```
01 //
02 // Created by along on 17-11-26.
03 //
04
05 #ifndef PROJECT_GRAPH_H
06 #define PROJECT_GRAPH_H
07
08 #ifdef USE_BOOST_LIB
09 #include <boost/dynamic_bitset.hpp>
10 #endif
11
12 #include <vector>
13 #include <forward_list>
14 #include <functional>
15
16 class Graph {
17 public:
18     /**
19      * 产生一个顶点为 0 -- n-1 的图
20      * @param n
21      */
22     explicit Graph(unsigned long n) : vexNum(n), edgeNum(0) {}
23     };
24
25     /**
26      * 使用已有的图构造一个新图
27      * @param rhs
28      */
29     Graph(const Graph &rhs);
30
31     /**
32      * 析构函数，避免内存泄漏
33      */
```

```
34      virtual ~Graph() { clear(); };
35
36      /**
37       * 拷贝赋值函数
38       * @param rhs
39       * @return
40       */
41      Graph &operator=(const Graph &rhs);
42
43      /**
44       * 添加一条边
45       * @param source
46       * @param sink
47       */
48      virtual void addEdge(unsigned long source, unsigned long
)
49      sink);
50
51      /**
52       * 删除一条边
53       * @param source
54       * @param sink
55       */
56      virtual void delEdge(unsigned long source, unsigned long
)
57      sink);
58
59      /**
60       * 返回顶点个数
61       * @return
62       */
63      virtual unsigned long vexCount() const;
64
```

```
65      /**
66       * 边的个数
67       * @return
68       */
69      virtual unsigned long edgeCount() const;
70
71      /**
72       * 顶点的出度
73       * @param source
74       * @return
75       */
76      virtual unsigned long outDegree(unsigned long source) ↩
77      const;
78
79      /**
80       * 顶点的入度
81       * @param source
82       * @return
83       */
84      virtual unsigned long inDegree(unsigned long source) ↩
85      const;
86
87      /**
88       * 两个顶点之间是否有边
89       * @param source
90       * @param sink
91       * @return
92       */
93      virtual bool hasEdge(unsigned long source, unsigned long
↩
94      sink) const;
95
96      /**
```

```
97      * 遍历与某个顶点相临接的所有顶点
98      * 当 func 返回值为 false 的时候可以停止访问
99      * @param source
100      * @param func
101      */
102     virtual void foreach(unsigned long source, std::function<
103     bool(unsigned long, unsigned long)> &func) const = 0;
104
105     /**
106      * 先深遍历
107      * @param DFSTree
108      * @param out
109      */
110     virtual void DFS(std::function<void(unsigned long)> 
111     &visit) const;
112
113     /**
114      * 先深遍历（递归）
115      * @param DFSTree
116      * @param out
117      */
118     virtual void DFSR(std::function<void(unsigned long)> 
119     &visit) const;
120
121     /**
122      * 先广遍历
123      * @param BFSTree
124      * @param out
125      */
126     virtual void BFS(std::function<void(unsigned long)> 
127     &visit) const;
128
129     /**
```

```
130      * 带起始点的先深遍历
131      * @param DFSTree
132      * @param start
133      * @param visit
134      */
135     virtual void DFS(Graph &DFSTree, unsigned long start, ↩
136     std::function<void(unsigned long)> &visit) const;
137
138     /**
139      * 带起始点的先深遍历（递归）
140      * @param DFSTree
141      * @param start
142      * @param out
143      */
144     virtual void DFSR(Graph &DFSTree, unsigned long start, ↩
145     std::function<void(unsigned long)> &visit) const;
146
147     /**
148      * 带起始点的先广遍历
149      * @param BFSTree
150      * @param start
151      * @param out
152      */
153     virtual void BFS(Graph &BFSTree, unsigned long start, ↩
154     std::function<void(unsigned long)> &visit) const;
155
156     /**
157      * 重置
158      */
159     virtual void reset();
160
161     /**
162      * 重置
```

```
163      * @param vexNum
164      */
165     virtual void reset(unsigned long vexNum);
166 
167     /**
168      * 将 dot 图打印到流
169      * @param out
170      */
171     void printDot(std::ostream &out);
172 
173     /**
174      * 从文件构造一个图
175      * @param filename
176      */
177     void resetFromStream(std::istream &theStream);
178 protected:
179     /**
180      * 对数据进行清空
181      */
182     virtual void clear();
183 private:
184 
185     /**
186      * 克隆一个图
187      * @param graph
188      */
189     void clone(const Graph &graph);
190 
191     unsigned long vexNum;
