Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №8

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Унарные и бинарные операции над графами»

Выполнили студенты группы 20ВВ3:

Мальков И.А.

Педай Н.Д.

Приняли:

Митрохин М.А.

Юрова О.В.

Пенза 2021

**Лабораторное задание**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) две матрицы M 1,

М 2 смежности неориентированных помеченных графов G 1, G 2. Выведите

сгенерированные матрицы на экран.

2. \* Для указанных графов преобразуйте представление матриц

смежности в списки смежности. Выведите полученные списки на экран.

**Задание 2**

1. Для матричной формы представления графов выполните операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

2. \* Для представления графов в виде списков смежности выполните

операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

**Задание 3**

1. Для матричной формы представления графов выполните операцию:

а) объединения G = G 1 

G 2

б) пересечения G = G 1 

G 2

в) кольцевой суммы G = G 1  G 2

Результат выполнения операции выведите на экран.

**Задание 4 \***

1. Для матричной формы представления графов выполните операцию

декартова произведения графов G = G 1 X G 2 .

Результат выполнения операции выведите на экран.

**Ход работы:**

1. Код программы:Начало формы

Конец формы

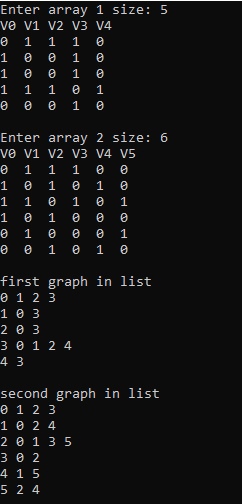
Начало формы

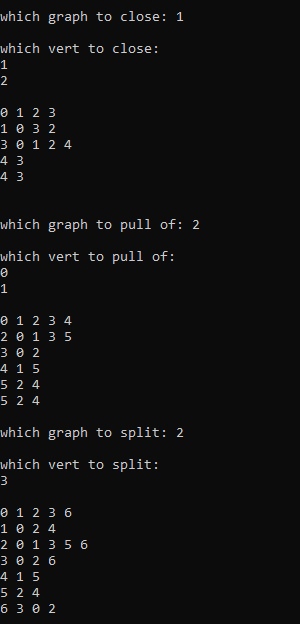
Конец формы

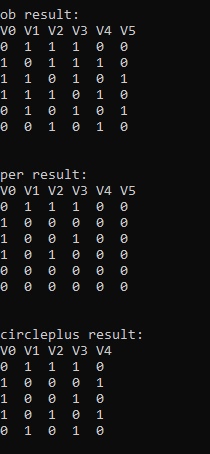
|  |  |
| --- | --- |
|  | #include <stdio.h> |
|  | #include <iostream> |
|  | #include <locale.h> |
|  | #include <stdlib.h> |
|  | #include <time.h> |
|  | #include <stack> // стек |
|  | #pragma warning(disable : 4703); |
|  | using namespace std; |
|  |  |
|  | int\*\* create(int n) |
|  | { |
|  | int\*\* G, k = 1; |
|  | G = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | srand(time(NULL)); |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | G[i] = (int\*)malloc(n \* sizeof(int)); |
|  | G[i][i] = 0; |
|  | for (int j = k; j < n; j++) |
|  | { |
|  | if (rand() % 100 > 50) |
|  | { |
|  | G[i][j] = 0; |
|  | } |
|  | else { |
|  | G[i][j] = 1; |
|  | } |
|  | } |
|  | k++; |
|  | } |
|  | k = 1; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = k; j < n; j++) |
|  | { |
|  | G[j][i] = G[i][j]; |
|  | } |
|  | k++; |
|  | } |
|  |  |
|  | for (int i = 0; i < n; i++) |
|  | cout << "V" << i << " "; |
|  |  |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << "\n"; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | cout << G[i][j] << " "; |
|  | } |
|  | } |
|  | return G; |
|  | } |
|  |  |
|  | int\*\* create\_G3(int n) |
|  | { |
|  | int\*\* G, k = 1; |
|  | G = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | srand(time(NULL)); |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | G[i] = (int\*)malloc(n \* sizeof(int)); |
|  | G[i][i] = 0; |
|  | for (int j = k; j < n; j++) |
|  | { |
|  | G[i][j] = 0; |
|  | } |
|  | k++; |
|  | } |
|  | k = 1; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = k; j < n; j++) |
|  | { |
|  | G[j][i] = G[i][j]; |
|  | } |
|  | k++; |
|  | } |
|  |  |
|  | /\*for (int i = 0; i < n; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << "\n"; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | cout << G[i][j] << " "; |
|  | } |
|  | }\*/ |
|  | return G; |
|  | } |
|  |  |
|  | int\*\* create\_G4(int u) |
|  | { |
|  | int\*\* G, k = 1; |
|  | G = (int\*\*)malloc(u \* sizeof(int\*)); |
|  | srand(time(NULL)); |
|  | for (int i = 0; i < u; i++) |
|  | { |
|  | G[i] = (int\*)malloc(u \* sizeof(int)); |
|  | G[i][i] = 0; |
|  | for (int j = k; j < u; j++) |
|  | { |
|  | G[i][j] = 0; |
|  | } |
|  | k++; |
|  | } |
|  | k = 1; |
|  | for (int i = 0; i < u; i++) |
|  | { |
|  | for (int j = k; j < u; j++) |
|  | { |
|  | G[j][i] = G[i][j]; |
|  | } |
|  | k++; |
|  | } |
|  |  |
|  | /\*for (int i = 0; i < n; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << "\n"; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | cout << G[i][j] << " "; |
