

**МIНIСТЕРСТВО  ОСВIТИ І НАУКИ  УКРАЇНИ**

**НАЦІОНАЛЬНИЙ   ТЕХНІЧНИЙ УНІВЕРСИТЕТ   УКРАЇНИ**

“**КИЇВСЬКИЙ  ПОЛІТЕХНІЧНИЙ  ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

**Лабораторна робота № 4**

**з дисципліни “ Основи програмування ”**

**тема “ Одновимірні та двовимірні масиви даних ”**

|  |  |  |
| --- | --- | --- |
| **Виконала**  **студентка I курсу**  **групи КП-83**  **Снітко Маріанна Дмитрівна**  **(прізвище, ім’я, по батькові)**  **варіант № 22** |  | **Перевірив**  “**\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.**  **викладач**  **Гадиняк Руслан Анатолійович**  **(прізвище, ім’я, по батькові)** |

**Київ 2018**

**Мета роботи**

Навчитися працювати зі масивами різних типів даних та розмірності.  
Застосувати на практиці різні види циклічних конструкцій при роботі з одновимірними та багатовимірними масивами даних.  
Навчитися оформлювати консольну програму для зручності роботи користувача.

**Постановка завдання**

При старті консольної програми користувачу вивести меню із 3-х пунктів, наприклад:

1. Array

2. Matrix

3. Quit

Реалізувати навігацію по меню за допомогою вкладених циклів. Із кожного підменю можна повернутись у головне меню.

Управління переходами між меню реалізувати за допомогою обробки вводу користувача. Весь ввід користувача перевіряти на валідність (чи допустиме від'ємне значення або чи введене значення є індексом поточного масиву).

#### **Підменю 1. Одномірний масив**

Перед переходом у дане меню користувачу необхідно ввести значення N - ціле додатнє число.  
У цьому меню у користувача є одномірний масив дробових чисел (розміром N елементів) ініціалізований нулями.

Вивести масив у консоль та запропонувати користувачу такі операції над ним:

1. Заповнити масив випадковими числами від L до H.
2. Обнулити всі елементи масиву.
3. Знайти максимальний елемент масиву та його індекс.
4. Вивести суму додатніх елементів масиву.
5. Поміняти місцями значення максимального і мінімального елементів масиву.
6. Помножити всі елементи масиву на введене число.

#### **Підменю 2. Двомірний масив**

Перед переходом у дане меню користувачу необхідно ввести значення N та M - цілі додатні числа.  
У цьому меню у користувача є двомірний масив цілих чисел (матриця NхM) ініціалізований нулями.

Вивести матрицю у консоль та запропонувати користувачу такі операції над нею:

1. Заповнити масив випадковими числами від L до H.
2. Обнулити всі елементи масиву.
3. Знайти мінімальний елемент та його індекси (i та j).
4. Знайти суму елементів рядка за заданим індексом.
5. Поміняти місцями максимальний і мінімальний елементи масиву.
6. Змінити значення елементу за вказаними індексами на задане.

#### **Підменю 2+3/4 (Бонус)**

Цей пункт не є обов'язковим.

У додатках завдання задано матрицю кодів кольорів (можна розмістити її як глобальну змінну) та функцію для отримання кольору на основі коду цього кольору.

