练习15.34：针对图15.3（第565页）构建的表达式：

1. 列举出在处理表达式的过程中执行的所有构造函数。
2. 列举出cout<<q所调用的rep。
3. 列举出q.eval()所调用的eval。

文件列表：

essay.txt

Query.h

TextQuery.h

15.36.cpp

详细内容：

essay.txt

Alice Emma has long flowing red hair.

Her Daddy say when the wind blows

through her hair, it looks almost alive,

like a fiery bird in flight.

A beautiful fiery bird, he tells her,

magical but untamed.

"Daddy, shush, there is no such thing."

she tells him, at the same time wanting

him to tell her more.

Shyly, she asks, "I mean, Daddy, is there?"

 C++ Code

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 |  | /\*Author     whulinus 1034765650@qq.com Program     Quary.h History     version 1.0 2016.8.10 \*/ #ifndef QUARY\_H #define QUARY\_H #include <iostream> #include <fstream> #include <string> #include <memory> #include <iterator> #include <algorithm> #include "TextQuery.h" class Query\_base { public: //friends     friend class Query; //constructors //copy control members //copy constructor //move constructor //copy-assignment operator //move-assignment operator //destructor //overloaded assignment operator //functions protected:     using line\_no=std::vector<std::string>::size\_type;     virtual ~Query\_base()=default; private: //private functions     virtual QueryResult eval(const TextQuery &)const=0;     virtual std::string rep()const=0; }; //constructors //overloaded assignment operator //public functions //private functions //other functions class WordQuery:public Query\_base { public: //friends     friend class Query; //constructors     WordQuery(const std::string &s):query\_word(s) {         std::cout<<"word query constructor"<<std::endl;     } //copy control members //copy constructor //move constructor //copy-assignment operator //move-assignment operator //destructor //overloaded assignment operator //functions protected: private:     std::string query\_word; //private functions     QueryResult eval(const TextQuery &t) const {         std::cout<<"WordQuery eval"<<std::endl;         return t.query(query\_word);     }     std::string rep()const {         std::cout<<"WordQuery:"<<query\_word<<std::endl;         return query\_word;     } }; //constructors //overloaded assignment operator //public functions //private functions //other functions class Query { public: //friends     friend Query operator~(const Query &);     friend Query operator|(const Query &,const Query &);     friend Query operator&(const Query &,const Query &);     friend std::ostream &operator<<(std::ostream &,const Query &); //constructors     Query(const std::string &s):q(new WordQuery(s)) {         std::cout<<"query constructor"<<std::endl;     } //copy control members //copy constructor //move constructor //copy-assignment operator //move-assignment operator //destructor //overloaded assignment operator //functions     QueryResult eval(const TextQuery &t)const {         std::cout<<"query eval"<<std::endl;         return q->eval(t);     }     std::string rep()const {         std::string s=q->rep();         std::cout<<"Query:"<<s<<std::endl;         return s;     } protected: private:     Query(Query\_base\* query):q(query) {     }     Query\_base\* q; //private functions }; //constructors //overloaded assignment operator std::ostream &operator<<(std::ostream &os,const Query &query) {     return os<<query.rep(); } //public functions //private functions //other functions  class NotQuery:public Query\_base { public: //friends     friend Query operator~(const Query &); //constructors     NotQuery(const Query &q):query(q) {         std::cout<<"~query constructor"<<std::endl;     } //copy control members //copy constructor //move constructor //copy-assignment operator //move-assignment operator //destructor //overloaded assignment operator //functions     QueryResult eval(const TextQuery &)const;     std::string rep()const {         std::string s="~("+query.rep()+")";         std::cout<<"NotQuery:"<<s<<std::endl;         return s;     } protected: private:     Query query; //private functions }; //constructors //overloaded assignment operator inline Query operator~(const Query &q) {     return new NotQuery(q); } //public functions QueryResult NotQuery::eval(const TextQuery &text)const {     std::cout<<"~query eval"<<std::endl;     auto result=query.eval(text);     auto ret\_lines=std::make\_shared<std::set<line\_no>>();     auto beg=result.begin(),end=result.end();     auto sz=result.get\_file()->size();     for(size\_t n=0; n!