|  | } |
|  | }\*/ |
|  | return G; |
|  | } |
|  |  |
|  | typedef struct List { |
|  | int Data; |
|  | List\* next; |
|  | }List; |
|  |  |
|  | List\* init(int num) |
|  | { |
|  | List\* node = (List\*)malloc(sizeof(List)); |
|  | node->Data = num; |
|  | node->next = NULL; |
|  | return node; |
|  | } |
|  |  |
|  | List\* add(List\* lst, int num) |
|  | { |
|  | List\* temp, \* p; |
|  | temp = (List\*)malloc(sizeof(List)); |
|  | p = lst->next; |
|  | lst->next = temp; |
|  | temp->Data = num; |
|  | temp->next = p; |
|  | return temp; |
|  | } |
|  |  |
|  | void initialize(int n, List\*\* vrt) |
|  | { |
|  | int i; |
|  | for (i = 0; i < n; i++) |
|  | { |
|  | vrt[i] = init(i); |
|  | } |
|  | } |
|  |  |
|  | void list(int n, int i, int\*\* G1, List\* lst) |
|  | { |
|  | int j; |
|  | for (j = n; j >= 0; j--) |
|  | { |
|  | if (G1[i][j] == 1) |
|  | add(lst, j); |
|  | } |
|  | } |
|  |  |
|  | void result(List\*\* vrt, int\*\* G1, int n) |
|  | { |
|  | int i; |
|  | for (i = 0; i < n; i++) |
|  | { |
|  | list(n, i, G1, vrt[i]); |
|  | } |
|  | } |
|  |  |
|  | void print(List\*\* vrt, int n) |
|  | { |
|  | int i, j; |
|  | for (i = 0; i < n; i++) |
|  | { |
|  | cout << "\n"; |
|  | List\* x = vrt[i]; |
|  | for (j = 0; j < n; j++) |
|  | { |
|  | if (vrt[i] != NULL) |
|  | { |
|  | cout << vrt[i]->Data << " "; |
|  | vrt[i] = vrt[i]->next; |
|  | } |
|  | } |
|  | vrt[i] = x; |
|  | } |
|  | cout << "\n"; |
|  | } |
|  |  |
|  | void list\_zam(List\*\* vrt, int n) { |
|  | int v1, v2; |
|  | List\*\* list1 = (List\*\*)malloc(n \* sizeof(List\*)); |
|  | List\* h, \* e, \*temp\_vrt; |
|  | initialize(n, list1); |
|  | int l = 0; |
|  | cout << "which vert to close: "; |
|  | cout << endl; |
|  | cin >> v1 >> v2; |
|  |  |
|  | for (int i = 0; i < n; i++) { |
|  | temp\_vrt = vrt[i]; |
|  | h = list1[i]; |
|  | list1[i]->Data = temp\_vrt->Data; |
|  | list1[i]->next = NULL; |
|  | temp\_vrt = temp\_vrt->next; |
|  | while (temp\_vrt != NULL) { |
|  | list1[i]->next = (List\*)malloc(sizeof(List)); |
|  | list1[i] = list1[i]->next; |
|  | list1[i]->Data = temp\_vrt->Data; |
|  | list1[i]->next = NULL; |
|  | temp\_vrt = temp\_vrt->next; |
|  | } |
|  | if (i == v1) { |
|  | e = list1[i]; |
|  | } |
|  | list1[i] = h; |
|  | } |
|  | List\* temp, \* temp1, \* end; |
|  | temp = list1[v1]; |
|  | temp1 = list1[v2]; |
|  | end = list1[v1]; |
|  | while (end != NULL && end->next != NULL) { |
|  | end = end->next; |
|  | } |
|  | while (temp1 != NULL) { |
|  | while (temp != NULL) { |
|  | if (temp1->Data == temp->Data) { |
|  | l++; |
|  | } |
|  |  |
|  | temp = temp->next; |
|  | } |
|  | temp = list1[v1]; |
|  | if (l == 0) { |
|  | end->next = (List\*)malloc(sizeof(List)); |
|  | end = end->next; |
|  | end->Data = temp1->Data; |
|  | end->next = NULL; |
|  | } |
|  | temp1 = temp1->next; |
|  | l = 0; |
|  | } |
|  | for (int i = v2; i < n - 1; i++) { |
|  | list1[i] = list1[i + 1]; |
|  | } |
|  | print(list1, n); |
|  | } |
|  |  |
|  | void list\_stg(List\*\* vrt, int n) { |
|  | int v1, v2; |
|  | List\*\* list2 = (List\*\*)malloc(n \* sizeof(List\*)); |
|  | List\* h, \* e, \*temp\_vrt; |
|  | initialize(n, list2); |
|  | int l = 0; |
|  | cout << "which vert to pull of: "; |
|  | cout << endl; |
|  | cin >> v1 >> v2; |
|  | for (int i = 0; i < n; i++) { |
|  | temp\_vrt = vrt[i]; |
|  | h = list2[i]; |
|  | list2[i]->Data = temp\_vrt->Data; |
|  | list2[i]->next = NULL; |
|  | temp\_vrt = temp\_vrt->next; |
|  | while (temp\_vrt != NULL) { |
|  | list2[i]->next = (List\*)malloc(sizeof(List)); |
|  | list2[i] = list2[i]->next; |
|  | list2[i]->Data = temp\_vrt->Data; |
|  | list2[i]->next = NULL; |
|  | temp\_vrt = temp\_vrt->next; |
|  | } |
|  | if (i == v1) { |
|  | e = list2[i]; |
|  | } |
|  | list2[i] = h; |
|  | } |
|  | List\* temp, \* temp1, \*end; |
|  | temp = list2[v1]; |
|  | temp1 = list2[v2]; |
|  | end = list2[v1]; |
|  | while (end != NULL && end->next != NULL) { |
