校园导游咨询

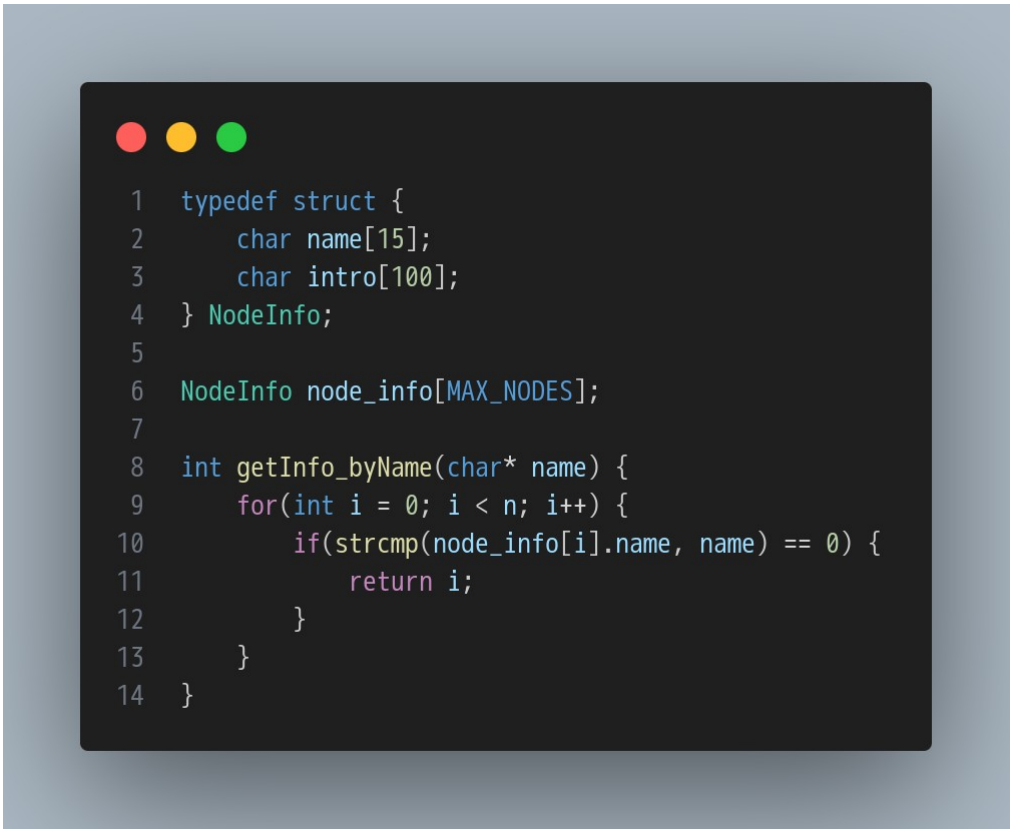
专业：计算机科学与技术班级：23052322 姓名：张逸轩学号：23051214

1. 实验目的

了解图的结构和图上的算法使用

2. 实验过程(实验方案、流程、程序等)(参考书上的格式需要写详细)

首先设计结构，使用一个 node_info 保存各个景点信息。



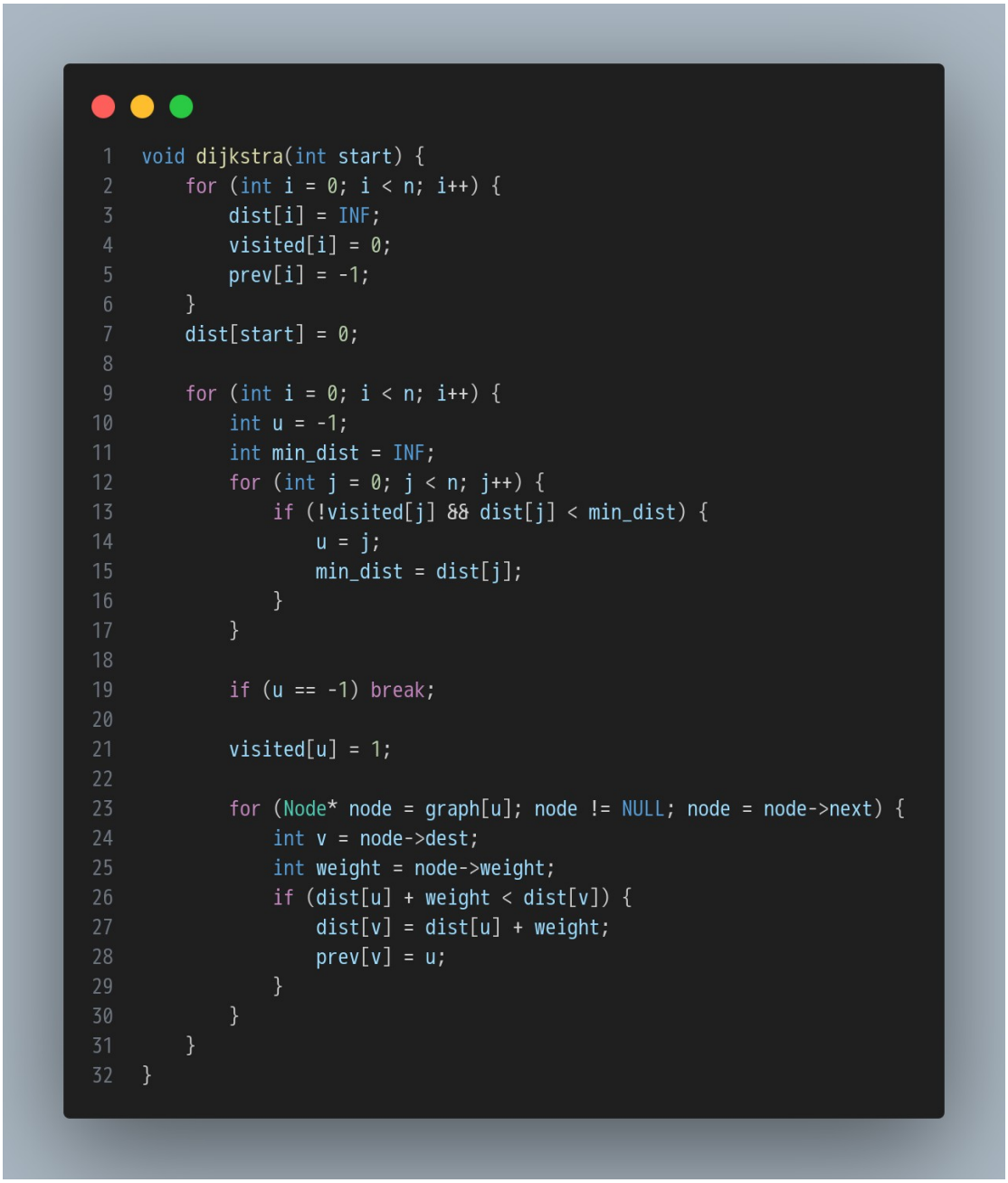
```
typedef struct {
    char name[15];
    char intro[100];
} NodeInfo;

NodeInfo node_info[MAX_NODES];

int getInfo_byName(char* name) {
    for(int i = 0; i < n; i++) {
        if(strcmp(node_info[i].name, name) == 0) {
            return i;
        }
    }
}
```

