonal. |
| 11 | float | Calculate the product of elements, which lay on the main and on the secondary diagonals. |
| 12 | double | Calculate the average of elements, which lay above the secondary diagonal. |
| 13 | integer | Calculate the geometric middle of elements in even columns. |
| 14 | float | Calculate the maximum of negative elements. |
| 15 | double | Calculate the minimum of elements, which lay under the secondary diagonal. |
| 16 | integer | Calculate the indexes of rows, which contain zero elements. |
| 17 | float | Calculate the quantity of elements, which are smaller than the product of their indexes. |
| 18 | double | Calculate the sum of elements in odd rows. |
| 19 | integer | Calculate the product of elements, which are multiple to one of their indexes. |
| 20 | float | Calculate the average of elements, which are bigger than entered integer number. |
| 21 | double | Calculate the maximum of elements of both diagonals. |
| 22 | integer | Calculate the geometric middle of elements, which are multiple to both indexes. |

***Individual variants with the high level of complexity***

Calculate matrix elements under the formula in table L2.2. Create new matrix of the first matrix elements according to individual task in table L2.2.

**Table L2.2**

| Var | Matrix dimension | Formula | Individual task |
|---|---|---|---|
| 1 | 3x4 | $\sum_{k=1}^{\infty}\frac{(-1)^k(i+j)^{2k+1}}{(2k+1)!}$ -i | Rows which have positive sums. |
| 2 | 4x3 | $\sum_{k=1}^{\infty}\frac{(-1)^k(i+j)^{2k+1}}{k^2(k+1)!}$ -j | Columns which have zero elements. |
| 3 | 3x5 | $\sum_{k=1}^{\infty}\frac{(-1)^{k-1}(i+j)^{3k-1}}{(k+2)k!}$ -ij$^2$ | Rows which have only positive elements |
| 4 | 5x3 | (-1)$^i$ $\sum_{k=1}^{\infty}\frac{(-1)^k(i+j)^{2k+1}}{(2k-1)!}$ | Columns which have only negative elements. |
| 5 | 4x5 | (-1)$^j$ $\sum_{k=1}^{\infty}\frac{(-1)^{k+1}(i+j)^{k+2}}{k(2k+1)!}$ | Rows with more than 1 negative element. |

Continuation of the table L2.2

| Var | Matrix dimension | Formula | Individual task |
|---|---|---|---|
| 6 | 5x4 | $(-1)^{i/j}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}(i+j)^{k-1}}{(2k-1)\cdot(k+1)!}$ | Columns with more than 2 negative elements |
| 7 | 3x6 | $(-1)^{i+j}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}(i+j)^{2k}}{k!2^{k-1}}$ | Rows with elements with absolute value is less than 2 |
| 8 | 6x3 | $(-1)^{i-j}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}(i+j)^{2k}}{(2k-1)!}$ | Columns with more than 2 positive elements. |
| 9 | 4x6 | $(-1)^{i*j}\sum_{k=1}^{\infty}\frac{(-1)^{k}(i+j)^{3k-1}}{(k+3)(3k)!}$ | Rows with positive products |
| 10 | 6x4 | $(-1)^{i/j}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}(i+j)^{k}}{(k+4)!}$ | Columns with negative sums of elements |
| 11 | 5x6 | $(-1)^{i+j}\sum_{k=1}^{\infty}\frac{(-1)^{k}(i+j)^{2(k+1)}}{(2k)!}$ | Rows with average bigger than 5 |
| 12 | 6x5 | $(-1)^{i-j}\sum_{k=1}^{\infty}\frac{(-1)^{k-1}(i+j)^{3k-2}}{2^{k+1}\cdot k!}$ | Columns without elements from[-3,4) |
| 13 | 4x4 | $(-1)^{i*j}\sum_{k=1}^{\infty}\frac{(-1)^{k-1}(i+j)^{k}}{k!}$ | Rows with positive sums |
| 14 | 5x5 | $(-1)^{i/j}\sum_{k=1}^{\infty}\frac{(-1)^{k-1}(i+j)^{3k-1}}{(2k)!}$ | Columns with more than 3 positive elements |
| 15 | 6x6 | $(-1)^{i+j}\sum_{k=1}^{\infty}\frac{(-1)^{k-1}(i+j)^{3k+1}}{3k\cdot(k+1)!}$ | Rows with sums bigger than 7 |
| 16 | 3x7 | $(-1)^{i-j}\sum_{k=1}^{\infty}\frac{(-1)^{k}(i+j)^{3k-1}}{(k+1)!k^{2}}$ | Columns with products smaller than maximum of matrix |
| 17 | 7x3 | $(-1)^{i*j}\sum_{k=1}^{\infty}\frac{(-1)^{k}(i+j)^{2k+1}}{k\cdot(2k+1)!}$ | Rows with less than 3 positive elements |
| 18 | 4x7 | $(-1)^{i/j}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}(i+j)^{k-1}}{2k(2k+1)!}$ | Columns which sums are less than 10 |
| 19 | 7x4 | $(-1)^{i+j}\sum_{k=1}^{\infty}\frac{(-1)^{k}(i+j)^{2k+1}}{(2k+1)!}$ | Rows which products are bigger than average value of matrix |
| 20 | 5x7 | $(-1)^{j-i}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}(i+j)^{k+3}}{k^{2}(k+2)!}$ | Columns which sums are not less than 8 |
| 21 | 7x5 | $(-1)^{i/j}\sum_{k=1}^{\infty}\frac{(-1)^{k}(i+j)^{2(k+1)}}{(k+2)k!}$ | Rows which absolute values of product are bigger than 6 |
| 22 | 6x7 | $(-1)^{i*j}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}(i+j)^{2k}}{k!\cdot 2^{k}}$ | Columns which sums are less than average of matrix |

# Laboratory work 3.3
# Libraries.

## 1.1 Libraries

Let us consider function which calculates the average of float array:

```
float aver(float arr[], int N)
{
   if (N==0) {ShowMessage("No elements"); return 0;}
   int i;
   float s=0;
   for (i=0; i<N; i++)
       s+=a[i];
   return (s/N);
}
```

We can use this function to calculate the average mark of students group or average temperature or average profit. If we want to use it in different programs, we must place it in **library**. Then we can include this library in any program and use this function like it is a standard function. We do the same when use commands from math library.

Library consists of two files. One of them has extension .cpp (source file). It contains declaration of functions with their code. Other one has extension .h and contains only the type definitions and functions headers (header file).

To create library with the function aver, we must do the following actions:

1. Select File – New – Unit from the main menu. File Unit1.cpp without form will appear.
2. Enter the cod of function in it:

```
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
float aver(Array arr)          //code of function aver, type Array will be declared in Unit1.h file
{
   if (N==0) return -10000;
   int i;
   float s=0;
   for (i=0; i<N; i++)
       s+=arr[i];
   return (s/N);
}
```

3. Do right click on the bookmark with filename Unit1.cpp and select Open Source/Header file. File with extension .h will appear.
4. Write the function header in it.

```
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
const int N=10;
typedef float Array [N];       //declare new type Array (array of N float elements),
                               //now we can use it as standard type
float aver(Array arr);         //header of function aver
#endif
```

5. Now compile it.
6. And save it with names lib.cpp and lib.h.
7. The library is ready! To use it in our program, simply write

#include <lib.h>
on the beginning of it.