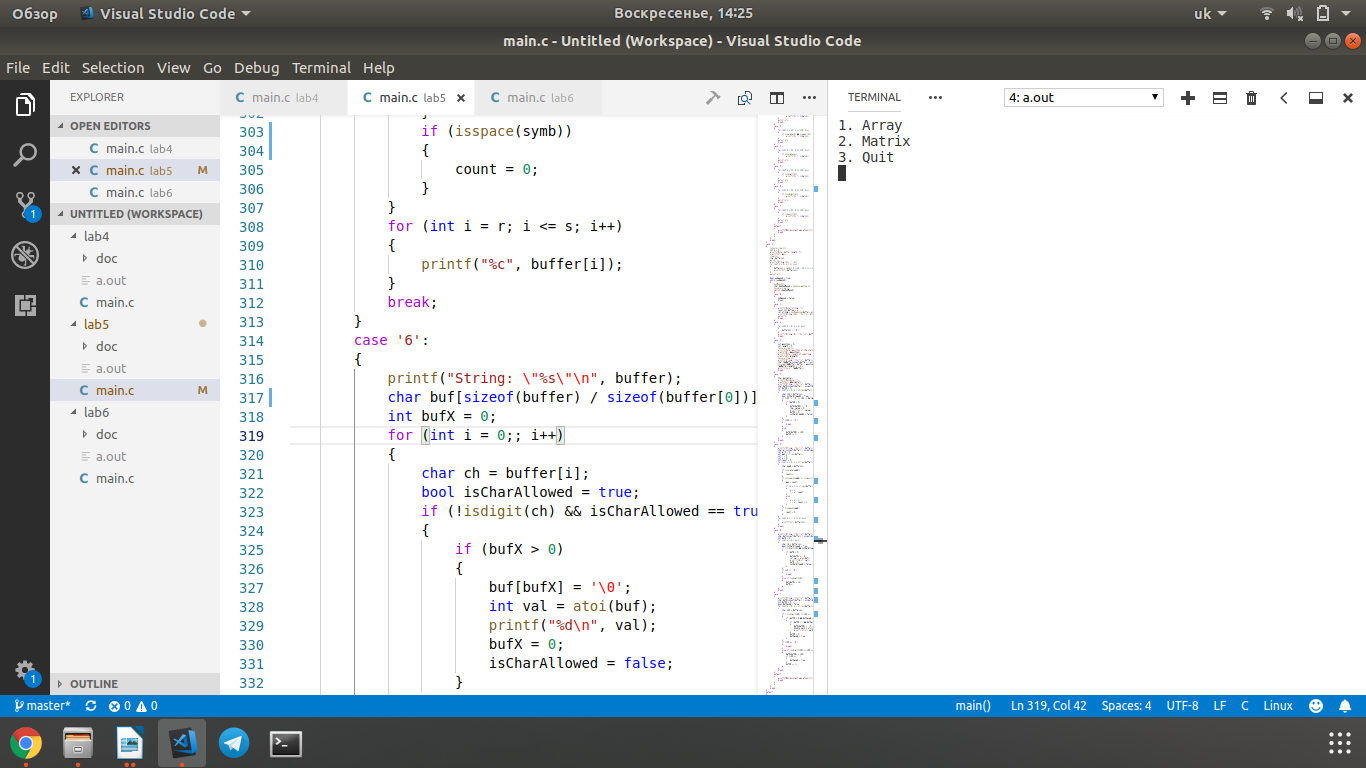
У даному підменю вивести зображення, що закодоване, у консоль.

