Chương 16 – ỨNG DỤNG XỬ LÝ VĂN BẢN

Phần này minh họa một ứng dụng có sử dụng cả lớp `List` và `String`. Đó là một chương trình xử lý văn bản, tuy chỉ có một vài lệnh đơn giản, nhưng nó cũng minh họa được những ý tưởng cơ bản để xây dựng những chương trình xử lý văn bản lớn và tinh tế hơn.

16.1. Các đặc tả

Chương trình xử lý văn bản của chúng ta cho phép đọc một tập tin từ đĩa vào bộ nhớ mà chúng ta gọi là vùng đệm (*buffer*). Vùng đệm này được hiện thực như một đối tượng của lớp **`Editor`**. Mỗi dòng văn bản trong đối tượng `Editor` là một `String`. Do đó lớp `Editor` sẽ được thừa kế từ lớp `List` các `String`. Các lệnh xử lý văn bản được chia làm hai nhóm: nhóm các tác vụ của `List` sẽ xử lý cho các dòng văn bản, và nhóm các tác vụ của `String` sẽ xử lý cho các ký tự trong mỗi dòng văn bản.

Tại mỗi thời điểm, người sử dụng có thể nhập hoặc các ký tự để chèn vào văn bản, hoặc các lệnh xử lý cho phần văn bản đã có. Chương trình xử lý văn bản cần biết bỏ qua những ký tự nhập không hợp lệ, nhận biết các lệnh, hoặc hỏi lại người sử dụng trước khi thực hiện các lệnh quan trọng (chẳng hạn như xóa toàn bộ vùng đệm).

Chương trình xử lý văn bản có các lệnh dưới đây. Mỗi lệnh sẽ được người sử dụng nhập vào khi có dấu nhắc '??' và có thể nhập chữ hoa hoặc chữ thường.

'R' (*Read*) Đọc tập tin văn bản vào vùng đệm. Tên tập tin văn bản đã được chỉ ra khi chạy chương trình. Nội dung có sẵn trong vùng đệm được xóa sạch. Dòng đầu tiên của văn bản được xem là dòng hiện tại.

'W'(*Write*) Ghi nội dung trong vùng đệm vào tập tin văn bản có tên đã được chỉ ra khi chạy chương trình. Vùng đệm cũng như dòng hiện tại đều không đổi.

'I' (*Insert*) Thêm một dòng mới. Người sử dụng có thể nhập số thứ tự của dòng mới sẽ được thêm vào.

'D' (*Delete*) Xóa dòng hiện tại và chuyển đến dòng kế.

'F' (*Find*) Bắt đầu từ dòng hiện tại, tìm dòng đầu tiên có chứa chuỗi ký tự do người sử dụng yêu cầu.

'L' (*Length*) Cho biết số ký tự có trong dòng hiện tại và số dòng có trong vùng đệm.

'C' (*Change*) Đổi một chuỗi ký tự sang một chuỗi ký tự khác. Chỉ đổi trong dòng hiện tại.

'Q' (*Quit*) Thoát khỏi chương trình.

'H' (*Help*) In giải thích về các lệnh. Có thể dùng '?' thay cho 'H'.

CuuDuongThanCong.com https://fb.com/tailieudientucntt

'N' (*Next*) Chuyển sang dòng kế trong vùng đệm.
'P' (*Previous*) Trở về dòng trước trong vùng đệm.
'B' (*Beginning*) Chuyển đến dòng đầu tiên trong vùng đệm.
'E' (*End*) Chuyển đến dòng cuối trong vùng đệm.
'G' (*Go*) Chuyển đến dòng có số thứ tự do người sử dụng yêu cầu.
'S' (*Subtitute*) Thay dòng hiện tại bởi dòng do người sử dụng nhập vào. Chương trình sẽ in dòng sẽ bị thay thế để kiểm tra lại và hỏi người sử dụng nhập dòng mới.
'V' (*View*) Xem toàn bộ nội dung trong vùng đệm.

16.2. Hiện thực

16.2.1. Chương trình chính

Nhiệm vụ đầu tiên của chương trình chính là sử dụng các thông số nhập vào từ dòng lệnh để mở tập tin đọc và tập tin ghi. Cách sử dụng chương trình:

```
edit  infile  outfile
```