|  | end = end->next; |
|  | } |
|  | while (temp1 != NULL) { |
|  | while (temp != NULL) { |
|  | if (temp1->Data == temp->Data) { |
|  | l++; |
|  | } |
|  |  |
|  | temp = temp->next; |
|  | } |
|  | temp = list2[v1]; |
|  | if (l == 0) { |
|  | end->next = (List\*)malloc(sizeof(List)); |
|  | end = end->next; |
|  | end->Data = temp1->Data; |
|  | end->next = NULL; |
|  | } |
|  | temp1 = temp1->next; |
|  | l = 0; |
|  | } |
|  | for (int i = v2; i < n - 1; i++) { |
|  | list2[i] = list2[i + 1]; |
|  | } |
|  | print(list2, n); |
|  | } |
|  |  |
|  | void list\_ras(List\*\* vrt, List\*\* vrt2, int n) { |
|  | int v1, v2; |
|  | int l, s = 0; |
|  | s = n + 1; |
|  | List\*\* list3 = (List\*\*)malloc(s \* sizeof(List\*)); |
|  | List\* h, \* e, \* h1, \* e1, \*temp\_vrt; |
|  |  |
|  |  |
|  | initialize(s, list3); |
|  | cout << "which vert to split: "; |
|  | cout << endl; |
|  | cin >> v1; |
|  |  |
|  | for (int i = 0; i < n; i++) { |
|  | temp\_vrt = vrt[i]; |
|  | h = list3[i]; |
|  | list3[i]->Data = temp\_vrt->Data; |
|  | list3[i]->next = NULL; |
|  | temp\_vrt = temp\_vrt->next; |
|  | while (temp\_vrt != NULL) { |
|  | list3[i]->next = (List\*)malloc(sizeof(List)); |
|  | list3[i] = list3[i]->next; |
|  | list3[i]->Data = temp\_vrt->Data; |
|  | list3[i]->next = NULL; |
|  | temp\_vrt = temp\_vrt->next; |
|  | } |
|  | if (i == v1) { |
|  | e = list3[i]; |
|  | } |
|  | list3[i] = h; |
|  | } |
|  |  |
|  | List\* temp, \* temp1, \* temp2; |
|  | temp = list3[v1]; |
|  | temp1 = list3[n]; |
|  | //c[v1] = vrt[v1]; |
|  | temp1->next = (List\*)malloc(sizeof(List)); |
|  | temp1 = temp1->next; |
|  | temp1->next = NULL; |
|  | if (temp != NULL) |
|  | { |
|  | temp1->Data = temp->Data; |
|  | temp = temp->next; |
|  | temp1->next = NULL; |
|  | } |
|  | while (temp) |
|  | { |
|  | temp1->next = (List\*)malloc(sizeof(List)); |
|  | temp1 = temp1->next; |
|  | temp1->Data = temp->Data; |
|  | temp1->next = NULL; |
|  | temp = temp->next; |
|  | } |
|  | //c[v1] = list3[v1]; |
|  | //a[v1] = c[v1]; |
|  |  |
|  | //while (temp1){ |
|  | //temp1 = temp1->next; |
|  | l = 0; |
|  | for (int j = 0; j < n; j++) { |
|  | temp2 = list3[j]; |
|  | while (temp2->next != NULL) { |
|  | if (temp2->Data == v1) { |
|  | l++; |
|  | } |
|  | temp2 = temp2->next; |
|  |  |
|  | } |
|  |  |
|  | if (l > 0) { |
|  | temp2->next = new List; |
|  | temp2 = temp2->next; |
|  | temp2->Data = n; |
|  | temp2->next = NULL; |
|  | } |
|  |  |
|  | l = 0; |
|  | if (temp2 != NULL) { |
|  | if (temp2->next == NULL) { |
|  | if (temp2->Data == v1) { |
|  | l++; |
|  | } |
|  | if (l > 0) { |
|  | temp2->next = new List; |
|  | temp2 = temp2->next; |
|  | temp2->Data = n; |
|  | temp2->next = NULL; |
|  | } |
|  | } |
|  | } |
|  | l = 0; |
|  | } |
|  | //} |
|  | //List \*temp, \*temp1; |
|  | //temp = list3[v1]; |
|  | //temp1 = list3[v2]; |
|  |  |
|  | /\*for (int i = n; i < n + 1; i++){ |
|  | list3[i + 1]->Data = c[v1]->Data; |
|  | }\*/ |
|  | print(list3, s); |
|  | cout << endl; |
|  | } |
|  |  |
|  | void ob(int\*\* G1, int\*\* G2, int n, int n1) |
|  | { |
|  | int\*\* M1 = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | int\*\* M2 = (int\*\*)malloc(n1 \* sizeof(int\*)); |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | M1[i] = (int\*)malloc(n \* sizeof(int)); |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = G1[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | M2[i] = (int\*)malloc(n1 \* sizeof(int)); |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = G2[i][j]; |
|  | } |
|  | } |
|  | if (n < n1) |
|  | { |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | if (M1[i][j] != M2[i][j]) |
|  | { |
|  | M2[i][j] = M1[i][j] + M2[i][j]; |
|  | } |
|  | } |
|  | } |
|  | cout << "ob result: " << endl; |
|  | for (int i = 0; i < n1; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n1; j++) |
|  | cout << M2[i][j] << " "; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | if (M1[i][j] != M2[i][j]) |