**Тексти коду програм**

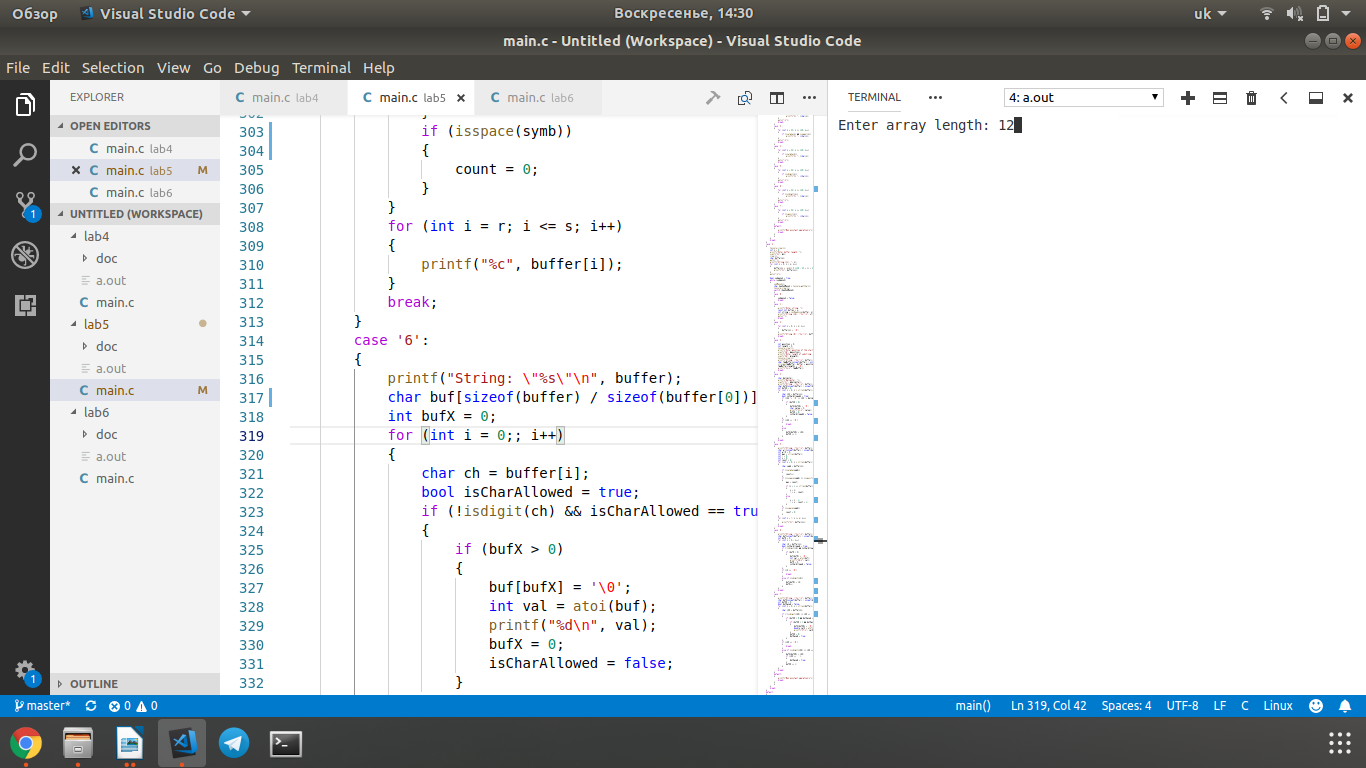
|  |
| --- |
| **main.c** |
| #include <stdio.h>  #include <math.h>  #include <stdlib.h>  #include <progbase.h>  #include <progbase/console.h>  #include <time.h>  #include <stdbool.h>  void mainMenu(void)  {  printf("1. Array\n2. Matrix\n3. Quit\n");  }  void menuArr(void)  {  printf("1. Fill array with random numbers from L to H\n");  printf("2. Reset array\n");  printf("3. Find maximal element of array and its index\n");  printf("4. Output the sum of positive elements\n");  printf("5. Swap places of maximal and minimal elements of array\n");  printf("6. Multiply all elements of array by entered number\n");  printf("7. < Back to main menu\n");  }  void menuMatrix(void)  {  printf("1. Fill matrix with random numbers from L to H\n");  printf("2. Reset matrix\n");  printf("3. Find minimal element of matrix and its indexes i and j\n");  printf("4. Output the sum of elements of row\n");  printf("5. Swap places of maximal and minimal elements of matrix\n");  printf("6. Replace element\n");  printf("7. < Back to main menu\n");  }  int getColor(char colorCode);  void clear()  {  while (getchar() != '\n')  ;  }  const char image[28][28] = {  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0xF, 0xF, 0xF, 0xF, 0xF, 0xF, 0xF, 0xF, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xF, 0xF, 0xF, 0x0, 0x0, 0x0, 0x0, 0xF, 0xF, 0xF, 0xF, 0xF, 0xF, 0xF},  {0xF, 0xF, 0xF, 0xF, 0xF, 0xF, 0x0, 0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0xF, 0x0, 0x4, 0x4, 0x4, 0x4, 0x0, 0xF, 0xF, 0xF, 0xF, 0xF, 0xF},  {0xF, 0xF, 0xF, 0xF, 0xF, 0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0xF, 0xF, 0xF, 0xF, 0xF},  {0xF, 0xF, 0xF, 0xF, 0x0, 0x0, 0x4, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x4, 0x4, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xF, 0xF, 0xF, 0xF, 0xF},  {0xF, 0xF, 0x0, 0x0, 0x4, 0x4, 0x4, 0x0, 0x0, 0x4, 0x4, 0x4, 0x4, 0x0, 0x0, 0x4, 0x0, 0x0, 0x4, 0x4, 0x0, 0x0, 0x4, 0x0, 0x0, 0xF, 0xF, 0xF},  {0xF, 0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0x0, 0x0, 0x4, 0x4, 0x0, 0x4, 0x4, 0x4, 0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0xF, 0xF},  {0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0x0, 0x4, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xF, 0x0, 0xF},  {0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0x0, 0xF, 0x0, 0x0, 0x0, 0xF, 0xF, 0x0, 0xF, 0x0, 0x0, 0xF, 0xF, 0x0, 0xF, 0xF, 0x0},  {0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0xF, 0xF, 0x0, 0x0, 0xF, 0x0, 0x0, 0x0, 0x0, 0xF, 0x0, 0xF, 0x0, 0x0, 0xF, 0xF, 0x0, 0xF},  {0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0x0, 0xF, 0xF, 0x0, 0x0, 0xF, 0x0, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xF, 0xF},  {0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0xF, 0xF, 0xF},  {0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0x0, 0x0, 0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xF},  {0x0, 0x4, 0x4, 0x4, 0x4, 0x4, 0x0, 0x2, 0x2, 0x2, 0x2, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x0},  {0xF, 0x0, 0x0, 0x4, 0x4, 0x4, 0x0, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xF},  {0xF, 0xF, 0xF, 0x0, 0x0, 0x4, 0x4, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x4, 0x0, 0x0, 0xF, 0xF, 0xF, 0xF},  {0xF, 0xF, 0xF, 0xF, 0xF, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xF, 