8. To call function aver in program (if a is an array declared above), write something like this:
   float sr=aver(a, 31);

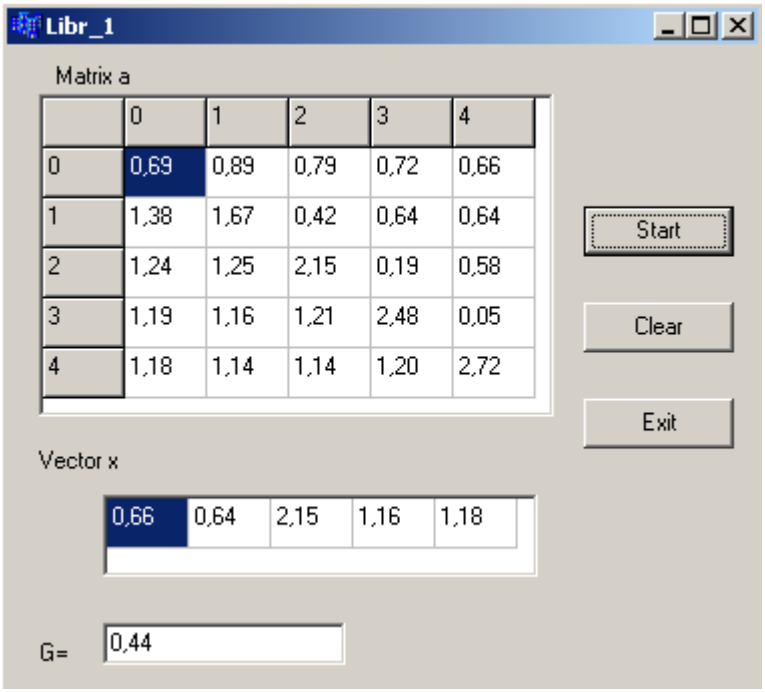
## 1.5 Examples of programs

**Example 1**

1. Create a user's library.
2. In this library write functions to calculate matrix ***a*** elements under the formula:

$$a_{i,j} = \begin{cases} arctg\left(\frac{i-j+7}{20}\right) + \lg\left(\frac{7i-j}{i^2-j^2}+3\right), & if \quad j \neq i \\ \ln(i+2.3*j+2), & othewise \end{cases}$$

3. Create one-dimensional array ***x*** of matrix ***a*** secondary diagonal elements.
4. Calculate the value of scalar variable **G** – the product of matrix ***a*** even rows elements**.**




| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0,69 | 0,89 | 0,79 | 0,72 | 0,66 |
| 1 | 1,38 | 1,67 | 0,42 | 0,64 | 0,64 |
| 2 | 1,24 | 1,25 | 2,15 | 0,19 | 0,58 |
| 3 | 1,19 | 1,16 | 1,21 | 2,48 | 0,05 |
| 4 | 1,18 | 1,14 | 1,14 | 1,20 | 2,72 |

| 0,66 | 0,64 | 2,15 | 1,16 | 1,18 |
|---|---|---|---|---|

**File Lib.h: (header file of our library)**

```
#ifndef LibH
#define LibH
//---------------------------------------------------------------------------
#endif
const int N=5, M=5;
typedef double Matr [N][M];              //declare type for matrix
typedef double Vect [N];                 //declare type for vector
void create_matrix(Matr mat);            //header of function create_matrix
void create_vector(Matr mat, Vect vec);  //header of function create_vector
double func(Matr mat);                   //header of function func
```

**File Lib.cpp: (source file of our library)**

```
#pragma hdrstop
#include "Lib.h"
#include <math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
void create_matrix(Matr mat)          //code of function create_matrix
{                                     //matrix is a parameter of function
```

```
int i,j;
for (i=0; i<N; i++)
  for (j=0; j<M; j++)
    if (i!=j)
      mat[i][j]=atan((i-j+7.)/20)+log10((7.*i-j)/(i*i-j*j)+3);
    else
      mat[i][j]=log(i+2.3*j+2);
}

void create_vector(Matr mat, Vect vec)      //code of function create_vector
{                                           //matrix and vector are parameters of function
int i;
for (i=0; i<N; i++)
  vec[i]=mat[i][N-i-1];
}

double func(Matr mat)                 //code of function func
{                                     //matrix is a  parameter of function
int i,j;
double p=1;
for (i=0; i<N; i++)
  if (i%2==0)
    for (j=0; j<M; j++)
      p*=mat[i][j];
return p;
}
```

**File Unit1.cpp: (source file of our project, which uses library Lib)**

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Lib.h"                //include library Lib to program
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
for (int i=0; i<N; i++)
  StringGrid1->Cells[0][i+1]=IntToStr(i);
for (int j=0; j<M; j++)
  StringGrid1->Cells[j+1][0]=IntToStr(j);
}
//---------------------------------------------------------------------------
Matr a;         //declare global variables – matrix a and vector x; types Matr and Vect are known
Vect x;         //from included Lib.h

void __fastcall TForm1::Button1Click(TObject *Sender)
{
int i, j;
create_matrix(a);       //call of function to create matrix a
for (i=0; i<N; i++)     //output matrix a
  for (j=0; j<M; j++)
```

