Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра теоретичних основ радіотехніки

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №4**

з дисципліни: «Інформатика 2»

|  |  |
| --- | --- |
|  | Виконав : Першко Федір Сергійович  Група: РЕ-11  Викладачі: доцент Катін П.Ю.  Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Підпис: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2021

**Мета роботи**: Написати програму для роботи з динамічними двовимірними масивами та реалізувати рішення наведених завдань у вигляді окремих функцій.

**Ключові моменти**:

1. Виділення пам’яті для роботи з деяким масивом

int \*\*matrix\_init( int rows, int cols, unsigned int var1){

unsigned int i, j;

int \*\*matrix = (int) malloc(rows\*sizeof(int\*));

for (i=0; i < rows; i++){

matrix[i] = (int\*)malloc(cols\*sizeof(int));

}

for (i = 0; i < rows; i++)

{

for (j = 0; j < cols; j++)

{

if (var1 == 1)

{

printf("\nEnter your matrix element[%d] [%d]\n", i, j);

scanf("%u", &matrix [i][j]);

}

else {matrix [i][j] = rand () % 31;}

}

}

return matrix;

}

1. Створення меню для вибору задачі програми

do {

printf("\n\n\Select the task of the program\n");

printf("1. Find the maximum element of the matrix A and the minimum of the elements below (above) the main diagonal\n");

printf("2. Transpose the matrix B.\n");

printf("3. Find the matrix product A x B\n");

printf("4. Sort in ascending order the elements of the array A specified from the keyboard (indexing starts from zero)\n");

printf("5. Display the sum of the elements of rows of matrix A and columns of matrix B\n");

scanf("%u", &select);

if (select != 1 && select != 2 && select != 3 && select != 4 && select != 5){

printf("\n\tYou are mistaken\n");

}

}while (select != 1 && select != 2 && select != 3 && select != 4 && select != 5);

1. Функція для очищення пам’яті для роботи з деяким масивом

void clear\_memory ( int \*\*matrix, int rows)

{

int i;

for (i = 0; i < rows; i++)

{

free (matrix[i]);

}

free (matrix);

}

**Код:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int maximum (int \*\*A, int Na);

int minimum (int \*\*A, int Na);

void transponate(int \*\*B, int rows, int cols);

void product (int \*\*A, int \*\*B, int N);

void sort (int \*\*matrix, int rows, int cols);

void sum\_rows (int \*\*matrix, int rows, int cols);

void sum\_cols (int \*\*matrix, int rows, int cols);

void print\_matrix (int \*\*matrix, int rows, int cols);

int \*\*matrix\_init( int rows, int cols, unsigned int var1){

unsigned int i, j;

int \*\*matrix = (int) malloc(rows\*sizeof(int\*));

for (i=0; i < rows; i++){

matrix[i] = (int\*)malloc(cols\*sizeof(int));

}

for (i = 0; i < rows; i++)

{

for (j = 0; j < cols; j++)

{

if (var1 == 1)

{

printf("\nEnter your matrix element[%d] [%d]\n", i, j);

scanf("%u", &matrix [i][j]);

}

else {matrix [i][j] = rand () % 31;}

}

}

return matrix;

}

void clear\_memory ( int \*\*matrix, int rows)

{

int i;

for (i = 0; i < rows; i++)

{

free (matrix[i]);

}

free (matrix);

}

int main()

{

unsigned int Na, Nb, Mb, var1, select, var2 = 1;

int \*\*A, \*\*B;

printf("\n\t\*==================================\*");

printf("\n\t| lab4 |");

printf("\n\t\*----------------------------------\*");

printf("\n\t| dynamic array |");

printf("\n\t\*==================================\*\n");

printf("\n\n\tEnter side of matrix A\nNa = ");

scanf("%u", &Na);

printf("\n\n\tEnter number of columns of matrix B\nNb = ");

scanf("%u", &Nb);

printf("\n\n\tEnter number of rows of matrix B\nMb = ");

scanf("%u", &Mb);

srand(time(NULL));

printf("\n\tHow would you like to enter the matrix elements?\n\tPrint 1, if manually, or print any other number, if auto");

scanf("%u", &var1);

A = matrix\_init(Na, Na, var1);

printf("\n\tMatrix A:\n");

print\_matrix(A, Na, Na);

if (var1 == 1){printf("\n\n\tNow enter the B-matrix elements\n");}

B = matrix\_init(Mb, Nb, var1);

printf("\n\tMatrix B:\n");

print\_matrix(B, Mb, Nb);

do{

do {

printf("\n\n\Select the task of the program\n");

printf("1. Find the maximum element of the matrix A and the minimum of the elements below (above) the main diagonal\n");

printf("2. Transpose the matrix B.\n");

printf("3. Find the matrix product A x B\n");

printf("4. Sort in ascending order the elements of the array A specified from the keyboard (indexing starts from zero)\n");

printf("5. Display the sum of the elements of rows of matrix A and columns of matrix B\n");

scanf("%u", &select);

if (select != 1 && select != 2 && select != 3 && select != 4 && select != 5){

printf("\n\tYou are mistaken\n");