|  | { |
|  | M1[i][j] = M1[i][j] + M2[i][j]; |
|  | } |
|  | } |
|  | } |
|  | cout << endl << "ob result: " << endl; |
|  | for (int i = 0; i < n; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n; j++) |
|  | cout << M1[i][j] << " "; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void per(int\*\* G1, int\*\* G2, int\*\* G3, int n, int n1) |
|  | { |
|  | int\*\* M1 = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | int\*\* M2 = (int\*\*)malloc(n1 \* sizeof(int\*)); |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | M1[i] = (int\*)malloc(n \* sizeof(int)); |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = G1[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | M2[i] = (int\*)malloc(n1 \* sizeof(int)); |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = G2[i][j]; |
|  | } |
|  | } |
|  | if (n < n1) |
|  | { |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | if (M1[i][j] == 1 && M2[i][j] == 1) |
|  | { |
|  | G3[i][j] = 1; |
|  | } |
|  | } |
|  | } |
|  | cout << endl << "per result: " << endl; |
|  | for (int i = 0; i < n1; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n1; j++) |
|  | cout << G3[i][j] << " "; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | if (M1[i][j] == 1 && M2[i][j] == 1) |
|  | { |
|  | G3[i][j] = 1; |
|  | } |
|  | } |
|  | } |
|  | cout << endl << "per result: " << endl; |
|  | for (int i = 0; i < n; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n; j++) |
|  | cout << G3[i][j] << " "; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void circleplus(int\*\* G1, int\*\* G2, int n, int n1) |
|  | { |
|  | int x = 0; |
|  | int\*\* M1 = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | int\*\* M2 = (int\*\*)malloc(n1 \* sizeof(int\*)); |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | M1[i] = (int\*)malloc(n \* sizeof(int)); |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = G1[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | M2[i] = (int\*)malloc(n1 \* sizeof(int)); |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = G2[i][j]; |
|  | } |
|  | } |
|  | //////////////////////////////// |
|  | if (n < n1) |
|  | { |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | if (M1[i][j] != M2[i][j]) |
|  | { |
|  | M2[i][j] = 1; |
|  | } |
|  | else |
|  | { |
|  | M2[i][j] = 0; |
|  | } |
|  | } |
|  | } |
|  | /\*for (int i = 0; i < n1; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n1; j++) |
|  | cout << M2[i][j] << " "; |
|  | }\*/ |
|  | ///////////////////////////////// |
|  | for (int p = 0; p < n1; p++) |
|  | { |
|  | for (int k = 0; k < n1; k++) |
|  | { |
|  | if (M2[p][k] == 1) |
|  | { |
|  | x++; |
|  | } |
|  | } |
|  | if (x == 0) |
|  | { |
|  | for (int i = p; i < n1 - 1; i++) |
|  | { |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = M2[i + 1][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | for (int j = p; j < n1 - 1; j++) |
|  | { |
|  | M2[i][j] = M2[i][j + 1]; |
|  | } |
|  | } |
|  | n1--; |
|  | p--; |
|  | } |
|  | x = 0; |
|  | } |
|  | cout << endl << "circleplus result: " << endl; |
|  | for (int i = 0; i < n1; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | cout << M2[i][j] << " "; |
|  | } |
|  | } |
|  | } |
|  | /////////////////////////////////// |
|  | else |
|  | { |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | if (M1[i][j] != M2[i][j]) |
|  | { |
|  | M1[i][j] = 1; |
|  | } |
|  | else |
|  | { |
|  | M1[i][j] = 0; |
|  | } |
|  | } |
|  | } |
|  | /\*for (int i = 0; i < n; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n; j++) |
|  | cout << M1[i][j] << " "; |
|  | }\*/ |
|  | //////////////////////////////// |
|  | for (int p = 0; p < n; p++) |
|  | { |
|  | for (int k = 0; k < n; k++) |
|  | { |
|  | if (M1[p][k] == 1) |
|  | { |
|  | x++; |
|  | } |
|  | } |
|  | if (x == 0) |
|  | { |
|  | for (int i = p; i < n - 1; i++) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = M1[i + 1][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = p; j < n - 1; j++) |
