

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 4

з дисципліни “ Основи програмування ”

тема “ Структури даних, вказівники, динамічне виділення пам’яті та робота із файлами ”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-52  Саприкін Артем Олексійович  (*прізвище, ім’я, по батькові*)  варіант № 14 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (2 бала)** | **Відп. на теор. питання (1 бал)** | **Відп. на прогр. питання (2 бала)** | |  |  |  | | Сумарний бал:   |  | | --- | |  | |

Київ 2014

**Мета роботи**

Навчитися використовувати cтруктури даних, вказівники, динамічне виділення пам’яті та робота із файлами

1. **Логістика.**

Система описує розташування населених пунктів і дороги – сполучення між ними.

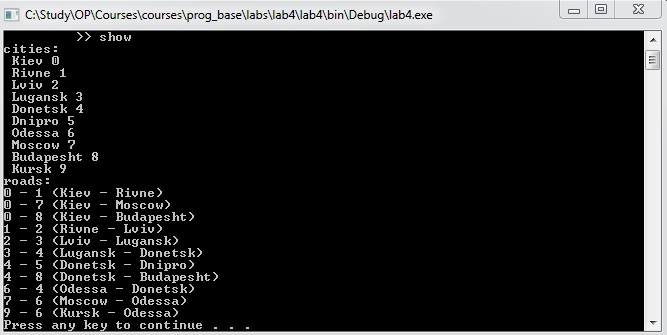
Чи можна дістатися із міста А у місто B? Скільки міст при цьому прийдеться проїхати.

Можливість перейменувати місто. Побудувати нове місто або дорогу між двома містами. Закрити дорогу на проїзд.

