**《数据结构》**

**课程设计报告**

|  |  |
| --- | --- |
| 课程名称： | 《数据结构》课程设计 |
| 课程设计题目： | 校园周游 |
| 姓 名： | 易子顺 |
| 院 系： | 计算机学院 |
| 专 业： | 计算机科学与技术 |
| 班 级： | 23052314 |
| 学 号： | 23050411 |
| 指导教师： | 葛瑞泉 |

2019年12月25日

1. **需求分析**

#### 功能需求：

设计并实现一个校园导游程序，为来访的客人提供各种信息查询服务

#### 界面需求：

（1）设计校园平面图，其中所含景点不少于10个。以图中顶点表示校内各景点，存放景点名称、代号、简介等信息：以边表示路径，存放路径长度等相关信息；

（2）为来访客人提供图中任意景点相关信息的查询；

（3）为来访客人提供途中任意景点的问路查询，即查询任意两个景点之间的一条最短的简单路径。

接口设计

extern Location locations[];

int vertexNum();

extern Path paths[];

int edgeNum();

AdjGraph \*adj\_graph\_init();

Path \*dijkstra(int start, int end, AdjGraph \*graph, int \*pathlen, int \*pathsz);

#### 数据结构设计

typedef struct {

  int no;

  char \*name;

  char \*description;

} Location;

typedef struct {

  int a;

  int b;

  int len;

} Path;

typedef struct AdjListNode {

  int dest;

  int weight;

  struct AdjListNode \*next;

} AdjListNode;

typedef struct {

  int adj\_v;

  AdjListNode \*head;

} AdjList;

typedef struct {

  int V;

  AdjList \*array;

} AdjGraph;