|  | { |
|  | M1[i][j] = M1[i][j + 1]; |
|  | } |
|  | } |
|  | n--; |
|  | p--; |
|  | } |
|  | x = 0; |
|  | } |
|  | cout << endl << "circleplus result: " << endl; |
|  | for (int i = 0; i < n; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | cout << M1[i][j] << " "; |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void zam(int\*\* G1, int\*\* G2, int n, int n1) |
|  | { |
|  | int x = 0; |
|  | int v1, v2, w = 0; |
|  | int\*\* M1 = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | int\*\* M2 = (int\*\*)malloc(n1 \* sizeof(int\*)); |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | M1[i] = (int\*)malloc(n \* sizeof(int)); |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = G1[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | M2[i] = (int\*)malloc(n1 \* sizeof(int)); |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = G2[i][j]; |
|  | } |
|  | } |
|  | cout << "which graph to close: "; |
|  | cin >> w; |
|  | if (w == 1) { |
|  | cout << "which vert to close: "; |
|  | cout << endl; |
|  | cin >> v1 >> v2; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | if (M1[v1][j] != M1[v2][j]) |
|  | { |
|  | M1[v1][j] = 1; |
|  | M1[j][v1] = M1[v1][j]; |
|  | } |
|  | M1[v2][j] = 0; |
|  | M1[j][v2] = 0; |
|  | } |
|  | if (M1[v1][v1] == 0) |
|  | M1[v1][v1] = 1; |
|  | /////////////////////////////////////////// |
|  | for (int p = 0; p < n; p++) |
|  | { |
|  | for (int k = 0; k < n; k++) |
|  | { |
|  | if (M1[p][k] == 1) |
|  | { |
|  | x++; |
|  | } |
|  | } |
|  | if (x == 0) |
|  | { |
|  | for (int i = p; i < n - 1; i++) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = M1[i + 1][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = p; j < n - 1; j++) |
|  | { |
|  | M1[i][j] = M1[i][j + 1]; |
|  | } |
|  | } |
|  | n--; |
|  | p--; |
|  | } |
|  | x = 0; |
|  | } |
|  | cout << endl << "Result:"; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | cout << M1[i][j] << " "; |
|  | } |
|  | } |
|  | } |
|  | if (w == 2) { |
|  | cout << "which vert to close: "; |
|  | cout << endl; |
|  | cin >> v1 >> v2; |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | if (M2[v1][j] != M2[v2][j]) |
|  | { |
|  | M2[v1][j] = 1; |
|  | M2[j][v1] = M2[v1][j]; |
|  | } |
|  | M2[v2][j] = 0; |
|  | M2[j][v2] = 0; |
|  | } |
|  | if (M2[v1][v1] == 0) |
|  | M2[v1][v1] = 1; |
|  | /////////////////////////////////////////// |
|  | for (int p = 0; p < n1; p++) |
|  | { |
|  | for (int k = 0; k < n1; k++) |
|  | { |
|  | if (M2[p][k] == 1) |
|  | { |
|  | x++; |
|  | } |
|  | } |
|  | if (x == 0) |
|  | { |
|  | for (int i = p; i < n1 - 1; i++) |
|  | { |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = M2[i + 1][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | for (int j = p; j < n1 - 1; j++) |
|  | { |
|  | M2[i][j] = M2[i][j + 1]; |
|  | } |
|  | } |
|  | n1--; |
|  | p--; |
|  | } |
|  | x = 0; |
|  | } |
|  | cout << endl << "Result:"; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | cout << M2[i][j] << " "; |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void stg(int\*\* G1, int\*\* G2, int n, int n1) |
|  | { |
|  | int x = 0; |
|  | int v1, v2, w = 0; |
|  | int\*\* M1 = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | int\*\* M2 = (int\*\*)malloc(n1 \* sizeof(int\*)); |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | M1[i] = (int\*)malloc(n \* sizeof(int)); |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = G1[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | M2[i] = (int\*)malloc(n1 \* sizeof(int)); |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = G2[i][j]; |
|  | } |
|  | } |
|  | cout << "which graph to pull of: "; |
|  | cin >> w; |
|  | if (w == 1) { |
|  | cout << "which vert to pull of: "; |
|  | cout << endl; |
|  | cin >> v1 >> v2; |
|  | if (M1[v1][v2] == M1[v2][v1] && M1[v1][v2] != 0) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | if (M1[v1][j] != M1[v2][j]) |
|  | { |
|  | M1[v1][j] = 1; |
