```
01 //
02 // Created by along on 17-11-26.
03 //
04
05 #include "Graph.h"
06 #include <iostream>
07 #include <fstream>
08 #include <ncurses.h>
09 #include <cstring>
10 #include <functional>
11 #include <sys/time.h>
12
13 #define KEY_ESC (27)
14
15 using namespace std;
16 int row, col;
17 const int Offset = 5;
18 const char title[] = "-----图管理器-----";
19 const char exitStr[] = " 右方向键确认 ESC 退出";
20
21 void readFromFile(Graph *&graph);
22 void menu(int choice, Graph *&graph);
23 void addEdge(Graph *&graph);
24 void delEdge(Graph *&graph);
25 void outDegree(Graph *&graph);
26 void printGraph(Graph *&graph);
27 void printDot(Graph *&graph);
28 void DFS(Graph *&graph);
29 void DFSR(Graph *&graph);
30 void BFS(Graph *&graph);
31
32 int main() {
33     Graph *graph = nullptr;
```

```
34
35      const vector<string> Choices({" 创建图 (邻接表)",
36                                    " 创建图 (邻接矩阵)",
37                                    " 创建图 (十字链表)",
38                                    " 查看某个顶点的出 ⤸
39                                    ę",
40                                    " 查看图的所有信息",
41                                    " 打印 Dot 图到文件",
42                                    " 添加一条边",
43                                    " 删除一条边",
44                                    " 重置本图",
45                                    " 先深遍历",
46                                    " 先深遍历 (递归)",
47                                    " 先广遍历"});
48
49      int key, choice = 0;//用于指定光标位置
50      setlocale(LC_ALL, "");
51      initscr();
52      keypad(stdscr, TRUE);
53      noecho();
54      cbreak();
55      //界面
56      do {
57          getmaxyx(stdscr, row, col);
58          clear();
59          mvprintw(0, (int) (col - strlen(title)) / 2, "%s", ⤸
60          title);
61          for (int i = 0; i != Choices.size(); ++i) {
62              mvprintw(i + 1, Offset, "%u.%s", i + 1, Choices[⤸
63              i].c_str());
64          }
65          mvprintw(choice + 1, Offset - 2, "*");
66          mvprintw(row - 2, (int) (col - strlen(exitStr)) / 2,
```

```
)
67          "%s", exitStr);
68
69          refresh();
70          key = getch();
71
72          switch (key) {
73          case KEY_UP:
74              if (--choice == -1)
75                  choice = (int) (Choices.size() - 1);
76              break;
77          case KEY_DOWN:
78              if (++choice == Choices.size())
79                  choice = 0;
80              break;
81          case KEY_RIGHT:menu(choice, graph);
82              break;
83          default:break;
84          }
85      } while (key != KEY_ESC);
86      endwin();
87      return 0;
88  }
89
90  void addEdge(Graph *&graph) {
91      if (graph == nullptr)
92          return;
93      mvprintw(row - 1, Offset, )
94      " 输入要添加的边的起点和终点 (使用',' 分隔)
95      :");
96      refresh();
97      echo();
98      unsigned long src, dst;
```

```
99      scanw("%ul,%ul", &src, &dst);
100     graph->addEdge(src, dst);
101 }
102
103 void delEdge(Graph *&graph) {
104     if (graph == nullptr)
105         return;
106     mvprintw(row - 1, Offset, ↩
107     " 输入要删除的边的起点和终点（使用',' 分隔）↩
108     :");
109     refresh();
110     echo();
111     unsigned long src, dst;
112     scanw("%ul,%ul", &src, &dst);
113     graph->delEdge(src, dst);
114     noecho();
115 }
116
117 void readFromFile(Graph *&graph) {
118     mvprintw(row - 1, Offset, ↩
119     " 输入图信息所在的文件名:");
120     refresh();
121     echo();
122     char filename[50];
123     getnstr(filename, sizeof(filename));
124     ifstream Stream(filename);
125     if (Stream) {
126         graph->resetFromStream(Stream);
127         Stream.close();
128         noecho();
129     } else {
130         return;
131     }
```

```
132     noecho();
133 }
134
135 void outDegree(Graph *&graph) {
136     if (graph == nullptr)
137         return;
138     mvprintw(row - 1, Offset, ↩
139     " 输入要查看出度的顶点:");
140     refresh();
141     echo();
142     unsigned long src;
143     scanw("%lu", &src);
144     auto outDegree = graph->outDegree(src);
145     mvprintw(row - 1, Offset, "");
146     clrtobot();
147     mvprintw(row - 1, Offset, " 顶点'%lu' 的出度是%lu", ↩
148     src, outDegree);
149     refresh();
150     noecho();
151     getch();
152 }
153
154 void printGraph(Graph *&graph) {
155     if (graph == nullptr)
156         return;
157     clear();
158     function<bool(unsigned long, unsigned long)> func = [&](↩