0xF, 0xF, 0xF, 0xF, 0xF},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1},  {0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1, 0x1}};  int main()  {  int L = 0;  int H = 0;  int N = 0;  int M = 0;  int index = 0;  int number = 0;  int n = 0;  float max = 0;  int min = 0;  int minInd1 = 0;  int minInd2 = 0;  int row = 0;  float maxim = 0;  float minim = 0;  int maxInd = 0;  int minInd = 0;  srand(time(0));  bool isMainMenuRunning = true;  while (isMainMenuRunning)  {  Console\_clear();  mainMenu();  char mainMenuUserInput = Console\_getChar();  switch (mainMenuUserInput)  {  case '3':  {  isMainMenuRunning = false;  break;  }  case '1':  {  Console\_clear();  int n = 0;  printf("Enter array length: ");  scanf("%d", &n);  clear();  while (n < 0)  {  printf("Enter array length: ");  scanf("%d", &n);  clear();  Console\_clear();  };  float arr[n];  for (int i = 0; i < n; i++)  {  arr[i] = 0;  }  bool isSubmenu1Running = true;  while (isSubmenu1Running)  {  puts("");  puts("Array: ");  for (int i = 0; i < n; i++)  {  printf("%.2f ", arr[i]);  }  puts(" ");  menuArr();  char submenu1Input = Console\_getChar();  Console\_clear();  switch (submenu1Input)  {  case '7':  {  isSubmenu1Running = false;  break;  }  case '1':  {  do  {  Console\_clear();  printf("Enter L-min: ");  scanf("%i", &L);  puts("");  printf("Enter H-max: ");  scanf("%i", &H);  } while (H - L < 0);  Console\_clear();  for (int i = 0; i < n; i++)  {  arr[i] = L + (rand() / (float)RAND\_MAX) \* (H - L);  }  break;  }  case '2':  {  for (int i = 0; i < n; i++)  {  arr[i] = 0;  }  break;  }  case '3':  {  max = arr[0];  for (int i = 0; i < n; i++)  {  if (arr[i] > max)  {  max = arr[i];  index = i;  }  }  printf("Index: [%d]\nElement: %.3f\n", index, max);  break;  }  case '4':  {  float sum = 0;  for (int i = 0; i < n; i++)  {  if (arr[i] > 0)  {  sum += arr[i];  }  }  printf("Sum = %5.2f\n", sum);  break;  }  case '5':  {  maxim = arr[0];  minim = arr[0];  for (int i = 0; i < n; i++)  {  if (arr[i] > maxim)  {  maxim = arr[i];  maxInd = i;  }  else if (arr[i] < minim)  {  minim = arr[i];  minInd = i;  }  }  float tmp = arr[maxInd];  arr[maxInd] = arr[minInd];  arr[minInd] = tmp;  break;  }  case '6':  {  printf("Multiplier:\n");  scanf("%i", &number);  for (int i = 0; i < n; i++)  {  arr[i] \*= number;  }  break;  }  default:  {  printf("Key does not exist\n");  break;  }  }  }  break;  }  case '2':  {  Console\_clear();  int N = 0;  int M = 0;  do  {  Console\_clear();  printf("Enter N - rows: ");  scanf("%i", &N);  puts("");  printf("Enter M - columns: ");  scanf("%i", &M);  Console\_clear();  } while (M < 1 || N < 1);  int matrix[N][M];  for (int n = 0; n < N; n++)  {  for (int m = 0; m < M; m++)  {  matrix[n][m] = 0;  }  }  bool isSubmenu2Running = true;  while (isSubmenu2Running)  {  puts("");  puts("Matrix:\n");  for (int n = 0; n < N; n++)  {  for (int m = 0; m < M; m++)  {  printf("%5i ", matrix[n][m]);  }  printf("\n");  }  puts("");  menuMatrix();  char submenu2Input = Console\_getChar();  Console\_clear();  switch (submenu2Input)  {  case '7':  {  isSubmenu2Running = false;  break;  }  case '1':  {  do  {  Console\_clear();  printf("Enter L-min: ");  scanf("%i", &L);  puts("");  printf("Enter H-max: ");  scanf("%i", &H);  Console\_clear();  } while (H - L < 0);  for (int n = 0; n < N; n++)  {  for (int