|  | M1[j][v1] = M1[v1][j]; |
|  | } |
|  | M1[v2][j] = 0; |
|  | M1[j][v2] = 0; |
|  | } |
|  | if (M1[v1][v1] == 1) |
|  | M1[v1][v1] = 0; |
|  | /////////////////////////////////////////// |
|  | for (int p = 0; p < n; p++) |
|  | { |
|  | for (int k = 0; k < n; k++) |
|  | { |
|  | if (M1[p][k] == 1) |
|  | { |
|  | x++; |
|  | } |
|  | } |
|  | if (x == 0) |
|  | { |
|  | for (int i = p; i < n - 1; i++) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = M1[i + 1][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = p; j < n - 1; j++) |
|  | { |
|  | M1[i][j] = M1[i][j + 1]; |
|  | } |
|  | } |
|  | n--; |
|  | p--; |
|  | } |
|  | x = 0; |
|  | } |
|  | cout << endl << "Result:"; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | cout << M1[i][j] << " "; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | cout << "NO"; |
|  | } |
|  | } |
|  | if (w == 2) { |
|  | cout << "which vert to close: "; |
|  | cout << endl; |
|  | cin >> v1 >> v2; |
|  | if (M2[v1][v2] == M2[v2][v1] && M2[v1][v2] != 0) |
|  | { |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | if (M2[v1][j] != M2[v2][j]) |
|  | { |
|  | M2[v1][j] = 1; |
|  | M2[j][v1] = M2[v1][j]; |
|  | } |
|  | M2[v2][j] = 0; |
|  | M2[j][v2] = 0; |
|  | } |
|  | if (M2[v1][v1] == 1) |
|  | M2[v1][v1] = 0; |
|  | /////////////////////////////////////////// |
|  | for (int p = 0; p < n1; p++) |
|  | { |
|  | for (int k = 0; k < n1; k++) |
|  | { |
|  | if (M2[p][k] == 1) |
|  | { |
|  | x++; |
|  | } |
|  | } |
|  | if (x == 0) |
|  | { |
|  | for (int i = p; i < n1 - 1; i++) |
|  | { |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = M2[i + 1][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | for (int j = p; j < n1 - 1; j++) |
|  | { |
|  | M2[i][j] = M2[i][j + 1]; |
|  | } |
|  | } |
|  | n1--; |
|  | p--; |
|  | } |
|  | x = 0; |
|  | } |
|  | cout << endl << "Result:"; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | cout << M2[i][j] << " "; |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | cout << "NO"; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void ras(int\*\* G1, int\*\* G2, int\*\* G3, int n, int n1) |
|  | { |
|  | int\*\* M1 = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | int\*\* M2 = (int\*\*)malloc(n1 \* sizeof(int\*)); |
|  | int v1, v2, w = 0; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | M1[i] = (int\*)malloc(n \* sizeof(int)); |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = G1[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | M2[i] = (int\*)malloc(n1 \* sizeof(int)); |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = G2[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | G3[i][j] = 0; |
|  | } |
|  | } |
|  | cout << "which graph to split: "; |
|  | cin >> w; |
|  | if (w == 1) { |
|  | cout << "which vert to split: "; |
|  | cout << endl; |
|  | cin >> v1; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | G3[i][j] = M1[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | G3[i][n] = M1[v1][i]; |
|  |  |
|  | } |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | G3[n][i] = M1[i][v1]; |
|  | } |
|  |  |
|  | } |
|  | if (w == 2) { |
|  | cout << "which vert to split: "; |
|  | cout << endl; |
|  | cin >> v1; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | G3[i][j] = M2[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | G3[i][n] = M2[v1][i]; |
|  | } |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | G3[n][i] = M2[i][v1]; |
|  | } |
|  | } |
|  | G3[n][v1] = 1; |
|  | G3[v1][n] = 1; |
|  | for (int i = 0; i < n1; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < n1; j++) |
|  | cout << G3[i][j] << " "; |
|  | } |
|  | } |
|  |  |
|  | void decm(int\*\* G1, int\*\* G2, int\*\* G4, int n, int n1) |
|  | { |
|  | int\*\* M1 = (int\*\*)malloc(n \* sizeof(int\*)); |
|  | int\*\* M2 = (int\*\*)malloc(n1 \* sizeof(int\*)); |
|  | int v1, v2, w, u, a = 0; |
|  | u = n \* n1; |