**详细设计**

**#include <campus.h>**

**Location locations[] = {**

**{0, "Gate", "Main gate in HDU"},          // Gate -> 0**

**{1, "Library", "Library in HDU"},         // Library -> 1**

**{2, "Sports", "Sports in HDU"},           // Sports -> 2**

**{3, "Science", "Science in HDU"},         // Science -> 3**

**{4, "Auditorium", "Auditorium in HDU"},   // Auditorium -> 4**

**{5, "Canteen", "Canteen in HDU"},         // Canteen -> 5**

**{6, "Cafeteria", "Cafeteria in HDU"},     // Cafeteria -> 6**

**{7, "Dormitory", "Dormitory in HDU"},     // Dormitory -> 7**

**{8, "Sports2", "Sports2 in HDU"},         // Sports2 -> 8**

**{9, "Science2", "Science2 in HDU"},       // Science2 -> 9**

**};**

**// 修改后的路径数组，从0开始**

**Path paths[] = {**

**{0, 1, 200},  // Gate -> Library**

**{0, 2, 150},  // Gate -> Sports**

**{1, 3, 300},  // Library -> Science**

**{1, 4, 400},  // Library -> Auditorium**

**{2, 5, 250},  // Sports -> Canteen**

**{2, 6, 350},  // Sports -> Cafeteria**

**{3, 7, 200},  // Science -> Dormitory**

**{4, 8, 500},  // Auditorium -> Sports2**

**{5, 9, 300},  // Canteen -> Science2**

**{6, 7, 150},  // Cafeteria -> Dormitory**

**{7, 8, 400},  // Dormitory -> Sports2**

**{8, 9, 250},  // Sports2 -> Science2**

**};**

**int edgeNum()**

**{**

**return sizeof(paths)/sizeof(paths[0]);**

**}**

**int vertexNum()**

**{**

**return sizeof(locations)/sizeof(locations[0]);**

**}**

**#include <campus.h>**

**/\***

**设计并实现一个校园导游程序，为来访的客人提供各种信息查询服务，具体包括：**

**（1）设计校园平面图，其中所含景点不少于10个。以图中顶点表示校内各景点，存放景点名称、代号、简介等信息：以边表示路径，存放路径长度等相关信息；**

**（2）为来访客人提供图中任意景点相关信息的查询；**

**（3）为来访客人提供途中任意景点的问路查询，即查询任意两个景点之间的一条最短的简单路径**

**\*/**

**void information()**

**{**

**printf("Please enter the name of the location you want to query:\n");**

**printf(">");**

**int num;**

**scanf("%d", &num);**

**printf("The information of %d is: name = %s\n", num, locations[num].name);**

**printf("Description:\n %s\n", locations[num].description);**

**}**

**int main()**

**{**

**int choice;**

**AdjGraph \*g = adj\_graph\_init();**

**printf("GRAPH: \n");**

**int vnum = vertexNum();**

**for (int i = 0; i < vnum; i++)**

**{**

**printf("no:%d %s (%s)\n", locations[i].no,locations[i].name, locations[i].description);**

**}**

**while (1){**

**printf("\nThis is a campus tour system.\n");**

**printf("Please select a function:\n");**

**printf("1. Query information of a location\n");**

**printf("2. Query the shortest path between two locations\n");**

**printf("3. Exit\n");**

**printf(">");**

**scanf("%d", &choice);**

**switch (choice)**

**{**

**case 1:**

**information();**

**break;**

**case 2:**

**printf("Please enter the number of the two locations you want to query:\n");**

**printf(">");**

**int a, b;**

**scanf("%d %d", &a, &b);**

**int pathlen = 0;**

**int pathsz = 0;**

**Path \*sp = dijkstra(a, b, g, &pathlen, &pathsz);**

**if(sp == NULL){**

**printf("No path found\n");**

**break;**

**}**

**printf("The shortest path between %s and %s is:\n", locations[a].name, locations[b].name);**

**for (int i = pathsz - 1; i >= 0; i--)**

**{**

**printf("%s -> %s distence = %d\n", locations[sp[i].a].name, locations[sp[i].b].name, sp[i].len);**

**}**

**printf("The total length of the path is %d\n", pathlen);**

**break;**

**case 3:**

**exit(0);**

**default:**

**printf("Invalid choice\n");**

**}**

**}**

**}**

**#include <campus.h>**

**#define INF 9999**

**typedef struct {**

**int value;**

**int parent;**

**int parent\_value;**

**int frozen;**

**} Node;**

**int extract\_min(AdjGraph \*graph, Node \*d);**

**Path \*dijkstra(int start, int end, AdjGraph \*graph, int \*pathlen, int \*pathsz)**

**{**

**Path \*sp;**

**Node \*d = malloc(sizeof(Node) \* graph->V);**

**//init d**

**d[start].value = 0;**

**d[start].parent = -1;**

**d[start].frozen = 0;**

**for(int i = 0; i < graph->V; i++){**

**if(i != start){**

**d[i].value = INF;**

**d[i].parent = -1;**

**d[i].frozen = 0;**

**}**

**}**

**//start!**

**int ing;**

**while((ing = extract\_min(graph, d)) != -1){**

**AdjListNode \*p = graph->array[ing].head;**

**while(p){**

**int v = p->dest;**

**int w = p->weight;**

**if(d[v].frozen == 0 && d[v].value > d[ing].value + w){**

**d[v].value = d[ing].value + w;**

**d[v].parent = ing;**

**d[v].parent\_value = w;**

**}**

**p = p->next;**

**}**

**d[ing].frozen = 1;**

**}**

**if(d[end].parent == -1){**

**#ifdef DEBUG**

**printf("[DEBUG]:No such Path\n");**

**#endif**

**return NULL;**

**}**

**sp = malloc(sizeof(Path));**

**int n = 1;**

**#ifdef DEBUG**

**printf("[DEBUG]:Path\n");**

**#endif**

**for(int i = end; i != start; i = d[i].parent){**

**sp[n - 1].a = d[i].parent;**

**sp[n - 1].b = i;**

**sp[n - 1].len = d[i].parent\_value;**

**n++;**

**sp = realloc(sp, sizeof(Path) \* n);**

**#ifdef DEBUG**

**printf("[DEBUG]:%d\n", i);**

**#endif**

**}**

**#ifdef DEBUG**

**printf("[DEBUG]:distence = %d\n", d[end].value);**

**printf("[DEBUG]:Path end\n");**

**#endif**

**\*pathlen = d[end].value;**

**\*pathsz = n - 1;**

**return sp;**

**}**

**int extract\_min(AdjGraph \*graph, Node \*d)**

**{**

**int min = INF;**

**int min\_index = -1;**

**for(int i = 0; i < graph->V; i++){**

**if(d[i].frozen == 0 && d[i].value < min){**

**min = d[i].value;**

**min\_index = i;**

**}**

**}**

**#ifdef DEBUG**

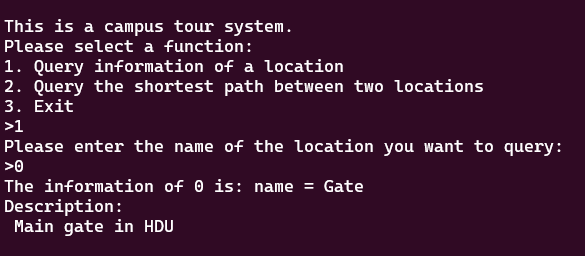
**printf("[DEBUG]:find min %d, index %d\n", min, min\_index);**

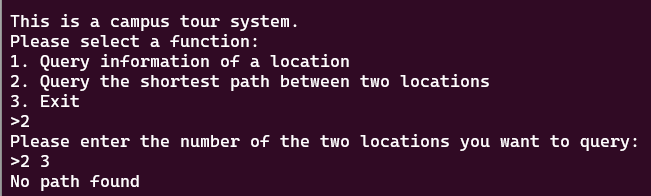
**#endif**

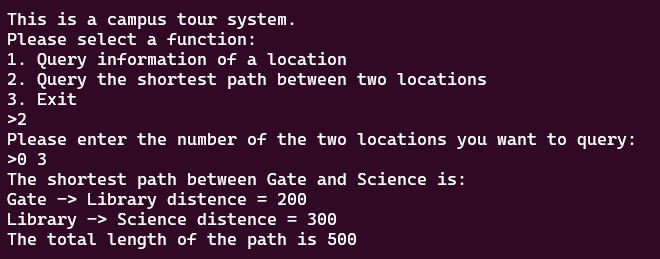
**return min\_index;**

**}**

1. **用户手册**







1. **附录**

源程序文件名清单：电梯（final）.cpp，电梯（final）.exe