```
//
// Created by along on 17-11-26.
//

#include "Graph.h"
#include <iostream>
#include <fstream>
#include <ncurses.h>
#include <cstring>
#include <functional>
#include <sys/time.h>

#define KEY_ESC (27)

using namespace std;
int row, col;
const int Offset = 5;
const char title[] = "-----图管理器-----";
const char exitStr[] = " 右方向键确认 ESC 退出";

void readFromFile(Graph *&graph);
void menu(int choice, Graph *&graph);
void addEdge(Graph *&graph);
void delEdge(Graph *&graph);
void outDegree(Graph *&graph);
void printGraph(Graph *&graph);
void printDot(Graph *&graph);
void DFS(Graph *&graph);
void DFSR(Graph *&graph);
void BFS(Graph *&graph);

int main() {
    Graph *graph = nullptr;

    const vector<string> Choices({" 创建图 (邻接表)",
                                  " 创建图 (邻接矩阵)",
                                  " 创建图 (十字链表)",
                                  " 查看某个顶点的出 ⤸
                                  ę",
                                  " 查看图的所有信息",
                                  " 打印 Dot 图到文件",
                                  " 添加一条边",
                                  " 删除一条边",
                                  " 重置本图",
                                  " 先深遍历",
                                  " 先深遍历 (递归)",
                                  " 先广遍历"});

    int key, choice = 0;//用于指定光标位置
    setlocale(LC_ALL, "");
    initscr();
    keypad(stdscr, TRUE);
    noecho();
    cbreak();
    //界面
    do {
        getmaxyx(stdscr, row, col);
        clear();
        mvprintw(0, (int) (col - strlen(title)) / 2, "%s", ⤸
        title);
        for (int i = 0; i != Choices.size(); ++i) {
            mvprintw(i + 1, Offset, "%u.%s", i + 1, Choices[⤸
            i].c_str());
        }
        mvprintw(choice + 1, Offset - 2, "*");
        mvprintw(row - 2, (int) (col - strlen(exitStr)) / 2,
        "%s", exitStr);

        refresh();
        key = getch();

        switch (key) {
        case KEY_UP:
            if (--choice == -1)
                choice = (int) (Choices.size() - 1);
            break;
        case KEY_DOWN:
            if (++choice == Choices.size())
                choice = 0;
            break;
        case KEY_RIGHT:menu(choice, graph);
            break;
        default:break;
        }
    } while (key != KEY_ESC);
    endwin();
    return 0;
}

void addEdge(Graph *&graph) {
    if (graph == nullptr)
        return;
    mvprintw(row - 1, Offset,
    " 输入要添加的边的起点和终点 (使用',' 分隔)
    :");
    refresh();
    echo();
    unsigned long src, dst;
```

```cpp
    scanw("%ul,%ul", &src, &dst);
    graph->addEdge(src, dst);
}

void delEdge(Graph *&graph) {
    if (graph == nullptr)
        return;
    mvprintw(row - 1, Offset, ⤸
    " 输入要删除的边的起点和终点 (使用',' 分隔)⤸
    :");
    refresh();
    echo();
    unsigned long src, dst;
    scanw("%ul,%ul", &src, &dst);
    graph->delEdge(src, dst);
    noecho();
}

void readFromFile(Graph *&graph) {
    mvprintw(row - 1, Offset, ⤸
    " 输入图信息所在的文件名:");
    refresh();
    echo();
    char filename[50];
    getnstr(filename, sizeof(filename));
    ifstream Stream(filename);
    if (Stream) {
        graph->resetFromStream(Stream);
        Stream.close();
        noecho();
    } else {
        return;
    }
    noecho();
}

void outDegree(Graph *&graph) {
    if (graph == nullptr)
        return;
    mvprintw(row - 1, Offset, ⤸
    " 输入要查看出度的顶点:");
    refresh();
    echo();
    unsigned long src;
    scanw("%lu", &src);
    auto outDegree = graph->outDegree(src);
    mvprintw(row - 1, Offset, "");
    clrtobot();
    mvprintw(row - 1, Offset, " 顶点'%lu' 的出度是%lu", ⤸
    src, outDegree);
    refresh();
    noecho();
    getch();
}

void printGraph(Graph *&graph) {
    if (graph == nullptr)
        return;
    clear();
    function<bool(unsigned long, unsigned long)> func = [&](⤸