```
    StringGrid1->Cells[j+1][i+1]=FormatFloat("0.00",a[i][j]);
create_vector(a,x);     //call of function to create one-dimensional array x
for (i=0; i<N; i++)     //output vector x
  StringGrid2->Cells[i][0]=FormatFloat("0.00",x[i]);
double G=func(a);     //call of function to calculate the scalar variable G
Edit1->Text=FormatFloat("0.00",G);          //output variable G
}
```

### 1.6 Laboratory task

1. Create a user's library.
2. In this library write functions to calculate matrix ***a*** elements according to the table L3.1.
3. Create one-dimensional array ***x*** according the individual task from table L3.1.
4. Calculate the value of scalar variable **G** according the individual task from table L3.1**.**

**Table L3.1**

| № | Dim | Matrix | Vector | Scalar |
|---|---|---|---|---|
| 1 | 8*3 | $a_{i,j} = \left(\sin^2 i + \cos^2 j\right)^{\frac{i-5}{j+1}} + 7{,}45 * tg\left(\frac{i-5}{j+1.5}\right)$ | Sums of matrix rows | Difference between matrix' maximum and product of its indexes |
| 2 | 4*6 | $a_{i,j} = \log_3^3(7.3i - j + 8) - \frac{(-e)^j}{(j+1)!}$ | Sums of matrix columns positive elements' squares | Product of positive matrix elements |
| 3 | 8*4 | $a_{i,j} = \left(-\frac{2+i}{3+j}\right)^i - e^{\cos j} - 3\pi$ | Averages of matrix rows | Sum of matrix elements with even sum of indexes (i+j) |
| 4 | 8*3 | $a_{i,j} = (-e)^{\frac{i+j}{35}} \lg\left(e^{\frac{i+5}{j+0.5}}\right) + \sqrt{\lvert ctg(i+j+2)\rvert}$ | Maximums of matrix rows | Matrix average |
| 5 | 8*8 | $a_{i,j} = \left(-\frac{2}{3*j+2.5}\right)^i + e^{\cos j}$ | Sums of matrix columns | Sum of secondary diagonal elements |
| 6 | 6*6 | $a_{i,j} = \left(\frac{2+e^i}{e^{3+j}}\right)^j - \cos\left(e^{\cos j}\right)$ | Scalar products of matrix rows and its secondary diagonal | Product of maximum and minimum of the matrix main diagonal |
| 7 | 6*5 | $a_{i,j} = \left(\frac{\cos^2 i}{3{,}5 - 0.5j}\right)^{\sin(j+i)} - e^{\frac{i}{j-2.5}}$ | Matrix row with maximal elements' sum | Product of matrix non-zero elements |
| 8 | 6*7 | $a_{i,j} = \log_4^4(i + 1.2 + j) - 1{,}1^i$ | Averages of matrix columns | Quantity of matrix negative elements |
| 9 | 7*7 | $a_{i,j} = \left(\sin^2 i + \cos^2 j\right)^3 - (-1.3)^{i+j} \lg\left(e^{\frac{i+5}{j+\pi}}\right)$ | Products of matrix rows | Difference between the sum of main diagonal and the product of secondary diagonal |
| 10 | 5*6 | $a_{i,j} = \left(\sin^3 i - \cos j\right)^3 + 7{,}4 * \lg\left(\left\lvert\frac{i-7.5}{j+3}\right\rvert + 1\right)$ | Column with minimal sum of elements' absolute values | Sum of matrix negative elements' squares |
| 11 | 6*6 | $a_{i,j} = \left(\frac{\ln^2(7-i)}{3{,}5 - 0.5j}\right)^2 + (-e)^{i+j}$ | Squares of matrix secondary diagonal elements | Sum of matrix' minimum and maximum |
| 12 | 5*6 | $a_{i,j} = \left(-\frac{\pi}{e}\right)^{i+j} \lg\left(e^{\frac{i+5}{j+4}}\right) + (-2)^{i+j-1} e^{2i-j}$ | Minimums of matrix columns | Average of matrix elements with odd indexes |
| 13 | 5*5 | $a_{i,j} = \frac{\log_5^2\left(i^2 + j^2 + 2\right) - e^5}{1{,}5i^2 - (-2{,}3)^i}$ | Matrix column, intersection with which is a minimum of main diagonal | Sum of matrix elements' absolute values |

Continuation of the table L3.1

| № | Dim | Matrix | Vector | Scalar |
|---|---|---|---|---|
| 14 | 6*6 | $a_{i,j}=\begin{cases} ctg(i-j+7)+\lg\left(\dfrac{7i-j}{i^2-j^2}+11.2\right), & если\ \ j\neq i \\ \log_4^4(i+2.3+j)-e, & в\ остальных\ случаях \end{cases}$ | Scalar products of matrix rows and its second column | Product of matrix elements with even row indexes |
| 15 | 5*6 | $a_{i,j}=\begin{cases} \lg(i-j)+\dfrac{7i-j}{i^2-j^2}, & если\ i>j \\ \log_7^7(j-i+7)-e^{\frac{i}{j-i+1}}, & если\ i\le j \end{cases}$ | Matrix row with minimal sum of the first and second elements | Product of matrix first three rows elements |
| 16 | 4*8 | $a_{i,j}=(-1{,}3)^{i+j}\log_2^2\left(3^{\frac{i+5}{j+1}}\right)+(-0{,}2)^{i+j+1}\left(\dfrac{\pi}{e}\right)^{2i+j}$ | Matrix row with minimal elements' sum | Sum of positive matrix elements |
| 17 | 4*4 | $a_{i,j}=\left(\sin^2 i+\cos^2 j\right)^{\frac{i-5}{j+2}}$ | Matrix column with minimal element' sum | $G=\log_4^5\left\lvert\prod_{n=0}^{3}a_{n,n}+\sum_{n=0}^{3}a_{3-n,n}^2\right\rvert$ |
| 18 | 8*9 | $a_{i,j}=\log_3^3(9i+j+1)+(-e)^i$ | Matrix column with maximal element' sum | Sum of matrix negative elements |
| 19 | 9*9 | $a_{i,j}=\left(\dfrac{7+i}{9+j}\right)^{i-j}+e^{\cos j}-3\pi$ | Scalar product of matrix rows and its main diagonal | Product of matrix elements' squares |
| 20 | 9*6 | $a_{i,j}=\lg\left(e^{\frac{i+7}{j+2}}\right)+\sqrt{\lvert tgj+1\rvert}$ | Maximums of matrix columns | Product of the main diagonal |
| 21 | 7*7 | $a_{i,j}=\left(\dfrac{3}{9*j-1}\right)^i+e^{\sin j}$ | Sums of matrix rows positive elements | $G=\log_5\left\lvert\prod_{k=0}^{6}3\sum_{j=0}^{k}\left(\lvert a_{j,6-j}-e^j\rvert\right)^4\right\rvert$ |
| 22 | 9*8 | $a_{i,j}=\left(\dfrac{\sin^3 i}{9{,}5i-1.5(j+1)}\right)^{2-\sin j}-e^{\frac{i}{j+1}}$ | Averages of matrix columns positive elements | Product of matrix elements with odd indexes |

# Laboratory work 3.4
# Characters. Arrays of characters

There are 256 characters: capital and low case Latin and Russian letters, digits, symbols of arithmetic operations, punctuation mark, special symbols (Esc, Space, Tab, CTRL etc.). They all are gathered in one table, which is called ASCII table. All characters in this table have their strong order (as letters in the alphabet). Character type in C++ is called char. Declaration:

char c;

The value of variable ***c*** is one character. It takes 1 byte in memory. Char considered being one of integer types. C++ stores characters in memory as codes (order numbers) in ASCII-table.

Examples of character constants: ‘d’, ‘x’, ‘+’, ‘8’.

Operations with characters:

1. Assignment:

c=’t’;

We can assign int numbers to char:

c=32;

32 is an order number of ‘ ‘ (space symbol). Now ***c*** is space.

2. Order number of character in ASCII table:

char c; *//c is a character*

int x=c; *//x is its code (order number) in ACSII-table*

3. Comparison (=, !=, <, <=, >, >=). One character is bigger than another, if the order number of the first character in ASCII table is bigger than the order number of the second one:

if (c1>c2) c1=c2;

There are many functions to work with characters. Some of them:

- **isalnum()** - The function isalnum() returns nonzero value if its argument is either an alphabet or integer. If the character is not an integer or alphabet then it returns zero.
- **isalpha()** - The function isalpha() returns nonzero if the character is an uppercase or lower case letter otherwise it returns zero.
- **isdigit()** - The function isdigit() returns nonzero if the character is a digit, i.e. through 0 to 9. It returns zero for non digit character.
- **islower()** - The function islower() returns nonzero for a lowercase letter otherwise it returns zero.
- **isspace()** - The function isspace() returns nonzero for space, horizontal tab, newline character, vertical tab, formfeed, carriage return; otherwise it returns zero.
- **ispunct()** - The function ispunct() returns nonzero for punctuation characters otherwise it returns zero. The punctuation character excludes alphabets, digits and space.
- **isupper()** - The function isupper() returns nonzero for an uppercase letter otherwise it returns zero.
- **tolower()**- The function tolower() changes the upper case letter to its equivalent lower case letter. The character other than upper case letter remains unchanged.
- **toupper()**- The function toupper() changes the lower case letter to its equivalent upper case letter. The character other than lower case letter remains unchanged.
- **isxdigit()** - The function isxdigit() returns nonzero for hexadecimal digit i.e. digit from 0 to 9, alphabet 'a' to 'f' or 'A' to 'F' otherwise it returns zero.

The statement

**#include<ctype.h>**

includes a header file <ctype.h> into the program. The statement

**if(isalpha(ch))**

checks whether character ch is alphanumeric. It returns nonzero value as 'a' is alphanumeric. The statement

**if(islower(ch))**

checks whether character ch is a lower case letter. It returns nonzero value as 'a' is lower case letter. The statement

**if(isupper(ch5))**

checks whether character ch5 is a upper case letter. It returns nonzero value as 'F' is a upper case letter. The statement

**if(isxdigit(ch5))**

checks whether character ch5 is a hexadecimal digit. It returns nonzero value as 'F' is a hexadecimal digit. The statement

**if(isdigit(ch3))**

checks whether character ch3 is a digit. It returns nonzero value as '0' is digit. The statement

**if(isspace(ch2))**

checks whether character ch2 is a space character. It returns nonzero value as character ch2 ' ' is space. The statement

**if(ispunct(ch4))**

checks whether character ch4 is punctuation character. It returns nonzero value as character ch4 '?' is a question mark. The statement

**cout << "The uppercase letter of 'a' is : " << static_cast<char>(toupper(ch)) << endl;**

displays the upper case equivalent of letter 'a'. The function toupper(ch) converts lower case letter to upper case letter and it returns integer value therefore type casting is used to convert integer type value to character type value. If type casting is not used then it will return the ASCII code of upper case letter.

Strings consist of characters. **String is an array of characters.** Declaration of string with 10 letters:

char c[11];

We must write one more character, than we need. The last character in string is "string-terminator" ("null-symbol"): '\0'.

There are some special symbols in C++. They all are written with two characters, which both are one symbol. First of them is \ (back slash):

'\n' – end of line (we'll use it later)

'\\' – back slash (we'll use it later to write file path)

and some other.

Edit1->Text.Length() – the length of text, written in Edit (how many characters are in Edit1)

Characters in Edit1 are numerated from 1.

To read in char array characters from Edit:

```
int N=Edit1->Text.Length();
for (int i=0; i<N; i++)
   c[i]=Edit1->Text[i+1];
```

To output character array in Edit1:

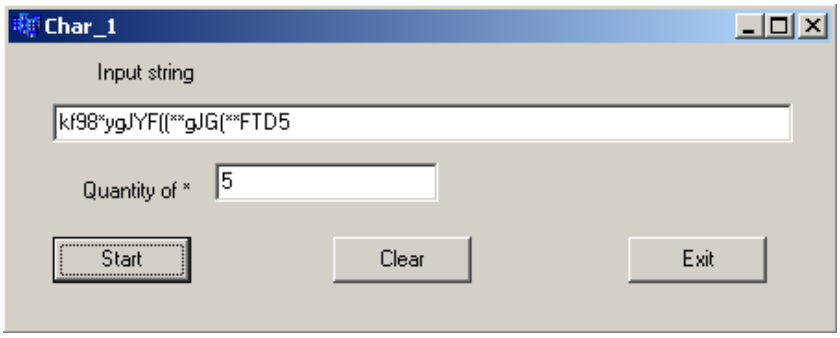
```
Edit1->Clear();
for (int i=0; i<N; i++)
   Edit1->Text= Edit1->Text+c[i];
```

Some conditions, which are often used in tasks with characters and strings:

- ✓ Is c a Latin letter?
- ✓ Is c a digit?
- ✓ Is c a capital letter?
- ✓ Is c a low case letter?
- ✓ Is c a letter 'y'

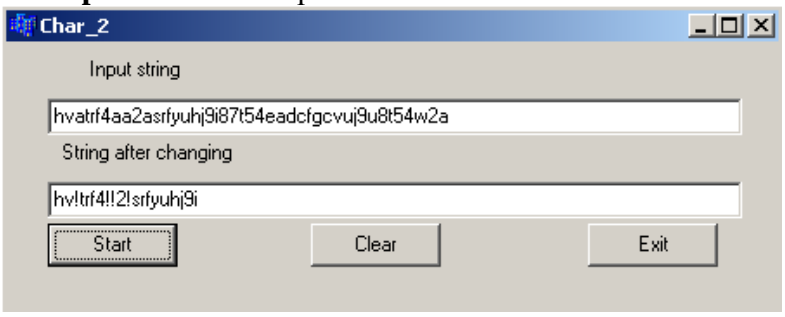
## Examples of programs

**Example 1.** Enter a sequence of 20 or less characters in Edit. Calculate the quantity of symbol "*".



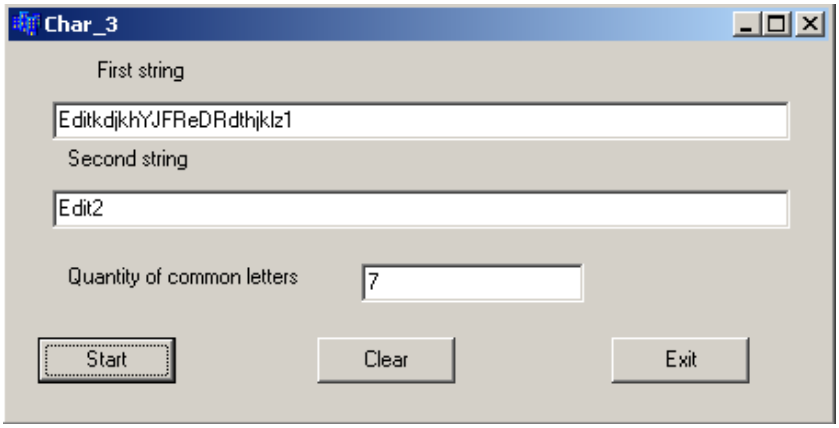

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
char c[20];
int kol=0;
int N=Edit1->Text.Length();
if (N>20) N=20;
for (int i=0; i<N; i++)
  {c[i]=Edit1->Text[i+1];
   if (c[i]=='*') kol++;
  }
Edit2->Text=IntToStr(kol);
}
```

**Example 2.** Enter a sequence of 20 or less characters in Edit. Change all “a” with “!”.




```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
char c[20];
int kol=0;
int N=Edit1->Text.Length();
if (N>20) N=20;
for (int i=0; i<N; i++)
  {c[i]=Edit1->Text[i+1];
   if (c[i]=='a') c[i]='!';
  }
for (int i=0; i<N; i++)
  Edit2->Text=Edit2->Text+c[i];
}
```

**Example 3.** Enter two sequences of 20 or less characters in Edit1 and Edit2. Calculate the quantity of characters in first sequence, which are in the second one.




The common letters in the picture are: E,d,i,d,t,d,t

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
char c1[20], c2[20];
int kol=0;
int N1=Edit1->Text.Length();
if (N1>20) N1=20;
int N2=Edit2->Text.Length();
if (N2>20) N2=20;
for (int i=0; i<N1; i++)
  c1[i]=Edit1->Text[i+1];
for (int i=0; i<N2; i++)
  c2[i]=Edit2->Text[i+1];

for (int i=0; i<N1; i++)
  for (int j=0; j<N2; j++)
    if (c1[i]==c2[j]) {kol++;break;}
Edit3->Text=IntToStr(kol);
}
```

**Example 4.** Enter char array of 20 or less elements. Output TRUE, if array contains capital letter, or FALSE otherwise.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
char c[20];
int kol=0;
int N=Edit1->Text.Length();
if (N>20) N=20;
for (int i=0; i<N; i++)
  {c[i]=Edit1->Text[i+1];
   if (isupper(c[i])) {ShowMessage("TRUE"); return;}
  }
ShowMessage("FALSE");
}
```

