Assigning element’s values definition:

int array[ ] = {1, 2, 3};
array[0] = 1
array[1] = 2
array[2] = 3

int array [4] = {1, 2};
array[0] = 1
array[1] = 2
array[2] = 0
array[3] = 0

```
int array[3][3] = {1, 2, 3, 4, 5};
array[0][0] = 1
array[0][1] = 2
array[0][2] = 3
array[1][0] = 4
array[1][1] = 5
array[1][2] = 0 // even though nowhere stated
array[2][0] = 0
array[2][1] = 0
array[2][2] = 0


int x[3][2][4]          3D array of integer numbers
float x[3][2][4][1]     4D array of real numbers

static char daytab[2][13] = {
{0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
{0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31} };


/* proaiterikh 2A */
/*grapste to parakatw me switch*/

/* month_name: return name of n−th month */
char *month_name(int n)
{
        char *name[] = {
        "Illegal month",
```

```
        "January", "February", "March",
        "April", "May", "June",
        "July", "August", "September",
        "October", "November", "December"
        };

        if (n < 1 || n > 12)
                return name[0]
        return name[n];
}
```

```
/* This program is an example of an array of integers */

#include <stdio.h>

void main()
{
        int values[12];
        int index;

        for (index = 0;index < 12;index++)
                values[index] = 2 * (index + 4);

        for (index = 0;index < 12;index++)
                printf("The value at index = %2d is %3d\n",index, values [index]);
}
```


```
/* This program is an example of an array of integers */

#include <stdio.h>
```

```
void main()
{
    int values[12][25]
    int i,j;

   for (i = 0;i< 12;i++)
          for (j = 0;j< 25;j++)
                    values[i][j] = i+j;

   for (i = 0;i< 12;i++)
          for (j = 0;j< 25;j++)
               printf("The value at index = [%2d]%[2d] is %3d\n",i, j, values[i][j]);

}
```

Example: Write your own program that asks user to input sequence of numbers, afterwards it calculates arithmetic middle of the given sequence. Program also prints numbers smaller than arithmetic middle, and afterwards prints numbers bigger than arithmetic middle.

```
#include <stdio.h>
#define DIMENSION 10

int main(void) {
int i;
  float sum = 0., arit_midd = 0., sequence[DIMENSION]={0};

  for (i = 0; i < DIMENSION; i++) {
    printf("Input number: ");
    scanf("%f",&sequence[i]);
    sum += sequence[i];
  }

  arit_midd = sum / DIMENSION;
  printf("Arithmetic middle of the sequence is %6.2f.\n", arit_midd);
  for (i = 0; i < DIMENSION; i++) {
    if (sequence[i] < arit_midd) {
      printf("%6.2f is smaller than arithmetic middle.\n", sequence[i]);
    }
  }

  for (i = 0; i < DIMENSION; i++) {
    if (sequence[i] > arit_midd) {
      printf("%6.2f is bigger than arithmetic middle.\n", sequence[i]);
    }
  }
}
```

Example: Write your own program that asks for input of sequence of numbers. After the program reads given numbers, it divides every number with the biggest sequence element and shows them in a way relative to the biggest element.

```
#include <stdio.h>
#define DIMENSION 10

int main(void) {
  int i;
  float max, array[DIMENSION];

  for (i = 0; i < DIMENSION; i++) {
    printf("array[%d] = ", i);
    scanf("%f",& array[i]);
    if (i == 0) {
      max = array[i];
    }
    if (max < array[i]) {
      max = array[i];
    }
  }

  printf("Biggest element in array is %f.\n\n", max);
  for (i = 0; i < DIMENSION; i++) {
    array[i] /= max;
    printf("array[%d] = %f\n", i, array[i]);
  }
}
```

Example: Compose your own program that reads given natural numbers that belong in [10, 99] interval and counts how many times each number showed up. Program stops reading numbers when element that doesn't belong to interval is given. Afterwards, program prints each number from the interval that has showed at least once, and number of times it has really been given.

```
#include <stdio.h>
#define LL 10          /* lower limit of the interval */
#define UL 99          /* upper limit of the interval */

int main(void) {
  int number, i;
  int counter[UL – LL + 1] = { 0 };
  do {
    printf("\nInput number from interval [%d, %d]: ", LL, UL);
    scanf("%d", &number);
    if (number >= LL && number <= UL) {
      counter[number - LL]++;
    }
  } while (number >= LL && number <= UL);

  for (i = DG; i <= UL; i++) {
    if (counter[i - LL] > 0) {
      printf("\nNumber %d showed up %d times", i, counter [i - LL]);
    }
  }
}
```