=sz; ++n)         if(beg==end||\*beg!=n)             ret\_lines->insert(n);         else if(beg!=end)             ++beg;     return QueryResult(rep(),ret\_lines,result.get\_file()); } //private functions //other functions class BinaryQuery:public Query\_base { public: //friends //copy control members //copy constructor //move constructor //copy-assignment operator //move-assignment operator //destructor //overloaded assignment operator protected: //constructors     BinaryQuery(const Query &l,const Query &r,std::string s):lhs(l),rhs(r),         opSym(s) {std::cout<<"binaryquery constructor"<<std::endl;     } //functions     std::string rep()const {         std::string s="("+lhs.rep()+" "+opSym+" "+rhs.rep()+")";         std::cout<<"BinaryQuery:"<<s<<std::endl;         return s;     } //data members     Query lhs,rhs;     std::string opSym; private: //private functions }; //constructors //overloaded assignment operator //public functions //private functions //other functions class AndQuery:public BinaryQuery { public: //friends     friend Query operator&(const Query &,const Query &); //constructors     AndQuery(const Query &l,const Query &r):BinaryQuery(l,r,"&") {         std::cout<<"&query constructor"<<std::endl;     } //copy control members //copy constructor //move constructor //copy-assignment operator //move-assignment operator //destructor //overloaded assignment operator //functions     QueryResult eval(const TextQuery &)const; protected: private: //data members //private functions }; //constructors //overloaded assignment operator inline Query operator&(const Query &l,const Query &r) {     return new AndQuery(l,r); } //public functions QueryResult AndQuery::eval(const TextQuery &text)const {     std::cout<<"&query eval"<<std::endl;     auto left=lhs.eval(text),right=rhs.eval(text);     auto ret\_lines=std::make\_shared<std::set<line\_no>>();     set\_intersection(left.begin(),left.end(),right.begin(),right.end(),                      inserter(\*ret\_lines,ret\_lines->begin()));     return QueryResult(rep(),ret\_lines,left.get\_file()); } //private functions //other functions class OrQuery:public BinaryQuery { public: //friends     friend Query operator|(const Query &,const Query &); //constructors     OrQuery(const Query &l,const Query &r):BinaryQuery(l,r,"|") {         std::cout<<"|query constructor"<<std::endl;     } //copy control members //copy constructor //move constructor //copy-assignment operator //move-assignment operator //destructor //overloaded assignment operator //functions     QueryResult eval(const TextQuery &)const; protected: private: //data members //private functions }; //constructors //overloaded assignment operator inline Query operator|(const Query &l,const Query &r) {     return new OrQuery(l,r); } //public functions QueryResult OrQuery::eval(const TextQuery &text)const {     std::cout<<"|query eval"<<std::endl;     auto left=lhs.eval(text),right=rhs.eval(text);     auto ret\_lines=std::make\_shared<std::set<line\_no>>(left.begin(),left.end());     ret\_lines->insert(right.begin(),right.end());     return QueryResult(rep(),ret\_lines,left.get\_file()); } //private functions //other functions std::ifstream &open\_file(std::ifstream &,const std::string &); TextQuery get\_file(int i,char \*\*s){     std::ifstream cin(s[i]);     return TextQuery(cin); } #endif |