**Example 5.** Enter char array of 20 or less elements. Change in array first letters of all words with same capital letters.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
char c[20];
int kol=0;
int N=Edit1->Text.Length();
if (N>20) N=20;
for (int i=0; i<N; i++)
```

```
  c[i]=Edit1->Text[i+1];
for (int i=0; i<N-1; i++)
  if (c[i]==' '&& c[i+1]!=' ') toupper(c[i+1]);
  }
for (int i=0; i<N; i++)
  Edit2->Text=Edit2->Text+c[i];
}
```

**Example 6.**

Enter character matrix 4x5 and create sequence, which corresponds to the last matrix column.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
char c[4][5];
char s[4];
int i,j;
for (i=0; i<4; i++)
  for (j=0; j<5; j++)
     c[i][j]=SG1->Cells[j][i][1];
for (i=0; i<4; i++)
  {s[i]=c[i][4];
   Edit1->Text=Edit1->Text+s[i];}
}
```

**Example 7.**

Enter array of 18 or less characters. Define, is there at least one arithmetical symbol.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
char c[18];
int i;
int N=Edit1->Text.Length();
if (N>18) N=18;
for (i=0; i<N; i++)
   c[i]=Edit1->Text[i+1];
for (i=0; i<N; i++)
   if (c[i]=='+' || c[i]=='-' || c[i]=='*' || c[i]=='/')
      {ShowMessage("Yes"); return;}
ShowMessage("No");
}
```

**Example 8.**

Enter array of 18 or less characters. Create new array of symbols, which are capital Russian letters.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
char c[18], s[18];
int i, j=0;
int N=Edit1->Text.Length();
if (N>18) N=18;
for (i=0; i<N; i++)
   c[i]=Edit1->Text[i+1];
for (i=0; i<N; i++)
   if (c[i]>='A' && c[i]<='Я')
      {s[j]=c[i]; j++;}
for (i=0; i<j; i++)
   Edit2->Text=Edit2->Text+s[i];
}
```

The same example with dynamic arrays.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
int N=Edit1->Text.Length();
if (N>18) N=18;
char *c = new char[N],
int i, j=0, k=0;
for (i=0; i<N; i++)
  c[i]=Edit1->Text[i+1];
for (i=0; i<N; i++)
  if (c[i]>='A' && c[i]<='Я')
        k++;
char *s = new char[k],
for (i=0; i<N; i++)
  if (c[i]>='A' && c[i]<='Я')
     {s[j]=c[i]; j++;}
for (i=0; i<k; i++)
  Edit2->Text=Edit2->Text+s[i];
}
```

### *Individual tasks with low level of complexity*

**1** Enter char array of 15 or less elements, find an element with the biggest ASCII-code and replace it with the first element.

**2** Enter char array of 20 or less elements and find an element with the smallest ASCII-code.

**3** Enter char array of 10 or less elements and replace the first and the last elements.

**4** Enter char array of 15 or less elements and find index of the first point in array.

**5** Enter char array of 13 or less elements and calculate the quantity of digits in array.

**6** Enter char array of 20 or less elements and create a new array, which only consists of the capital Latin letters from the first array.

**7** Enter char array of 20 or less elements and change all letters with "+".

**8** Enter two character arrays of 10 or less elements and output “Equal”, if arrays are identical, and “Different” otherwise.

**9** Enter two character arrays of 7 or less elements and output “Intersect”, if there is at least one common element in both arrays.

**10** Enter char sequence of 4 or less elements and create matrix 4x4, which diagonal is entered sequence, and other elements are ‘.’.

**11** Enter character matrix 3x4 and define, is there ‘N’ or ‘n’ in it.

**12** Enter character matrix 3x3 and define, is there at least one digit in it.

**13** Enter array of 10 natural numbers and create array of characters, which correspond to their codes in ASCII table.

**14** Enter two character arrays of 13 or less elements and calculate the quantity of low case Latin letters in it.

**15** Enter char array of 15 or less elements and find index of the last coma.

**16** Enter char array of 20 or less elements and find the symbol with the smallest ASCII-code in it.

**17** Enter char array of 20 or less elements and create new array, which only contains digits from the first array.

**18** Enter char array of 20 or less elements and change all spaces with points in it.

**19** Enter character matrix 3x4 and create sequence, which corresponds to the second matrix row.

**20** Enter character matrix 4x4 and create character array, which corresponds to the matrix diagonal.

**21** Enter character matrix 3x4 and create sequence, which corresponds to the first matrix column.

**22** Enter a sequence of 10 or less characters and define, is there ‘Φ’ or ‘ϕ’in this sequence.

### *Individual tasks with medium level of complexity*

Write the same program with dynamic array (arrays).

# Laboratory work 3.5
# AnsiString

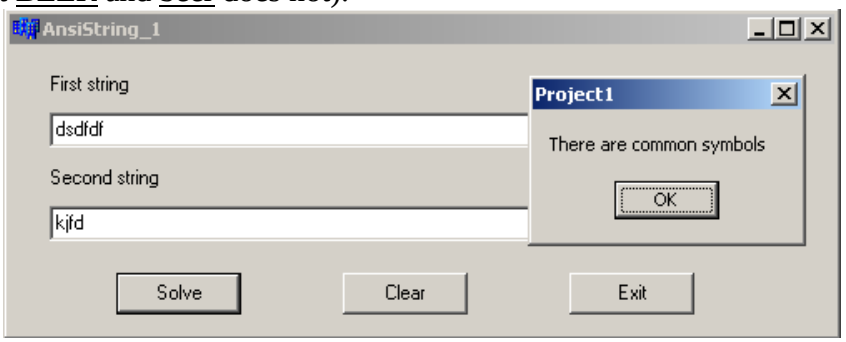
Purpose: to get practical skills for work with strings of AnsiString class in C++ Builder.

**Examples of programs**

**Example 1.** Compose a logical scheme and a C++ program that:

<input>) for 2 given strings (each includes 5 or less items)

<output>) checks whether or not they include any common symbols (e.g.: Bear and Beer have common symbols, but BEER and beer does not).



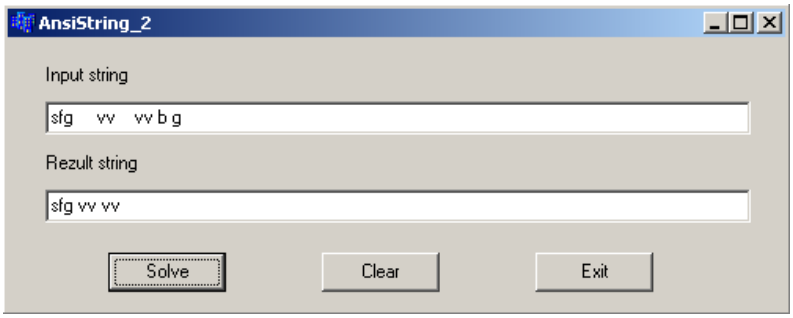

**Text of the program:**

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
AnsiString S1, S2;
S1=Edit1->Text;
S2=Edit2->Text;
int N1=S1.Length();
int N2=S2.Length();
if (N1>5)
  {S1.SetLength(5); N1=5;}
if (N1>5)
  {S2.SetLength(5); N2=5;}
 ShowMessage(S1);
int i,j;
for (i=1; i<=N1; i++)
   for (j=1; j<=N2; j++)
      if (S1[i]==S2[j])
         {ShowMessage("There are common symbols");
          return;
         }
ShowMessage("There are no common symbols");
}
```

**Example 2.** Compose a logical scheme and a C++ program that:

<input>) for a given string (up to 16 characters)

<output>) erases all double blanks in the string (the result allows to include only single occurrences of blanks).