159     unsigned long src, unsigned long dst) {
160         printw(" %d ", dst);
161         return true;
162     };
163     mvprintw(0, (int) (col - strlen(title)) / 2, "%s", title)↩
164     ;
```

```
165     mvprintw(1, Offset - 2, " 顶点 | 邻接点");
166     for (unsigned long vex = 0; vex != graph->vexCount(); ++
167     vex) {
168         mvprintw((int) vex + 2, Offset, " %d |", vex);
169         graph->foreach(vex, func);
170     }
171     move(-1, -1);
172     refresh();
173     getch();
174 }
175
176 void printDot(Graph *&graph) {
177     if (graph == nullptr)
178         return;
179     mvprintw(row - 1, Offset, " 输入要写入的文件名:"))
180     ;
181     refresh();
182     echo();
183     char filename[50];
184     getnstr(filename, sizeof(filename));
185     ofstream Stream(filename);
186     if (Stream) {
187         graph->printDot(Stream);
188         Stream.close();
189         noecho();
190     } else {
191         return;
192     }
193     noecho();
194 }
195
196 void menu(int choice, Graph *&graph) {
197     if (choice > 2 && graph == nullptr) {
```

```
198         mvprintw(row - 1, Offset, " 图未建立");
199         refresh();
200         getch();
201         return;
202     }
203     switch (choice) {
204     case 0:delete graph;
205         graph = new GraphT(0);
206         readFromFile(graph);
207         break;
208     case 1:delete graph;
209         graph = new GraphM(0);
210         readFromFile(graph);
211         break;
212     case 2:delete graph;
213         graph = new GraphL(0);
214         readFromFile(graph);
215         break;
216     case 3:outDegree(graph);
217         break;
218     case 4:printGraph(graph);
219         break;
220     case 5:printDot(graph);
221         break;
222     case 6:addEdge(graph);
223         graph->reset();
224         break;
225     case 7:delEdge(graph);
226         break;
227     case 8:graph->reset();
228         break;
229     case 9:DFS(graph);
230         break;
```

```
231     case 10:DFSR(graph);
232         break;
233     case 11:BFS(graph);
234         break;
235     default:break;
236     }
237     if (graph->vexCount() == 0) {
238         delete graph;
239         graph = nullptr;
240     }
241 }
242 void DFS(Graph *&graph) {
243     if (graph == nullptr)
244         return;
245     function<void(unsigned long)> func = [&](unsigned long
    )
246     dst) {
247         printw("%d ", dst);
248     };
249     mvprintw(row - 1, Offset, " 先深遍历: ");
250 
251     //计时准备
252     struct timeval tpstart{}, tpend{};
253     double timeuse;
254     gettimeofday(&tpstart, nullptr);
255     //遍历
256     graph->DFS(func);
257 
258     //计时结束
259     gettimeofday(&tpend, nullptr);
260     timeuse = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + )
261     tpend.tv_usec - tpstart.tv_usec;)
262     //注意, 秒的读数和微秒的读数都应计算在 )
```

```
263     ĘĚ
264     mvprintw(row - 2, Offset, " 用时:%lfus", timeuse);
265
266     refresh();
267     getch();
268 }
269 void DFSR(Graph *&graph) {
270     if (graph == nullptr)
271         return;
272     function<void(unsigned long)> func = [&](unsigned long
⤸
273     dst) {
274         printw("%d ", dst);
275     };
276     mvprintw(row - 1, Offset, " 先深遍历: ");
277
278     //计时准备
279     struct timeval tpstart{}, tpend{};
280     double timeuse;
281     gettimeofday(&tpstart, nullptr);
282     //遍历
283     graph->DFSR(func);
284     //计时结束
285     gettimeofday(&tpend, nullptr);
286     timeuse = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + ⤸
287     tpend.tv_usec - tpstart.tv_usec;⤸
288     //注意, 秒的读数和微秒的读数都应计算在 ⤸
289     ĘĚ
290     mvprintw(row - 2, Offset, " 用时:%lfus", timeuse);
291
292     refresh();
293     getch();
294 }
```

```
295 void BFS(Graph *&graph) {
296     if (graph == nullptr)
297         return;
298     function<void(unsigned long)> func = [&](unsigned long
↩
299     dst) {
300         printw("%d ", dst);
301     };
302     mvprintw(row - 1, Offset, " 先深遍历: ");
303 
304     //计时准备
305     struct timeval tpstart{}, tpend{};
306     double timeuse;
307     gettimeofday(&tpstart, nullptr);
308     //遍历
309     graph->BFS(func);
310     //计时结束
311     gettimeofday(&tpend, nullptr);
312     timeuse = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + ↩
313     tpend.tv_usec - tpstart.tv_usec;↩
314     //注意，秒的读数和微秒的读数都应计算在 ↩
315     ĘĚ
316     mvprintw(row - 2, Offset, " 用时:%lfus", timeuse);
317 
318     refresh();
319     getch();
320 }
```