192     unsigned long edgeNum;
193 };
194 
195 /**
```

```
196  * 图的邻接表实现
197  * T:Table
198  */
199 class GraphT : public Graph {
200 public:
201     explicit GraphT(unsigned long n);
202     explicit GraphT(const Graph &rhs);
203     void addEdge(unsigned long source, unsigned long sink) ↩
204     override;
205     void delEdge(unsigned long source, unsigned long sink) ↩
206     override;
207     inline unsigned long vexCount() const override;
208     unsigned long edgeCount() const override;
209     unsigned long outDegree(unsigned long source) const ↩
210     override;
211     unsigned long inDegree(unsigned long source) const ↩
212     override;
213     bool hasEdge(unsigned long source, unsigned long sink) ↩
214     const override;
215     void foreach(unsigned long source, std::function<bool(↩
216     unsigned long, unsigned long)> &func) const override;
217     void reset() override;
218     void reset(unsigned long vexNum) override;
219 private:
220     void clear() override;
221     /** 顶点表的数据结构 */
222     typedef struct VexNode {
223         unsigned long in;
224         unsigned long out;
225         std::forward_list<unsigned long> adjVex;
226     } VexNode;
227     /** 邻接表顶点 */
228     std::vector<VexNode> vexes;
```

```
229 };
230
231 /**
232  * 图的邻接矩阵实现
233  * M:Matrix
234  */
235 class GraphM : public Graph {
236 public:
237     explicit GraphM(unsigned long n);
238     explicit GraphM(const Graph &rhs);
239     void addEdge(unsigned long source, unsigned long sink) ↩
240     override;
241     void delEdge(unsigned long source, unsigned long sink) ↩
242     override;
243     inline unsigned long vexCount() const override;
244     unsigned long edgeCount() const override;
245     unsigned long outDegree(unsigned long source) const ↩
246     override;
247     unsigned long inDegree(unsigned long source) const ↩
248     override;
249     bool hasEdge(unsigned long source, unsigned long sink) ↩
250     const override;
251     void foreach(unsigned long source, std::function<bool(↩
252     unsigned long, unsigned long)> &func) const override;
253     void reset() override;
254     void reset(unsigned long vexNum) override;
255 private:
256     void clear() override;
257 #ifdef USE_BOOST_LIB
258     std::vector<boost::dynamic_bitset<>> vexes;
259 #else
260     std::vector<std::vector<bool>> vexes;
261 #endif
```

```
262 };
263
264 /**
265  * 图的十字链表实现
266  * L:List
267  */
268 class GraphL : public Graph {
269 public:
270     explicit GraphL(unsigned long n);
271     explicit GraphL(const Graph &rhs);
272     ~GraphL() override { clear(); };
273     void addEdge(unsigned long source, unsigned long sink) ↩
274     override;
275     void delEdge(unsigned long source, unsigned long sink) ↩
276     override;
277     unsigned long vexCount() const override;
278     unsigned long edgeCount() const override;
279     unsigned long outDegree(unsigned long source) const ↩
280     override;
281     unsigned long inDegree(unsigned long source) const ↩
282     override;
283     bool hasEdge(unsigned long source, unsigned long sink) ↩
284     const override;
285     void foreach(unsigned long source, std::function<bool(↩
286     unsigned long, unsigned long)> &func) const override;
287     void foreachIn(unsigned long dst, std::function<bool(↩
288     unsigned long, unsigned long)> &func) const;
289     void reset() override;
290     void reset(unsigned long vexNum) override;
291 private:
292     typedef struct ArcBox {
293         unsigned long headVex, tailVex;
294         struct ArcBox *hLink, *tLink;
```

```
295         ArcBox(unsigned long head, unsigned long tail, ↵
296         ArcBox *headLink, ArcBox *tailLink) :
297             headVex(head), tailVex(tail), hLink(headLink), ↵
298             tLink(tailLink) {}
299     } ArcBox;
300
301     typedef struct VexNode {
302         unsigned long in, out;
303         ArcBox *firstIn, *firstOut;
304         VexNode() : in(0), out(0), firstIn(nullptr), ↵
305         firstOut(nullptr) {};
306     } VexNode;
307
308     void clear() override;
309     std::vector<VexNode> vexes;
310 };
311
312 #endif //PROJECT_GRAPH_H
```