}

}while (select != 1 && select != 2 && select != 3 && select != 4 && select != 5);

switch (select)

{

case 1:

{

system ("cls");

printf("\n\tMatrix A:\n");

print\_matrix (A, Na, Na);

printf("\n\tThe maximum A element is %d", maximum(A, Na));

printf("\n\tThe minimum of the elements below (above) the main diagonal is %d", minimum(A, Na));

} break;

case 2:

{

system ("cls");

printf("\n\tMatrix B:\n");

print\_matrix (B, Mb, Nb);

transponate(B, Mb, Nb);

}break;

case 3:

{

if (Na == Mb)

{

system ("cls");

printf("\n\tMatrix A:\n");

print\_matrix (A, Na, Na);

printf("\n\tMatrix B:\n");

print\_matrix (B, Mb, Nb);

product (A, B, Na);

}

else{printf("\n\n\tThis is impossible");}

} break;

case 4:

{

system ("cls");

printf("\n\tMatrix A:\n");

print\_matrix (A, Na, Na);

sort (A, Na, Na);

} break;

case 5:

{

system ("cls");

printf("\n\tMatrix A:\n");

print\_matrix (A, Na, Na);

printf("\n\tMatrix B:\n");

print\_matrix (B, Mb, Nb);

sum\_rows (A, Na, Na);

printf("\n");

sum\_cols (B, Mb, Nb);

} break;

}

clear\_memory(A, Na);

clear\_memory(B, Mb);

printf("\n\n\tWould you like to do any task one more time?\n\tprint 1, if yes, or other number to exit");

scanf("%d", &var2);

system ("cls");

}while (var2 == 1);

}

int maximum (int \*\*A, int Na)

{

int max = A[0][0];

int i, j;

for (i = 0; i < Na; i++)

{

for (j = 0; j < Na; j++)

{

if (A [i][j] > max)

{

max = A [i][j];

}

}

}

return max;

}

int minimum (int \*\*A, int Na)

{

int min = A[0][0];

int i, j;

for (i = 0; i < Na; i++)

{

for (j = 0; j < Na; j++)

{

if (i != j)

{

if (A [i][j] < min)

{

min = A [i][j];

}

}

}

}

return min;

}

void transponate(int \*\*B, int rows, int cols)

{

int i, j;

printf("\n\n\tTransposed matrix B:\n");

for (i = 0; i < cols; i++)

{

for (j = 0; j < rows; j++)

{

printf("\t %d", B[j][i]);

}

printf("\n");

}

}

void product (int \*\*A, int \*\*B, int N)

{

unsigned int i, j, k;

int \*\*C;

C = matrix\_init (N, N, 0);

for(i = 0; i < N; i++)

{

for(j = 0; j < N; j++)

{

C[i][j] = 0;

for (k = 0; k < N; k++)

{

C[i][j] += A[i][k] \* B[k][j];

}

}

}

printf("\n\tThe result of multiplying:\n");

print\_matrix (C, N, N);

clear\_memory (C, N);

}

void sort (int \*\*matrix, int rows, int cols)

{

int i, j, M, temp;

do

{

printf("\n\n\tChoose the row to sort\n");

scanf("%u", &M);

if (M < 0 && M >= rows){printf("\nYou are mistaken"); }

}while (M < 0 && M >= rows);

for(i = 0 ; i < cols - 1; i++)

{

for(j = 0 ; j < cols - i - 1 ; j++)

{

if(matrix[M][j] > matrix[M][j+1])

{

temp = matrix[M][j];

matrix[M][j] = matrix[M][j+1];

matrix[M][j+1] = temp;

}

}

}

printf("\nThe result of sorting is:\n");

for (i = 0; i < cols; i++)

{

printf("%d ", matrix[M][i]);

printf("\t");

}

}

void sum\_rows (int \*\*matrix, int rows, int cols)

{

int i, j, sum = 0;

for (i = 0; i < rows; i++)

{

for (j = 0; j < cols; j++)

{

sum += matrix[i][j];

}

printf("\nSum of A-matrix row %d is %d", i, sum);

sum = 0;

}

}

void sum\_cols (int \*\*matrix, int rows, int cols)

{

int i, j, sum = 0;

for (j = 0; j < cols; j++)

{

for (i = 0; i < rows; i++)

{

sum += matrix[i][j];

}

printf("\nSum of B-matrix column %d is %d", j, sum);

sum = 0;

}

}

void print\_matrix (int \*\*matrix, int rows, int cols)

{

int i, j;

for (i = 0; i < rows; i++)

{

for (j = 0; j < cols; j++)

{

printf("\t%d", matrix[i][j]);

}

printf("\n");

}

}

**Висновок:**

Я навчився роботі із двовимірними динамічними масивами, а саме виділення для них пам’яті, її очищення, та передавання динамічних масивів до окремих функцій для подальшої там роботи.