然后使用

邻接表建立图，建立每个边的信息。

```c
typedef struct Node {
    int dest;
    int weight;
    struct Node* next;
} Node;

Node* graph[MAX_NODES];
int dist[MAX_NODES];
int prev[MAX_NODES];
int visited[MAX_NODES];
int n;


void init_graph(int nodes) {
    n = nodes;
    for (int i = 0; i < n; i++) {
        graph[i] = NULL;
    }
}

void add_edge(int src, int dest, int weight) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->dest = dest;
    new_node->weight = weight;
    new_node->next = graph[src];
    graph[src] = new_node;

    // 因为是无向图，也需要加上反向边
    new_node = (Node*)malloc(sizeof(Node));
    new_node->dest = src;
    new_node->weight = weight;
    new_node->next = graph[dest];
    graph[dest] = new_node;
}
```

然后如果需要求得最短路径，则使用 Dijkstra 算法，在松弛阶段加入 prev 用于后续查找路径。



```c
void dijkstra(int start) {
    for (int i = 0; i < n; i++) {
        dist[i] = INF;
        visited[i] = 0;
        prev[i] = -1;
    }
    dist[start] = 0;

    for (int i = 0; i < n; i++) {
        int u = -1;
        int min_dist = INF;
        for (int j = 0; j < n; j++) {
            if (!visited[j] && dist[j] < min_dist) {
                u = j;
                min_dist = dist[j];
            }
        }

        if (u == -1) break;

        visited[u] = 1;

        for (Node* node = graph[u]; node != NULL; node = node->next) {
            int v = node->dest;
            int weight = node->weight;
            if (dist[u] + weight < dist[v]) {
                dist[v] = dist[u] + weight;
                prev[v] = u;
            }
        }
    }
}
```

当 Dijkstra 跑完后则利用 prev 倒序，输出完整最短路径。

```c
void print_path(int start, int end) {
    if (dist[end] == INF) {
        printf("没有 %s 到 %s 的路径\n", node_info[start].name, node_info[end].name);
        return;
    }

    printf(" %s 到 %s 的最短路径长度为 %d\n", node_info[start].name, node_info[end].name, dist[end]);
    int path[MAX_NODES];
    int path_length = 0;

    for (int v = end; v != -1; v = prev[v]) {
        path[path_length++] = v;
    }

    printf("路径为: \n");
    for (int i = path_length - 1; i >= 0; i--) {
        printf("%s ", node_info[path[i]].name);
        if (i > 0) printf("-> ");
    }
    printf("\n");
}
```

3. 实验结果及结果分析

```
输入景点数: 5
输入景点 0 的名字: a
输入景点 0 的简介: is a
输入景点 1 的名字: b
输入景点 1 的简介: is b
输入景点 2 的名字: c
输入景点 2 的简介: is c
输入景点 3 的名字: d
输入景点 3 的简介: is d
输入景点 4 的名字: e
输入景点 4 的简介: is e
输入边的数目: 6
输入边的信息（起点 终点 长度）:
a b 3
添加边 0 -> 1, 长度 3
b e 7
添加边 1 -> 4, 长度 7
a d 10
添加边 0 -> 3, 长度 10
a c 1
添加边 0 -> 2, 长度 1
c d 2
添加边 2 -> 3, 长度 2
d e 4
添加边 3 -> 4, 长度 4
请输入查询类型 (path/info/exit): info
请输入景点名字: a
景点名称: a
景点编号: 0
景点简介: is a


请输入查询类型 (path/info/exit): path
请输入起点和终点: a e
 a 到 e 的最短路径长度为 7
路径为:
a -> c -> d -> e
```

符合测试结果

4. 实验总结

通过这次实验，理解了图的本质，如何建立图，如何在图上求最短路径并且获取路径本身。