m = 0; m < M; m++)  {  matrix[n][m] = rand() % (H - L + 1) + L;  }  }  break;  }  case '2':  {  for (int n = 0; n < N; n++)  {  for (int m = 0; m < M; m++)  {  matrix[n][m] = 0;  }  }  break;  }  case '3':  {  min = matrix[0][0];  for (int n = 0; n < N; n++)  {  for (int m = 0; m < M; m++)  {  if (min > matrix[n][m])  {  min = matrix[n][m];  minInd1 = n;  minInd2 = m;  }  }  }  printf("Indexes: [%i][%i] Element: %i\n", minInd1, minInd2, min);  break;  }  case '4':  {  int plus = 0;  Console\_clear();  printf("Index of row: ");  scanf("%d", &row);  puts("");  while (row < 0 || row > N - 1)  {  Console\_clear();  printf("Enter valid index\n");  scanf("%d", &row);  }  for (int n = 0; n < N; n++)  {  plus = plus + matrix[row][n];  }  printf("Sum of elements = %i\n", plus);  break;  }  case '5':  {  int maxMat = 0;  int minMat = 0;  int indMax1 = 0;  int indMin1 = 0;  int indMax2 = 0;  int indMin2 = 0;  minMat = matrix[0][0];  maxMat = matrix[0][0];  for (int n = 0; n < N; n++)  {  for (int m = 0; m < M; m++)  {  if (matrix[n][m] > maxMat)  {  maxMat = matrix[n][m];  indMax1 = n;  indMax2 = m;  }  else if (matrix[n][m] < minMat)  {  minMat = matrix[n][m];  indMin1 = n;  indMin2 = m;  }  }  }  int tmp1 = matrix[indMax1][indMax2];  matrix[indMax1][indMax2] = matrix[indMin1][indMin2];  matrix[indMin1][indMin2] = tmp1;  break;  }  case '6':  {  int indRow;  int indCol;  int digit;  printf("Enter index of row: ");  scanf("%i", &indRow);  while (indRow < 0 || indRow > N - 1)  {  Console\_clear();  printf("Enter valid index\n");  scanf("%i", &indRow);  }  printf("Enter index of column: ");  scanf("%i", &indCol);  while (indCol < 0 || indCol > M - 1)  {  printf("Enter valid index\n");  scanf("%i", &indCol);  }  printf("Digit: ");  scanf("%i", &digit);  for (int n = 0; n < N; n++)  {  for (int m = 0; m < M; m++)  {  matrix[indRow][indCol] = digit;  }  }  break;  }  default:  {  printf("Key does not exist\n");  break;  }  }  }  break;  }  case '4':  {  Console\_clear();  bool consoleDrawing = true;  while (consoleDrawing)  {  for (int i = 0; i < 28; i++)  {  for (int j = 0; j < 28; j++)  {  int color = getColor(image[i][j]);  Console\_setCursorAttribute(color);  Console\_setCursorPosition(i, j);  printf(" ");  printf("%c", image[i][j]);  }  Console\_reset();  puts("\n");  }  printf("1. Back to main menu\n");  int exitMenu3 = 0;  scanf("%i", &exitMenu3);  if (exitMenu3 == 1)  {  consoleDrawing = false;  }  else  {  isMainMenuRunning = false;  }  }  break;  }  }  }  puts("Bye!");  return 0;  }  int getColor(char colorCode)  {  // colors encoding table (hex code -> console color)  const char colorsTable[16][2] = {  {0x0, BG\_BLACK},  {0x1, BG\_INTENSITY\_BLACK},  {0x2, BG\_RED},  {0x3, BG\_INTENSITY\_RED},  {0x4, BG\_GREEN},  {0x5, BG\_INTENSITY\_GREEN},  {0x6, BG\_YELLOW},  {0x7, BG\_INTENSITY\_YELLOW},  {0x8, BG\_BLUE},  {0x9, BG\_INTENSITY\_BLUE},  {0xa, BG\_MAGENTA},  {0xb, BG\_INTENSITY\_MAGENTA},  {0xc, BG\_CYAN},  {0xd, BG\_INTENSITY\_CYAN},  {0xe, BG\_WHITE},  {0xF, BG\_INTENSITY\_WHITE}};  const int tableLength = sizeof(colorsTable) / sizeof(colorsTable[0]);  for (int i = 0; i < tableLength; i++)  {  char colorPairCode = colorsTable[i][0];  char colorPairColor = colorsTable[i][1];  if (colorCode == colorPairCode)  {  return colorPairColor; // we have found our color  }  }  return 0; // it's an error  } |