|  | int b = -1; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | M1[i] = (int\*)malloc(n \* sizeof(int)); |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | M1[i][j] = G1[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < n1; i++) |
|  | { |
|  | M2[i] = (int\*)malloc(n1 \* sizeof(int)); |
|  | for (int j = 0; j < n1; j++) |
|  | { |
|  | M2[i][j] = G2[i][j]; |
|  | } |
|  | } |
|  |  |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | for (int k = 0; k < n1; k++) |
|  | { |
|  | a = 0; |
|  | b++; |
|  | for (int j = 0; j < n; j++) |
|  | { |
|  | for (int p = 0; p < n1; p++) |
|  | { |
|  | if (i == p && j == k) |
|  | G4[b][a] = 0; |
|  | else |
|  | if (i == j) |
|  | G4[b][a] = M2[k][p]; |
|  | else |
|  | if (k == p) |
|  | G4[b][a] = M1[i][j]; |
|  | else |
|  | G4[b][a] = 0; |
|  | a++; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | cout << endl << "decm result:" << endl; |
|  | for (int i = 0; i < u; i++) |
|  | cout << "V" << i << " "; |
|  | for (int i = 0; i < u; i++) |
|  | { |
|  | cout << endl; |
|  | for (int j = 0; j < u; j++) |
|  | cout << G4[i][j] << " "; |
|  | } |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | int n, n1, \*\* t, \*\* t1, \*\* t2, \*\* t3, \*\* t4, s, o, u, w; |
|  |  |
|  | cout << "Enter array 1 size: "; |
|  | cin >> n; |
|  |  |
|  | t = create(n); |
|  | cout << endl; |
|  | cout << endl; |
|  | cout << "Enter array 2 size: "; |
|  | cin >> n1; |
|  |  |
|  | u = n \* n1; |
|  | t1 = create(n1); |
|  | if (n < n1) |
|  | { |
|  | t2 = create\_G3(n1); |
|  | } |
|  | else |
|  | { |
|  | t2 = create\_G3(n); |
|  | } |
|  | cout << endl; |
|  | List\*\* vrt = (List\*\*)malloc(n \* sizeof(List\*)); |
|  | initialize(n, vrt); |
|  | result(vrt, t, n); |
|  | cout << endl; |
|  | cout << "first graph in list"; |
|  | print(vrt, n); |
|  | List\*\* vrt1 = (List\*\*)malloc(n1 \* sizeof(List\*)); |
|  | s = n + 1; |
|  | List\*\* vrt2 = (List\*\*)malloc(s \* sizeof(List\*)); |
|  | initialize(n, vrt2); |
|  | result(vrt2, t, n); |
|  | initialize(n1, vrt1); |
|  | result(vrt1, t1, n1); |
|  | cout << endl; |
|  | cout << "second graph in list"; |
|  | print(vrt1, n1); |
|  | cout << endl; |
|  | /\*for (int i = 0; i < n + 1; i++){ |
|  | vrt[i] = 0; |
|  | }\*/ |
|  | cout << "which graph to close: "; |
|  | cin >> w; |
|  | if (w == 1) { |
|  | cout << endl; |
|  | list\_zam(vrt, n); |
|  | cout << endl; |
|  | } |
|  | if (w == 2) { |
|  | cout << endl; |
|  | list\_zam(vrt1, n1); |
|  | cout << endl; |
|  | } |
|  | cout << endl; |
|  | cout << "which graph to pull of: "; |
|  | cin >> w; |
|  | if (w == 1) { |
|  | cout << endl; |
|  | list\_stg(vrt, n); |
|  | cout << endl; |
|  | } |
|  | if (w == 2) { |
|  | cout << endl; |
|  | list\_stg(vrt1, n1); |
|  | cout << endl; |
|  | } |
|  | cout << "which graph to split: "; |
|  | cin >> w; |
|  | /\*for (int i = 0; i < n + 1; i++){ |
|  | vrt[i] = 0; |
|  | }\*/ |
|  | if (w == 1) { |
|  | cout << endl; |
|  | list\_ras(vrt, vrt2, n); |
|  | cout << endl; |
|  | } |
|  | if (w == 2) { |
|  | cout << endl; |
|  | list\_ras(vrt1, vrt2, n1); |
|  | cout << endl; |
|  | } |
|  | ob(t, t1, n, n1); |
|  | cout << endl; |
|  | cout << endl; |
|  | per(t, t1, t2, n, n1); |
|  | cout << endl; |
|  |  |
|  | cout << endl; |
|  | circleplus(t, t1, n, n1); |
|  | cout << endl; |
|  |  |
|  | cout << endl; |
|  | zam(t, t1, n, n1); |
|  | cout << endl; |
|  |  |
|  | cout << endl; |
|  | stg(t, t1, n, n1); |
|  | cout << endl; |
|  |  |
|  | if (n < n1) |
|  | { |
|  | t3 = create\_G3(n1 + 1); |
|  | } |
|  | else |
|  | { |
|  | t3 = create\_G3(n + 1); |
|  | } |
|  |  |
|  | cout << endl; |
|  | ras(t, t1, t3, n, n1); |
|  | cout << endl; |
|  |  |
|  | t4 = create\_G4(u); |
|  | cout << endl; |
|  | decm(t, t1, t4, n, n1); |
|  |  |
|  | for (int i = 0; i < n; i++) |
|  | free(t[i]); |
|  | free(t); |
|  | for (int i = 0; i < n1; i++) |
|  | free(t1[i]); |
|  | free(t1); |
|  | for (int i = 0; i < n; i++) |
|  | free(t2[i]); |
|  | free(t2); |
|  | //system("pause"); |
|  | return 0;  } |
|  |  |