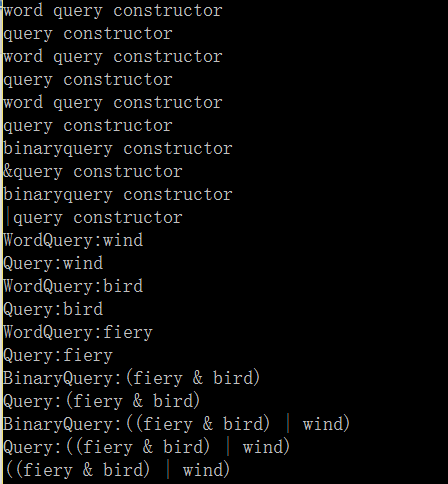
 C++ Code

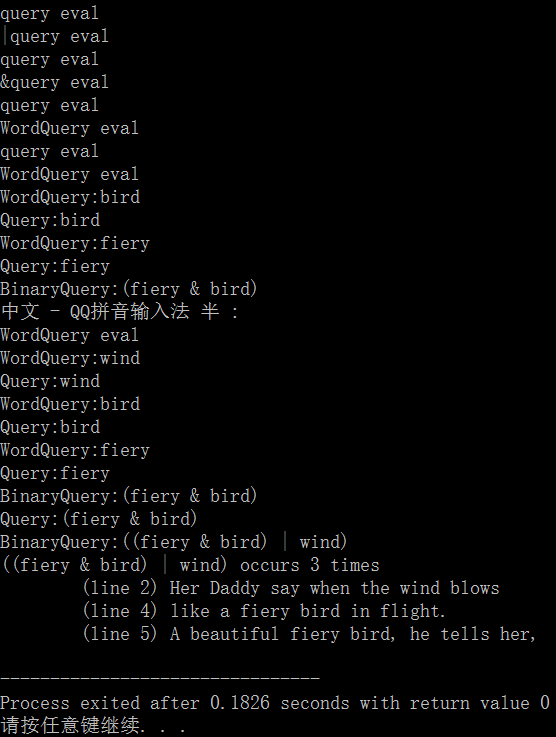
|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 |  | /\*Author  \*  whulinus 1034765650@qq.com  \*Program     TextQuery.h  \*History  \*  ver 1.0 2016.7.12  \*/ #ifndef TEXTQUERY\_H #define TEXTQUERY\_H #include <string> #include <set> #include <map> #include <vector> #include <memory> #include <sstream> #include <fstream> std::string make\_plural(std::size\_t ctr,const std::string &word,                         const std::string &ending) {     return (ctr>1)?word+ending:word; } class QueryResult; class TextQuery { public:     using line\_no=std::vector<std::string>::size\_type;     TextQuery(std::ifstream &); //functions     QueryResult query(const std::string &) const;  private:     std::shared\_ptr<std::vector<std::string>> file;     std::map<std::string,std::shared\_ptr<std::set<line\_no>>> wm; //private functions      static std::string cleanup\_str(const std::string&); }; std::string TextQuery::cleanup\_str(const std::string &word){     std::string ret;     for (auto it = word.begin(); it != word.end(); ++it)          if (!ispunct(\*it))             ret += \*it;     return ret; } TextQuery::TextQuery(std::ifstream &is):file(new std::vector<std::string>) {     std::string text;     while(getline(is,text)) {         file->push\_back(text);         int n=file->size()-1;         std::istringstream line(text);         std::string word;         while(line>>word) {             word = cleanup\_str(word);             auto &lines=wm[word];             if(!lines)                 lines.reset(new std::set<line\_no>);             lines->insert(n);         }     } } struct QueryResult {     typedef std::vector<std::string>::size\_type line\_no;     typedef std::set<line\_no>::const\_iterator line\_it; //friends     friend std::ostream &print(std::ostream &,const QueryResult &); //constructors     QueryResult(std::string s,std::shared\_ptr<std::set<line\_no>> p,                 std::shared\_ptr<std::vector<std::string>> f):sought(s),lines(p),file(f) {} //functions     std::set<line\_no>::size\_type size() const  {         return lines->size();     }     line\_it begin() const {         return lines->cbegin();     }     line\_it end() const   {         return lines->cend();     }     std::shared\_ptr<std::vector<std::string>> get\_file() {         return file;     } private:     std::shared\_ptr<std::vector<std::string>> file;     std::string sought;     std::shared\_ptr<std::set<line\_no>> lines; }; std::ostream &print(std::ostream &os,const QueryResult &qr) {     os<<qr.sought<<" occurs "<<qr.lines->size()<<" "<<make\_plural(qr.lines->size(),                                                                   "time","s")<<std::endl;     for(auto num:\*qr.lines)         os<<"\t(line "<<num+1<<") "<<\*(qr.file->begin()+num)<<std::endl;     return os; } QueryResult TextQuery::query(const std::string &sought)const {     static std::shared\_ptr<std::set<line\_no>> nodata(new std::set<line\_no>);     auto loc=wm.find(sought);     if(loc==wm.end())         return QueryResult(sought,nodata,file);     else         return QueryResult(sought,loc->second,file); } #endif |

 C++ Code

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 |  | /\* Author     whulinus 1034765650@qq.com Program     exercise 15.36 History     version 1.0     2016.8.12 \*/ #include <iostream> #include <fstream> #include "Query.h" using namespace std; int main(int argc,char \*argv[]) {     ifstream cin("essay.txt");     TextQuery t(cin);     Query q=Query("fiery") & Query("bird") | Query("wind");     cout<<q<<endl;     print(cout,q.eval(t));     return 0; } |

运行结果:





结果流分析：

(a)构造函数

首先是3个Query->WordQuery

然后AndQuery->binaryQuery

最后OrQuery->binaryQuery

(b)rep调用

类似于构造函数

(c)eval调用

Query->OrQuery分流

左边AndQuery分流

左边Query->WordQuery

右边Query-> WordQuery

左边返回时输出了rep()

右边Query->WordQuery

右边返回时输出了rep()

输出结果

ps:1-15章之前的习题解答请google，github上非常全