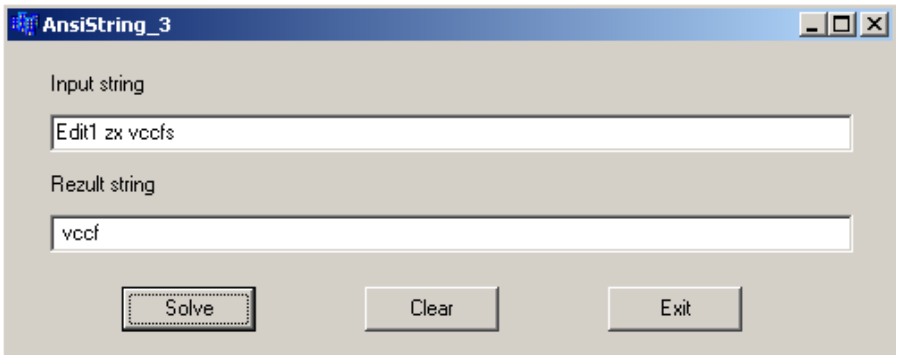

**Text of the program**

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
AnsiString S;
S=Edit1->Text;
int N=S.Length();
if (N>16)
  {S.SetLength(16); N=16;}
int i;
for (i=1; i<N; i++)
   if (S[i]==' ' && S[i+1]==' ')
         {S.Delete(i,1); i--; N--;}
Edit2->Text=S;
}
```

**Example 3.** Compose a logical scheme and a C++ program that:

<input>) for a string of 15 or less symbols

<output>) finds the last word in input string (taken blank as the only possible delimiter of words).




**Text of the program**

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
AnsiString S,S1,S2;
S=Edit1->Text;
int N=S.Length();
if (N>15)
  {S.SetLength(15);  N=15;}
int i; int x;
x=0;
for (i=N; i>=1; i--)
   if (S[i]==' ')
         {x=i; break;}
S1=S.SubString(x,N-x);
Edit2->Text=S1;
}
```

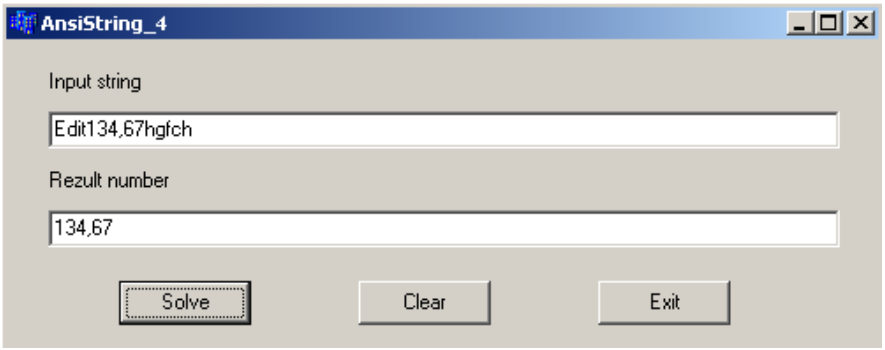
**Example 4.** Compose a logical scheme and a C++ program that:

<input>) for a string of 15 or less symbols

<output>) finds the first word that represents some number in input string (taken blank as the only possible delimiter of words). E.g.: for the string

"I am 17 years old and have 2 younger sisters"

the result will be 17.