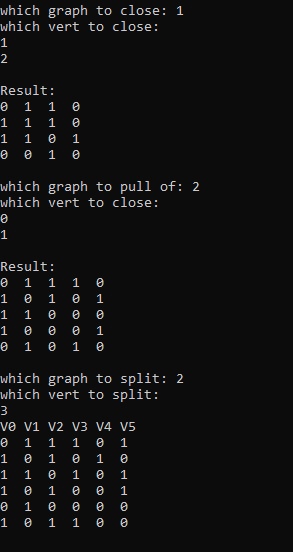
|  |  |
| --- | --- |
|  |  |

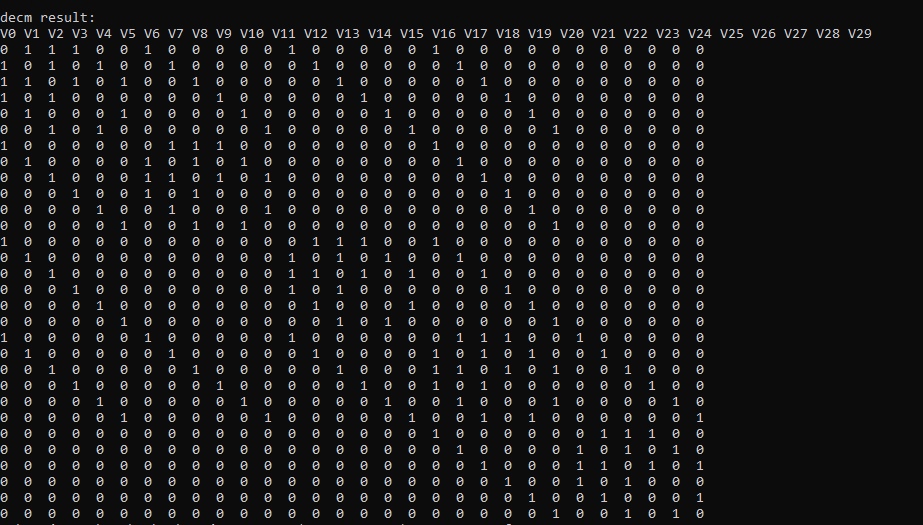
1. Результат работы программы.











Вывод: в ходе выполнения данной лабораторной работы разработали программу, согласно заданию.