**Тексти коду програм**

|  |
| --- |
| Lab4.c |
| #include <stdio.h>  #include <stdlib.h>  typedef struct tagCity  {  char szName[32];  int id;  } City;  void help ()  {  printf ("help - info\n");  printf ("show - show graph\n");  printf ("add %%s - add new city\n");  printf ("rename %%d %%s - rename city by id\n");  printf ("close %%d %%d - close road between two cities\n");  printf ("open %%d %%d - open road between two cities\n");  printf ("dist %%d %%d - calculates the shortest distance between two cities\n");  printf ("exit - exit\n");  }  int main()  {  int resElements;  City\* cities;  int\*\* roads;  int count;  FILE\* pIn = fopen ("graph.dat", "r");  if (pIn == NULL)  {  resElements = 30;  cities = malloc (resElements\*sizeof(City));  roads = malloc (resElements\*sizeof(int\*));  for (int i = 0; i < resElements; i++)  {  roads[i] = malloc (resElements\*sizeof(int));  memset (roads[i], 0, resElements\*sizeof(int));  }  memset (cities, 0, resElements\*sizeof(City));  count = 0;  }  else  {  fscanf (pIn, "%d", &count);  resElements = (count/10)\*10 + 10;  cities = malloc (resElements\*sizeof(City));  roads = malloc (resElements\*sizeof(int\*));  for (int i = 0; i < resElements; i++)  {  roads[i] = malloc (resElements\*sizeof(int));  memset (roads[i], 0, resElements\*sizeof(int));  }  memset (cities, 0, resElements\*sizeof(City));  fread (cities, sizeof(City), count, pIn);  for (int i = 0; i < count; i++)  {  fread (roads[i], sizeof(int), count, pIn);  }  }  char szCommand[256];  while (1)  {  system ("cls");  printf ("Print command (\"help\" for info): \n");  printf ("\t >> ");  gets (szCommand);  if ( strcmp (szCommand, "help") == 0 )  {  help ();  system ("pause");  }  else if ( strncmp (szCommand, "add", 3) == 0 )  {  if (count >= (resElements))  {  printf ("Error");  int d = 30;  resElements += d;  cities = realloc (cities, (resElements\*sizeof(City)));  memset (cities + count, 0,d\*sizeof(City));  roads = realloc (roads, resElements\*sizeof(int\*));  for (int i = 0; i < resElements-d; i++)  {  roads[i] = realloc (roads[i],resElements\*sizeof(int));  memset (roads[i] + count, 0, resElements\*sizeof(int));  }  for (int i = resElements-d; i < resElements; i++)  {  roads[i] = malloc (resElements\*sizeof(int));  memset (roads[i] + count + 1, 0, resElements\*sizeof(int));  }  }  //printf ("\n\n\n%d\n\n\n", sizeof(cities));  City city;  strcpy (city.szName, szCommand + 4);  city.id = count;  \*(cities + count++) = city;  //system ("pause");  }  else if ( strcmp (szCommand, "show") == 0 )  {  printf ("cities: \n");  for (int i = 0; i < count; i++)  printf (" %s %d \n", cities[i].szName, cities[i].id);  printf ("roads: \n");  for (int i = 0; i < count; i++)  for (int j = 0; j < count; j++)  if (roads[i][j] == 1)  printf ("%d - %d (%s - %s)\n", i, j, cities[i].szName, cities[j].szName);  system ("pause");  }  else if ( strncmp (szCommand, "rename", 6) == 0 )  {  int id = 0;  char szNewName[256];  sscanf (szCommand + 7, "%d %s", &id, szNewName);  strcpy (cities[id].szName, szNewName);  }  else if ( strncmp (szCommand, "open", 4) == 0 )  {  int id1 = 0, id2 = 0;  sscanf (szCommand + 5, "%d %d", &id1, &id2);  roads[id1][id2] = 1;  }  else if ( strncmp (szCommand, "close", 5) == 0 )  {  int id1 = 0, id2 = 0;  sscanf (szCommand + 6, "%d %d", &id1, &id2);  roads[id1][id2] = 0;  }  else if ( strcmp (szCommand, "exit") == 0 )  break;  else if ( strncmp (szCommand, "dist", 4) == 0 )  {  int from = 0, to = 0;  sscanf (szCommand + 5, "%d %d", &from, &to);  int\* q = malloc (2\*count\*sizeof(int) + 16);  int\* visited = malloc (count\*sizeof(int));  int\* parent = malloc (count\*sizeof(int));  int\* way = malloc (count\*sizeof(int));  int top = 0;  int down = 0;  memset (q, 0, count\*sizeof(int));  memset (visited, 0, count\*sizeof(int));  for (int i = 0; i < count; i++)  parent[i] = -1;  q[top++] = from;  visited[from] = 1;  parent[from] = -1;  while (top != down)  {  int current = q[down];  q[down++] = -1;  for (int i = 0; i < count; i++)  {  if (i == current)  continue;  if (roads[current][i] == 1)  {  if (!visited[i])  {  q[top++] = i;  visited[i] = 1;  parent[i] = current;  }  }  }  }  int r = 0;  if (parent[to] == -1)  printf ("No way\n");  else  {  for (int i = to; i != -1; i = parent[i])  way[r++] = i;  printf ("The shortest distance: %d\n", r);  printf ("Way: ");  printf ("%d", way[r-1]);  for (int i = r - 2; i >= 0; i--)  printf (" - %d", way[i]);  printf ("\n");  }  free (way);  free (q);  free (visited);  free (parent);  system ("pause");  }  else  {  printf ("unrecognized command\n");  system ("pause");  }  }  FILE\* pOut = fopen ("graph.dat", "w");  fprintf(pOut, "%d", count);  fwrite (cities, sizeof(City), count, pOut);  for (int i = 0; i < resElements; i++)  fwrite (roads[i], sizeof(int), count, pOut);  free (cities);  for (int i = 0; i < resElements; i++)  free (roads[i]);  free (roads);  return 0;  } |

**Приклади результатів**



**Висновки**

У ході лабораторної було розглянуто cтруктури даних, вказівники, динамічне виділення пам’яті та робота із файлами