    unsigned long src, unsigned long dst) {
        printw(" %d ", dst);
        return true;
    };
    mvprintw(0, (int) (col - strlen(title)) / 2, "%s", title)⤸
    ;
```

```cpp
    mvprintw(1, Offset - 2, " 顶点 | 邻接点");
    for (unsigned long vex = 0; vex != graph->vexCount(); ++
    vex) {
        mvprintw((int) vex + 2, Offset, " %d |", vex);
        graph->foreach(vex, func);
    }
    move(-1, -1);
    refresh();
    getch();
}

void printDot(Graph *&graph) {
    if (graph == nullptr)
        return;
    mvprintw(row - 1, Offset, " 输入要写入的文件名:")
    ;
    refresh();
    echo();
    char filename[50];
    getnstr(filename, sizeof(filename));
    ofstream Stream(filename);
    if (Stream) {
        graph->printDot(Stream);
        Stream.close();
        noecho();
    } else {
        return;
    }
    noecho();
}

void menu(int choice, Graph *&graph) {
    if (choice > 2 && graph == nullptr) {
        mvprintw(row - 1, Offset, " 图未建立");
        refresh();
        getch();
        return;
    }
    switch (choice) {
    case 0:delete graph;
        graph = new GraphT(0);
        readFromFile(graph);
        break;
    case 1:delete graph;
        graph = new GraphM(0);
        readFromFile(graph);
        break;
    case 2:delete graph;
        graph = new GraphL(0);
        readFromFile(graph);
        break;
    case 3:outDegree(graph);
        break;
    case 4:printGraph(graph);
        break;
    case 5:printDot(graph);
        break;
    case 6:addEdge(graph);
        graph->reset();
        break;
    case 7:delEdge(graph);
        break;
    case 8:graph->reset();
        break;
    case 9:DFS(graph);
        break;
    case 10:DFSR(graph);
        break;
    case 11:BFS(graph);
        break;
    default:break;
    }
    if (graph->vexCount() == 0) {
        delete graph;
        graph = nullptr;
    }
}
void DFS(Graph *&graph) {
    if (graph == nullptr)
        return;
    function<void(unsigned long)> func = [&](unsigned long
↪
    dst) {
        printw("%d ", dst);
    };
    mvprintw(row - 1, Offset, " 先深遍历: ");

    //计时准备
    struct timeval tpstart{}, tpend{};
    double timeuse;
    gettimeofday(&tpstart, nullptr);
    //遍历
    graph->DFS(func);

    //计时结束
    gettimeofday(&tpend, nullptr);
    timeuse = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + ↪
    tpend.tv_usec - tpstart.tv_usec;↪
    //注意，秒的读数和微秒的读数都应计算在 ↪
    ĘĚ
    mvprintw(row - 2, Offset, " 用时:%lfus", timeuse);

    refresh();
    getch();
}
void DFSR(Graph *&graph) {
    if (graph == nullptr)
        return;
    function<void(unsigned long)> func = [&](unsigned long
⤸
    dst) {
        printw("%d ", dst);
    };
    mvprintw(row - 1, Offset, " 先深遍历: ");

    //计时准备
    struct timeval tpstart{}, tpend{};
    double timeuse;
    gettimeofday(&tpstart, nullptr);
    //遍历
    graph->DFSR(func);
    //计时结束
    gettimeofday(&tpend, nullptr);
    timeuse = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + ⤸
    tpend.tv_usec - tpstart.tv_usec;⤸
    //注意，秒的读数和微秒的读数都应计算在 ⤸
    ĘĚ
    mvprintw(row - 2, Offset, " 用时:%lfus", timeuse);

    refresh();
    getch();
}
```

```
void BFS(Graph *&graph) {
    if (graph == nullptr)
        return;
    function<void(unsigned long)> func = [&](unsigned long
⤸
    dst) {
        printw("%d ", dst);
    };
    mvprintw(row - 1, Offset, " 先深遍历: ");

    //计时准备
    struct timeval tpstart{}, tpend{};
    double timeuse;
    gettimeofday(&tpstart, nullptr);
    //遍历
    graph->BFS(func);
    //计时结束
    gettimeofday(&tpend, nullptr);
    timeuse = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + ⤸
    tpend.tv_usec - tpstart.tv_usec;⤸
    //注意, 秒的读数和微秒的读数都应计算在 ⤸
    ĘĚ
    mvprintw(row - 2, Offset, " 用时:%lfus", timeuse);

    refresh();
    getch();
}
```