Example: Write your own C program that reads through real matrix, 10x10 dimensioned and finds the smallest element in main diagonal and smallest element in secondary diagonal.

```
#include <stdio.h>
#define NR_ROW      10
#define NR_COL      10

int main(void) {

int     i, j;
float mat[NR_ROW][NR_COL], min_maindg, min_secdg;

printf("Input matrix elements :");
for (i = 0; i < NR_ROW; i++) {
  for (j = 0; j < NR_COL; j++) {
      printf("\nInput element [%d][%d] : ", i, j);
      scanf("%f", &mat[i][j]);
  }
}

min_maindg = mat[0][0];
//min el. is mat(0,0), this is why loop starts from 1
    for (i = 1; i < NR_ROW; i++) {
           if (mat[i][i] < min_maindg) {
                min_maindg = mat[i][i];
          }
    }
```

```
min_secdg = mat[0][NR_COL -1];
for (i = 1; i < NR_ROW; i++) {
 if (mat[i][NR_COL-i-1] < min_secdg) {
     min_secdg = mat[i][NR_COL-i-1];
 }
}

printf("\nSmallest el. in main diagonal is: %f",
       min_maindg);
printf("\nSmallest el. in second diagonal is: %f",
       min_secdg);
}
```

Example: Write your own C program that transposes matrix. Program stores given matrix dimensions and every single matrix element must be given. Transposed matrix is the one with rows and columns switched.

```
#include <stdio.h>
#define MAX_ROW      50
#define MAX_COL      50

int main(void) {
int i, j, m, n, temp;
int mat[MAX_ROW][MAX_COL];

    int dim = (MAX_ROW < MAX_COL)? MAX_ROW : MAX_COL;
    do {
        printf("Input number of rows < %d   :", dim);
        scanf("%d", &m);
        printf("Input number of columns < %d:", dim);
        scanf("%d", &n);
    }   while (m < 1 || m > dim || n < 1 || n > dim);

   printf("\nInput of matrix elements :\n");
   for (i = 0; i < m; i++) {
      for (j = 0; j < n; j++) {
         printf("Input element [%d][%d] : ", i, j);
           scanf("%d", &mat[i][j]);
        }
    }

    printf("\n\nMatrix before transposing:\n");
    for (i = 0; i < m; i++) {
```

```
        for (j = 0; j < n; j++) {
         printf("%3d", mat[i][j]);
      }
        printf("\n");
    }


      for ( i=0; i<m; ++i ) {
          for ( j=i+1; j<n; ++j ) {
              temp = mat[i][j];
               mat[i][j] = mat[j][i];
               mat[j][i] = temp;
        }
    }


    printf("\nMatrix after transposing:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
              printf("%3d", mat[i][j]);
        }
        printf("\n");
    }
} // main
```

Example of program's execution:

```
Input number of rows < 50: 3
Input number of columns < 50: 2
Input of matrix elements :
Input element [0][0] : 1
Input element [0][1] : 2
Input element [1][0] : 3
Input element [1][1] : 4
Input element [2][0] : 5
Input element [2][1] : 6


Matrix before transposing:

1    2
3    4
5    6


Matrix after transposing:

1    3    5
2    4    6
```

Example: Write your own C program that stores real matrix whose dimensions are 10x10, and finds the sum of elements from every column and product of elements from every row. Program prints the smallest sum (including parent column's index), and biggest product (including parent row's index). Sums and products should be stored in one-dimensional arrays.

```
#include <stdio.h>
#define NR_ROW     10
#define NR_COL     10

int main(void) {
  int     i, j;
  int     min_sum_ind, max_prod_ind;
   float     mat[NR_ROW][NR_COL];
   float     sum[NR_COL], prod[NR_ROW];

/*1.variant of input and calculating sums & products*/

   for (i = 0; i < NR_ROW; i++) {
      for (j = 0; j < NR_COL; j++) {
        printf("\nInput element[%d][%d] : ", i, j);
         scanf("%f", &mat[i][j]);
      }
   }

   for (j = 0; j < NR_COL; j++) {
         sum[j] = 0;
          for (i = 0; i < NR_ROW; i++) {
            sum[j] += mat[i][j];
          }
   }

   for (i = 0; i < NR_ROW; i++) {
    prod[i] = 1;
    for (j = 0; j < NR_COL; j++) {
              prod[i] *= mat[i][j];
```

```
      }
    }

/*end of input, and sums & products calculation*/


/* finding column's index for smallest sum */

    min_sum_ind = 0;
    for (j = 1; j < NR_COL; j++) {
        if (sum[j] < sum[min_sum_ind]) {
            min_sum_ind = j;
        }
    }


    /* finding row's index for biggest product */
    max_prod_ind = 0;
    for (i = 1; i < NR_ROW; i++) {
        if (prod[i] > prod[max_prod_ind]) {
            max_prod_ind = i;
        }
    }

    printf("\nSmallest sum: %f, parent index: %d\n",
            sum[min_sum_ind], min_sum_ind);
    printf("\nBiggest product: %f, parent index: %d\n",
            prod[max_prod_ind], max_prod_ind);
}
```

Example

Program reads student’s grade given from keyboard (from 1 to 5) and prints it’s description.

```
#include <stdio.h>

void main() {

	int grade;
	printf ("Input grade :");
	scanf("%d", & grade);

	switch (grade) {

				case 1:
					printf("Fall (F)\n");break;
				case 2:
					printf("Bad (D)\n");break;
				case 3:
					printf("Good (C)\n");break;
				case 4:
					printf("Very Good (B)\n");break;
				case 5:
					printf("Excellent (A)\n");break;
				default:
					printf("You have inputted false grade\n");
```

```
                                    break;       // break isn't necessary here
        }
}
```