**Text of program**

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
AnsiString S,S1,S2;
S=Edit1->Text;
int N=S.Length();
if (N>15)
  {S.SetLength(15);  N=15;}
int i,k; double x;
x=0;
for (i=1; i<=N; i++)
   if (S[i]>='0'&& S[i]<='9') {k=i;break;}
while(S[i]>='0'&& S[i]<='9'||S[i]==',') i++;
S1=S.SubString(k,i-k);
x=StrToFloat(S1);
Edit2->Text=FloatToStr(x);
}
```

***Individual tasks with normal level of complexity***

**Variant 1**.

**Task 1.** Compose a logical scheme and a C++ program that:

<input>) for a given string that includes 8 or less items

<output>) calculates the number of digits in it.

**Task 2.** Compose a logical scheme and a C++ program that:

<input>) for a given 2 strings (both up to 10 items length)

<output>) builds the third string which contains:

a) the shorter of the given two strings on the beginning,

b) the blank after it,

c) the longest of the two input strings as the end.

**Task 3.** Compose a logical scheme and a C++ program that erases all Latin letters from a string containing up to 15 symbols.

**Variant 2**.

**Task 1**. Compose a logical scheme and a C++ program that:

<input>) for a given string that includes 10 or less items

<output>) swaps the first and the last elements of input string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string (up to 20 items length)
<output>) in the case when input string is factually shorter than 20, appends it by multiple symbols & to get the length of 20.

**Task 3**. Compose a C++ program that:
1) inputs a real number R;
2) calculates the length of a circle having the radius R;
3) builds and types the string containing the text:
"The length of a circle with radius … equals to …",
where corresponding numbers rounded to thousandth must replace dots.

**Variant 3**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string that includes 10 or less items
<output>) substitutes each comma and each dot in the string for a blank.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string (up to 15 items length)
<output>) erases all digits from the input string.

**Task 3**. Compose a logical scheme and a C++ program that:
<output>) finds the index of the last dot
<input>) in a string of 10 or less symbols.

**Variant 4**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for 2 given strings (each includes 5 or less items)
<output>) displays the text EQUAL in the case when input strings are symbolically equal and displays NotEQUAL otherwise.

**Task 2.** Compose a logical scheme and a C++ program that:
<input>) for a given 7 integer numbers
<output>) builds string that contains all positive input numbers each preceded by the symbol $ and delimited by comma (e.g.: $13, $5 ).

**Task 3.** Compose a logical scheme and a C++ program that:
<input>) for a string of 8 or less symbols
<output>) builds the new string – the reversed sequence of input symbols.

**Variant 5**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string that includes 8 or less items
<output>) finds the number of Latin letters in the string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string (up to 10 characters)

<output>) builds a new string by inserting blanks between every two neighbor symbols in input string.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a string of 15 or less symbols
<output>) finds the first word in input string (taken blank as the only possible delimiter of words).

**Variant 6**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 10 or less items
<output>) finds the index of the first dot in the string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given array of 6 strings (each of 5 or less characters)
<output>) finds the alphabetically minimal string.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a string of 10 or less symbols
<output>) inserts a blank before and after any sign + occur in input string (e.g: a+b must be converted to a + b ).

**Variant 7**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for 2 given strings (each includes 6 or less items)
<output>) displays the text SAME in the case when input strings are symbolically equal and displays DIFFERENT otherwise.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for 2 given string (each up to 10 characters)
<output>) checks whether one of them is a substring of the other, and if so, delete the substring from wider string.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a string of 16 or less symbols
<output>) finds and types all numbers being words of this string (taken blank as the only possible delimiter of words).

**Variant 8**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 7 or less items
<output>) finds the index of the last blank in the string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given 2 strings (each of 8 or less characters)
<output>) compare lengths of input strings and builds the 3rd string having:
a) the shorter string at the beginning

b) blank as the next symbol
c) the longest string as the tail.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a string of 20 or less symbols
<output>) finds the first "personal name" in the string (it is assumed that personal name is any word begins with capital letter; and that blanks are only possible delimiters of words in a string).

**Variant 9**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 10 or less items
<output>) swaps the first and the last inputted elements in the string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string (up to 10 characters)
<output>) erases all beginning and end blanks in the string.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a real array of 5 numbers
<output>) builds a stringcontaining these numbers rounded to hundredth and the text Om after each of then (e.g.: 5.43 Om, 0.17 Om, … ).

**Variant 10**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 7 or less items
<output>) builds the array of decimal ANSI-numbers of these symbols.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string(up to 15 characters)
<output>) substitutes all input commas and dots for blanks.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a string of 6 or less symbols
<output>) builds new string that defers from input in reverse order of items.

**Variant 11**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 9 or less items
<output>) substitutes all commas and dots in this string for blanks.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for 2 given string (each of 10 or less characters)
<output>) builds a new string that has:
a) alphabetically 1st input string as the beginning;
b) the blank after it;

c) alphabetically 2nd input string as the end.

**Task 3**. Compose a logical scheme and a C++ program that:
<output>) erases all multiple (repeated) blanks
<input>) from a given string (up to 15 characters)

**Variant 12**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 7 or less items
<output>) builds the array that includes indexes of all occurrences of symbol # in the input string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string (up to 30 characters)
<output>) in the case when input string is factually shorter than 30, appends it by multiple symbols $ to get the length of 30.

**Task 3**. Compose a C++ program that:
1) inputs a string (it is expected to be symbolic representation of real number );
2) checks whether inputted string treating as number contains any mistakes;
3) in the case of mistakes proposes to repeat the input.

**Variant 13**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 8 or less items
<output>) finds the number of all Cyrillic letters in the string.

**Task 2**. Compose a logical scheme and a C++ program that:
<output>) erases all blanks
<input>) from a given string of up to 20 characters.

**Task 3**. Compose a logical scheme and a C++ program that:
<output>) finds the 1st word
<input>) in a given string of 15 or less items (the blank is supposed to be the only possible delimiter of words in the string).

**Variant 14**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 10 or less items
<output>) finds the index of the 1st occurrence of dot in the input string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given integer array of 5 items
<output>) builds the string that contains only positive input numbers each preceded by the symbol $ (e.g.: $100, $13, …).

**Task 3**. Compose a C++ program that:

1) inputs a string (it is expected to be symbolic representation of integer number );
2) checks whether inputted strings treating as number contains any mistakes;
3) in the case of mistakes proposes to repeat the input.

**Variant 15**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for 2 given strings (each includes 5 or less items)
<output>) displays the text EQUAL in the case when input strings are symbolically equal and displays NotEQUAL otherwise.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string (up to 10 characters)
<output>) builds a new string by inserting blanks between every two neighbor symbols in input string.

**Task 3**. Compose a logical scheme and a C++ program that:
<output>) erases all multiple (repeated) blanks
<input>) from a given string (up to 20 characters).

**Variant 16**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 7 or less items
<output>) finds the index of the 1st occurrence of a blank in the input string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given array of 6 symbolic strings (each of 7 or less items)
<output>) finds the alphabetically maximal string.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a string of 15 or less symbols
<output>) finds the last word in input string (taken blank as the only possible delimiter of words).

**Variant 17**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string that includes 10 or less items
<output>) swaps the first and the last elements of input string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for 2 given string (each up to 10 characters)
<output>) checks whether one of them is a substring of the other, and if so, delete the substring from wider string.

**Task 3**. Compose a C++ program that:
1) inputs a string (it is expected to be symbolic representation of integer number );
2) checks whether inputted strings treating as number contains any mistakes;

3) in the case of mistakes proposes to repeat the input.

**Variant 18**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string that includes 8 or less items
<output>) calculates the number of digits in it.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given 2 strings (both up to 8 items length)
<output>) builds the third string which contains:
a) the shorter of the given two strings on the beginning,
b) the blank after it,
c) the longest of the two input strings as the end.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a string of 6 or less symbols
<output>) builds the new string – the reversed sequence of input symbols.

**Variant 19**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string that includes 9 or less items
<output>) substitutes each comma and each dot in the string for a blank.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string (up to 10 characters)
<output>) erases all beginning and ending blanks in the string.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a real array of 5 numbers
<output>) builds a string containing these numbers rounded to hundredth and the text Грн. after each of then (e.g.: 515.13 Грн., 1000.74 Грн., … ).

**Variant 20**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for 2 given arrays of symbols (each includes 5 or less items)
<output>) displays the text SAME in the case when input arrays are symbolically equal and displays DIFFERENT otherwise.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string(up to 20 characters)
<output>) substitutes all input commas and dots for blanks.

**Task 3**. Compose a C++ program that:
1) inputs a real number R;
2) calculates the volume of a sphere having the radius R;
3) builds and types the string containing the text:
"The volume of a sphere with radius … equals to …",

where corresponding numbers rounded to thousandth must replace dots.

**Variant 21**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string that includes 10 or less items
<output>) finds the number of Cyrillic letters in the string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given 2 symbolic strings (each of 8 or less characters)
<output>) builds the 3rd string having:
a) the alphabetically smaller string at the beginning
b) blank as the next symbol
c) the alphabetically greater string as the tail.

**Task 3**. Compose a logical scheme and a C++ program that:
<input>) for a string of 20 or less symbols
<output>) finds the first word in input string (taken blank as the only possible delimiter of words).

**Variant 22**.

**Task 1**. Compose a logical scheme and a C++ program that:
<input>) for a given string of 10 or less items
<output>) finds the index of the first dot in the string.

**Task 2**. Compose a logical scheme and a C++ program that:
<input>) for a given string (up to 25 items length)
<output>) in the case when input string is factually shorter than 20, appends it by multiple symbols @ to get the length of 20.

**Task 3**. Compose a C++ program that:
1) inputs a string (it is expected to be symbolic representation of a number );
2) checks whether inputted strings treating as number contains any mistakes;
3) in the case of mistakes proposes to repeat the input.

# Laboratory work 3.6
# C-strings (char *)

```
AnsiString S;
char s[20];

        strcpy(s, S.c_str());
```

**c_str()** converts AnsiString in c-string.

Convertion c-string -> AnsiString:
S=**AnsiString(s)**;

char ***strcat**(char **s1*, const char **s2)*; *//add s2 to the end of s2*

char ***strchr**(const char ***s**, int **c**); *//search for the first occurrence of character **c** in the string **s***

char ***strrchr**(const char ***s**, int **c**); *//search for the last occurrence of character **c** in the string **s***

int **strcmp**(const char **s1*, const char **s2)*; *//compare strings s1 and s2*

char ***strcpy**(char **s1*, const char **s2)*; *//copy s2 in s1*

char ***strncpy**(char **s1*, const char **s2*, size_t **n**); *//copy n letters of **s**2 in **s1***

*// this command must be in pare with:*