**Приклади результатів**

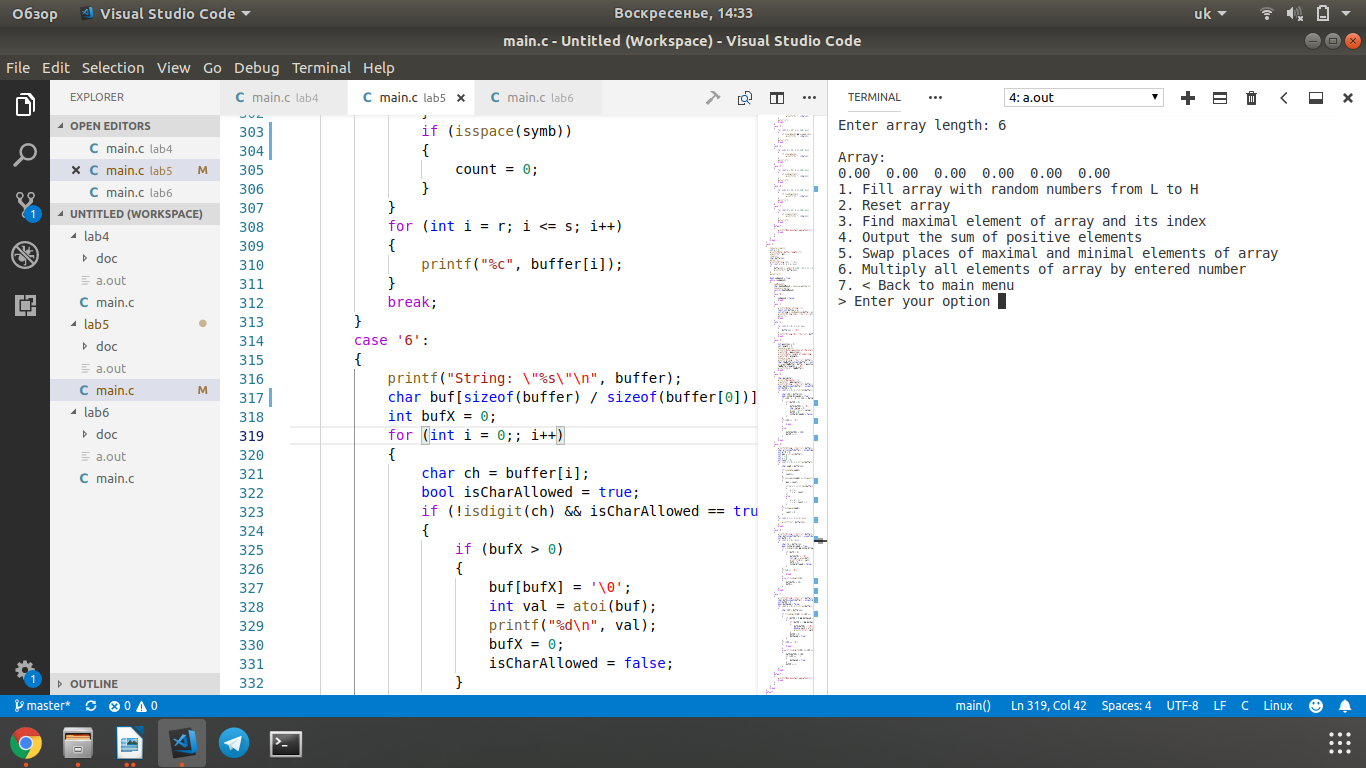
При старті програми у консолі виводиться головне меню:



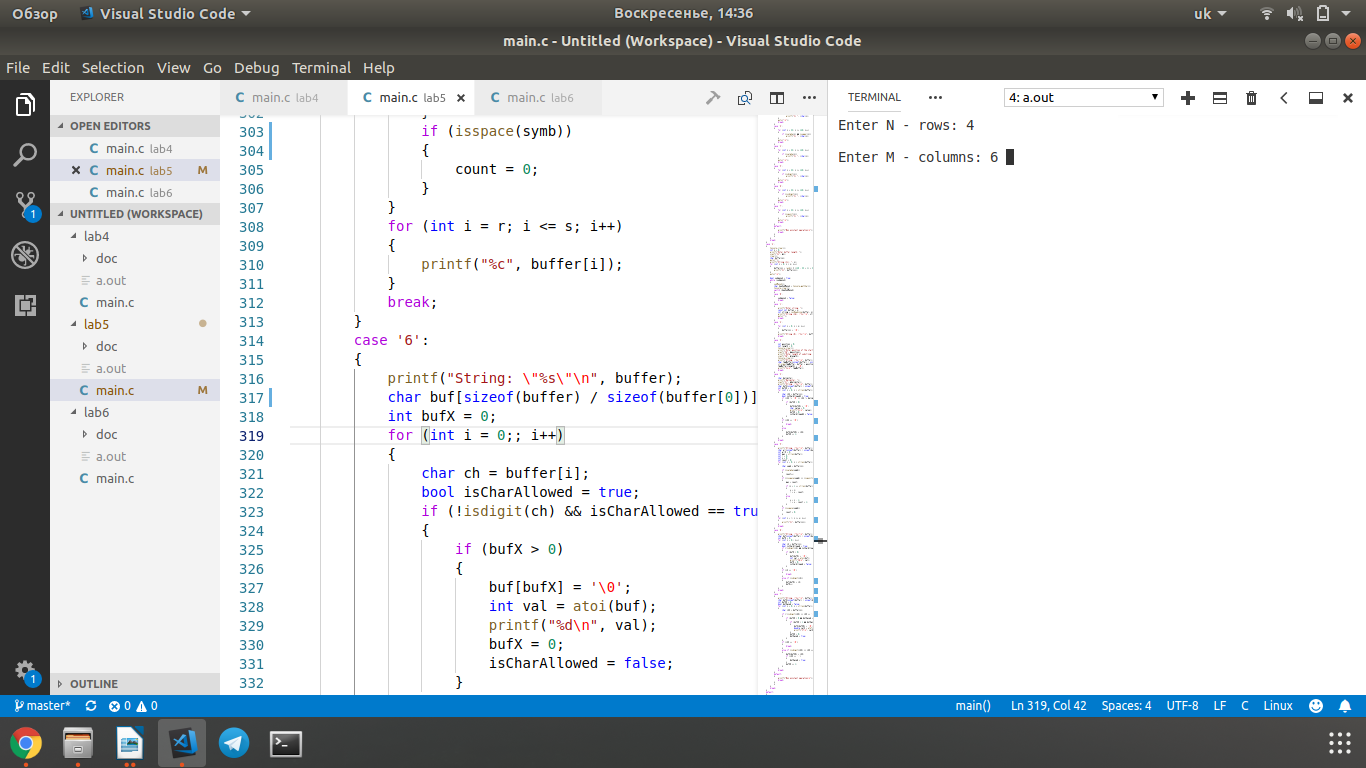
При вході у перше підменю потрібно ввести довжину масиву:



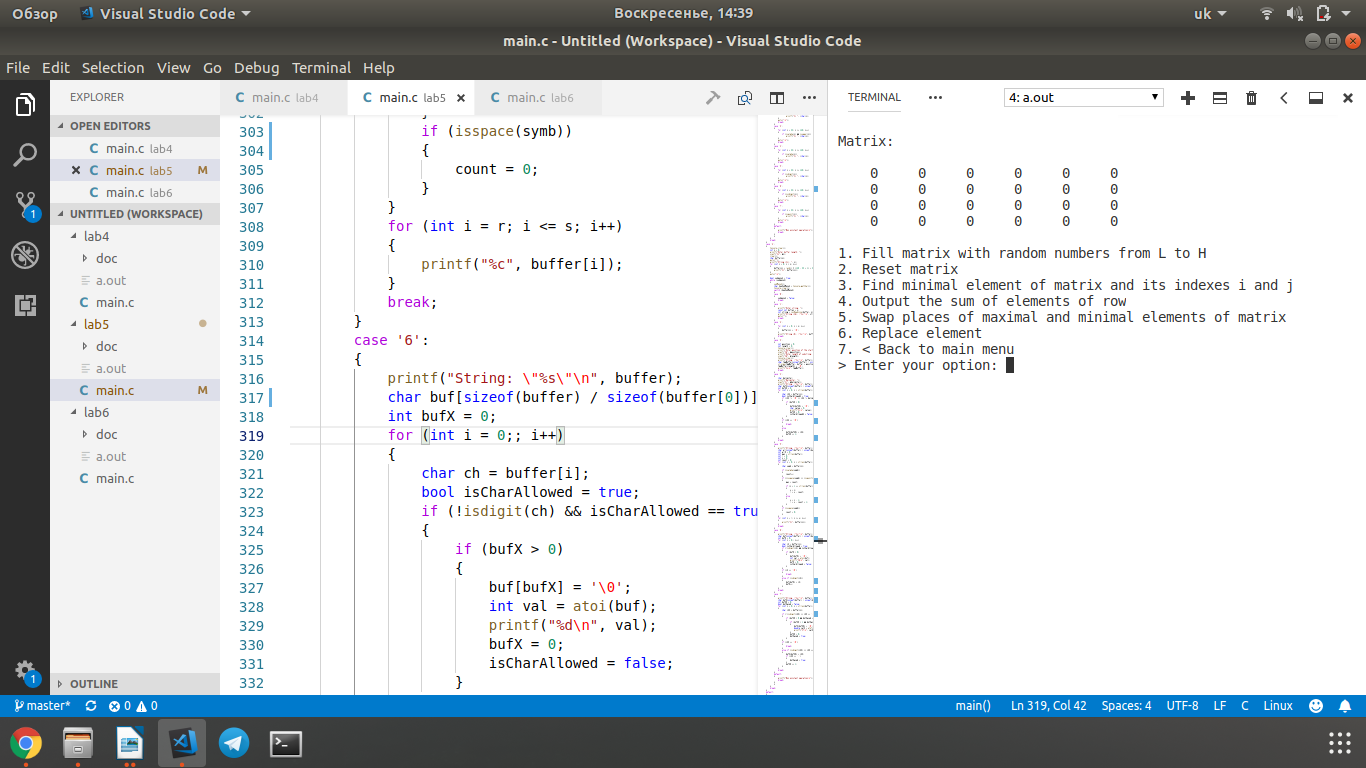
Виводиться масив, ініціалізований нулями і пропонуються операції з ним:



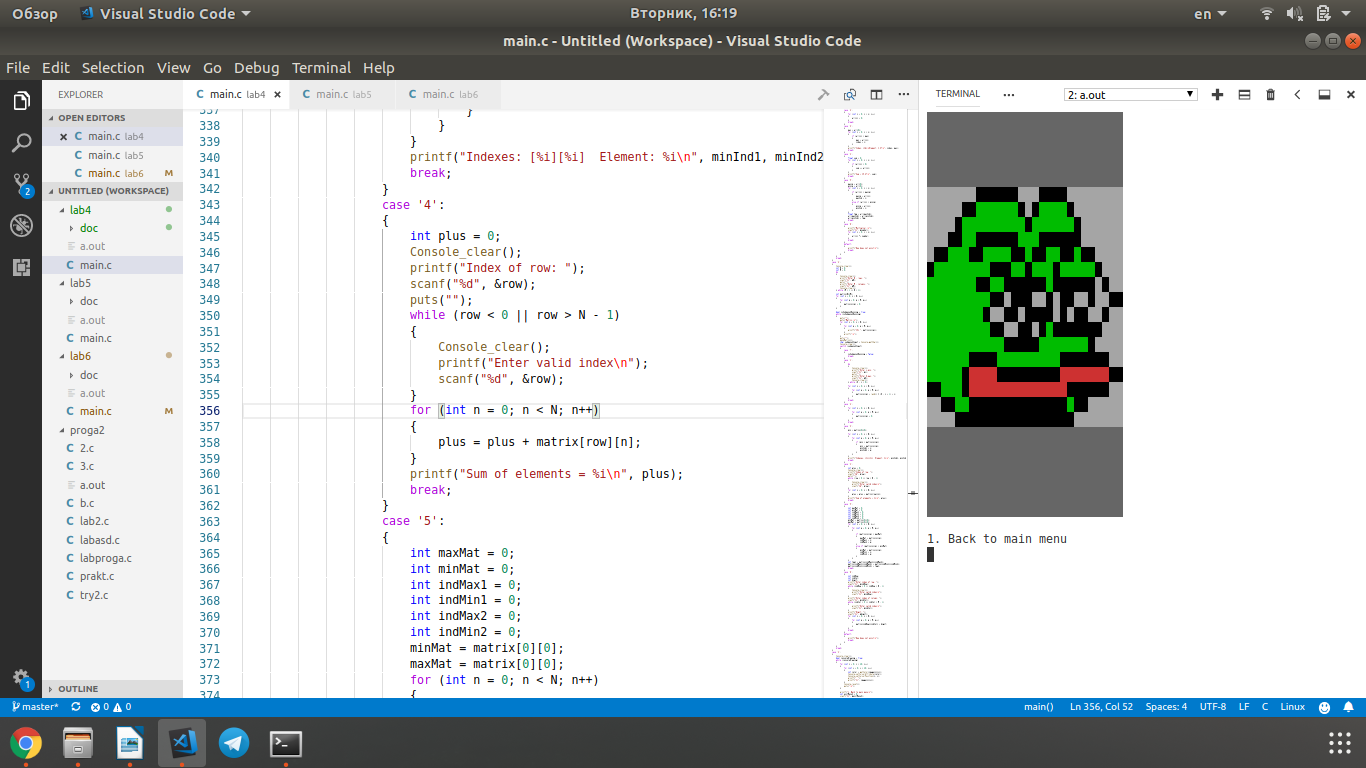
Друге підменю — матриця:



Підменю матриці:



Секретне підменю:



**Висновок**

Протягом виконання цієї лабораторної роботи я ознайомилася із програмуванням на мові С у спеціалізованому редакторі коду.

Більше того, на практиці я навчилася працювати з масивами різних типів даних та розмірності. Також я застосувала різні види циклічних конструкцій при роботі з одновимірними та багатовимірними масивами даних. Окрім цього, мною було засвоєно оформлення консольної програми для зручності роботи користувача.

Компіляція всього коду відбувалася за допомогою компілятора gcc.