```
    s1[n]='\0';
```

size_t **strlen**(const char ***s**); *//calculate the number of characters in **s***

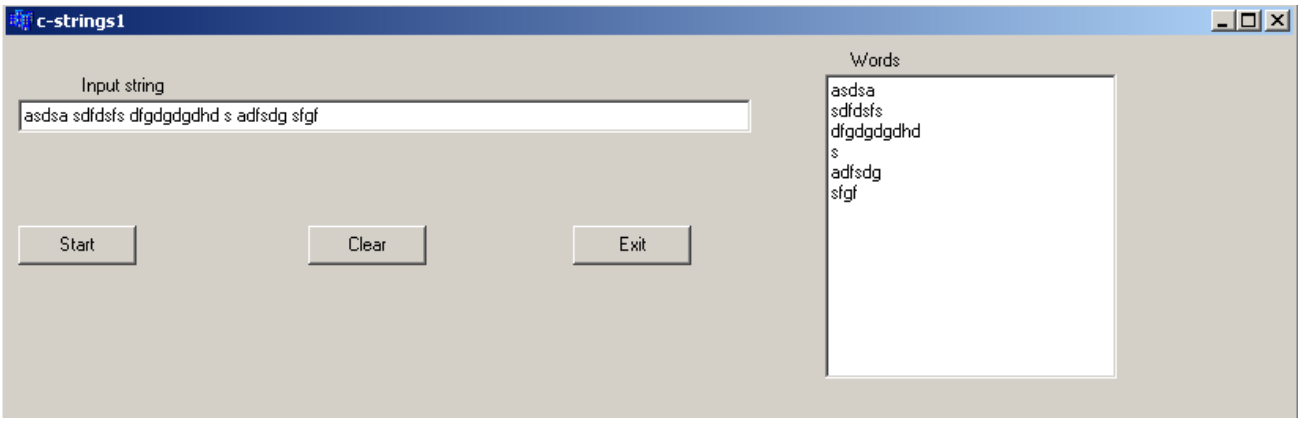
char ***strs**tr(const char **s1*, const char **s2)*; *//search for the first occurrence of word **s2** in the string **s1***

char ***strtok**(char **s1*, const char **s2)*;

*//read the word from the string (s1 is s at first, then NULL, s2 is a delimiter)*

**Example 1.**

Input string. Output in Memo all words of this string separately.




**Text of program**

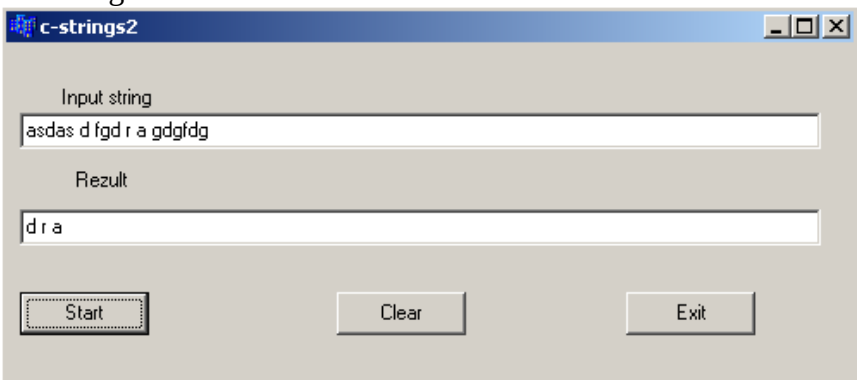
```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
int N=Edit1->Text.Length();
char *s=new char [N];
```

***AnsiString S=Edit1->Text;  //read text from Edit1 in S***

```
strcpy(s, S.c_str());          //convert S in s
char *D=" ";                   //" " is delimiter
char *p=strtok(s,D);           //first word
while (p!=0){                  //while there are words in the string
    Memo1->Lines->Add(AnsiString(p));
    p=strtok(NULL,D);          //read the next word
}
delete []s;
}
```

**Example 2.**

Input string. Create new string of one-letter words.




**Text of program**

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
int N=Edit1->Text.Length();
char *s=new char [N];
char *s1=new char [N];
AnsiString S=Edit1->Text;
strcpy(s, S.c_str());
strcpy(s1, "\0");
char *D=" ";
char *p=new char [2];
p=strtok(s,D);              //read the first word
while (p!=0){
  if (strlen(p)==1)         //check the condition, if it is true
     {strcat(s1,p);         //add word to s1
     strcat(s1," "); }      //add ' ' to s1
   p=strtok(NULL,D);        //read the next word
    }
ShowMessage("hvjh"+AnsiString(s1));
Edit2->Text=AnsiString(s1);
delete []s;
delete []s1;
delete []p;
}
```

"Delete" means: to create a new string without given words.

## Variants of the individual tasks

### Variant 1

1. Input a string. Calculate a quantity of words, which begin from capital letter.
2. Input a string. Replace all digits with their names ("one" instead of '1', "two" instead of 2, etc).

### Variant 2

1. Input a string. Output the words, which do not contain letter 'w'.
2. Input a string. Insert your name after each '!'.

**Variant 3**

1. Input a string. Create a new string of words, which start from ‘F’.
2. Input a string. Delete all words, which contain digits.

**Variant 4**

1. Input a string. Calculate a quantity of words, which begin from low-case letter.
2. Input a string. Replace all symbols of arithmetic operations with their names (“plus” instead of ‘+’, etc).

**Variant 5**

1. Input a string. Output the of words, which contain letter ’d’.
2. Input a string and a word. Insert this word after all words with odd length.

**Variant 6**

1. Input a string. Create a new string of words, which end with ‘y’.
2. Input a string. Delete all words which contain capital letters.

**Variant 7**

1. Input a string. Calculate a quantity of words, which contain low-case letters only.
2. Input two strings. Delete from the first string all letters, which present in the second string.

**Variant 8**

1. Input a string. Output the words, which start and end finish with the same.
2. Input a string. Insert after each word its order number in string.

**Variant 9**

1. Input a string. Create a new string of words, which contain letter ‘k’.
2. Input a string. Delete all words with length less than 4 symbols.

**Variant 10**

1. Input a string. Calculate a quantity of words, which contain capital letters only.
2. Input two strings. Create third string, alternating words of the first and the second strings.

**Variant 11**

1. Input a string. Output the words, which begin from capital letter.
2. Input a string. Create two strings. First of them contain consists of words with length less than 5. Second one contains other words.

**Variant 12**

1. Input a string. Create a new string of words, which do not contain letter ‘o’.
2. Input a string. Delete all words with digits.

**Variant 13**

1. Input a string. Calculate a quantity of words, which start and end finish with the same.
2. Input a string. Create a new string of words with capital letters.

**Variant 14**

1. Input a string. Output the words, which begin from low-case letter.
2. Input a string. Delete all words, which contain ‘x’.

**Variant 15**

1. Input a string. Create a new string of words, which start from ‘s’.
2. Input a string and a word. Insert this word before the first common letter (of the string and the word).

**Variant 16**

1. Input a string. Calculate a quantity of words with even length.
2. Input a string. Create a new string of the shorter and the longer words of the string and 3 spaces between them.

**Variant 17**

1. Input a string. Output the words, which contain low-case letters only.
2. Input a string. Delete all words with the first letter ‘G’.

**Variant 18**

1. Input a string. Create a new string of words, which end with ‘e’.
2. Input a string and a word. Insert this word after spaces with odd index in string.

**Variant 19**

1. Input a string. Calculate a quantity of words, which do not contain letter ‘w’.
2. Input a string. Delete all words with length bigger than 6.

**Variant 20**

1. Input a string. Output the words, which contain low-case letters only.

2. Input a string. Insert order numbers of words before each word.

**Variant 21**

1. Input a string. Create a new string of words, which contain letter ‘a’.
2. Input a string and a word. Insert this word after each digit.

**Variant 22**

1. Input a string. Output the words in reverse order.
2. Input a string. Delete all words with at least one non-letter.

**Variant 23**

1. Input a string. Output the second word of this string.
2. Input a string. Insert symbol ‘*’ before each word.

# Laboratory work 3.7
# Structures