trong đó `infile` và `outfile` là tên tập tin đọc và tên tập tin ghi tương ứng. Khi các tập tin đã mở thành công, chương trình khai báo một đối tượng `Editor` gọi là `buffer`, lặp lại việc chạy phương thức `get_command` của `Editor` để đọc các lệnh rồi xử lý các lệnh này.

```
int main(int argc, char *argv[]) // count, values of command-line arguments
/*
pre:  Thông số của dòng lệnh là tên tập tin đọc và tập tin ghi.
post: Chương trình đọc nội dung từ tập tin đọc, cho phép soạn thảo, chỉnh sửa văn bản, và ghi
      vào tập tin ghi.
uses: Các phương thức của lớp Editor.
*/
{
   if (argc != 3) {
      cout << "Usage:\n\t edit  inputfile  outputfile" << endl;
      exit (1);
   }
   ifstream file_in(argv[1]);              // Khai báo và mở tập tin đọc.
   if (file_in == 0) {
      cout << "Can't open input file " << argv[1] << endl;
      exit (1);
   }
   ofstream file_out(argv[2]);             // Khai báo và mở tập tin ghi.
   if (file_out == 0) {
      cout << "Can't open output file " << argv[2] << endl;
      exit (1);
   }
   Editor buffer(&file_in, &file_out);
   while (buffer.get_command())
      buffer.run_command();
}
```


CuuDuongThanCong.com https://fb.com/tailieudientucntt

16.2.2. Đặc tả lớp Editor

Lớp `Editor` cần chứa một `List` các đối tượng `String`, và cho phép các tác vụ di chuyển theo cả hai hướng của `List` một cách hiệu quả. Chúng ta cũng không biết trước vùng đệm sẽ phải lớn bao nhiêu, do đó chúng ta sẽ khai báo lớp `Editor` dẫn xuất từ hiện thực danh sách liên kết kép (*doubly linked list*). Lớp dẫn xuất này cần bổ sung thêm hai phương thức `get_command` và `run_command` mà chương trình chính sẽ gọi. Ngoài ra lớp `Editor` còn cần thêm thuộc tính để chứa lệnh từ người sử dụng (`user_command`) và các tham chiếu đến dòng nhập và xuất (`infile` và `outfile`).

```
class Editor:public List<String> {
public:
   Editor(ifstream *file_in, ofstream *file_out);
   bool get_command();
   void run_command();
private:
   ifstream *infile;
   ofstream *outfile;
   char user_command;

//  Các hàm phụ trợ
   Error_code next_line();
   Error_code previous_line();
   Error_code goto_line();
   Error_code insert_line();
   Error_code substitute_line();
   Error_code change_line();
   void read_file();
   void write_file();
   void find_string();
};
```

Trong đặc tả trên chúng ta còn thấy một số hàm phụ trợ để hiện thực các lệnh xử lý văn bản khác nhau.

Constructor thực hiện nối dòng nhập và dòng xuất với đối tượng của lớp `Editor`.

```
Editor::Editor(ifstream *file_in, ofstream *file_out)
/*
post: Khởi tạo đối tượng Editor với trị cho hai thuộc tính infile, outfile.
*/
{
   infile = file_in;
   outfile = file_out;
}
```


CuuDuongThanCong.com https://fb.com/tailieudientucntt

16.2.3. Nhận lệnh từ người sử dụng

Do chương trình xử lý văn bản phải biết bỏ qua những ký tự nhập không hợp lệ, nên các lệnh nhập vào phải được kiểm tra kỹ lưỡng. Chương trình dùng hàm `tolower` chuyển ký tự hoa thành ký tự thường có trong thư viện `<cctype>`, cho phép người sử dụng có thể nhập chữ hoa hoặc chữ thường. `Get_command` sẽ in dòng hiện tại, hiện dấu nhắc chờ lệnh, đổi lệnh sang ký tự thường.

```
bool Editor::get_command()
/*
post: Gán trị cho thuộc tính user_command; trả về true trừ khi người sử dụng gõ 'q'
uses: Hàm tolower của thư viện C..
*/
{
   if (current != NULL)
      cout << current_position << " : "
           << current->entry.c_str() << "\n??" << flush;
   else
      cout << "File is empty. \n??" << flush;

   cin >> user_command;// Bỏ qua các khoảng trắng và nhận lệnh của người sử dụng
   user_command = tolower(user_command);
   while (cin.get() != '\n');          //  Bỏ qua phím "enter"
   if (user_command == 'q')
      return false;
   else
      return true;
}
```

16.2.4. Thực hiện lệnh

Phương thức `run_command` chứa lệnh `switch` để chọn các hàm khác nhau tương ứng với các lệnh cần thực hiện. Một vài lệnh trong số này (tựa như `remove`) là các phương thức của `List`. Những lệnh khác dựa trên các tác vụ xử lý của `List` nhưng có bổ sung xử lý những yêu cầu của người sử dụng.

```
void Editor::run_command()
/*
post: Lệnh trong user_command được thực hiện.
uses: Các phương thức và các hàm phụ trợ của các lớp Editor,
      String, vá các hàm xử lý chuỗi ký tự.
*/
{
   String temp_string;
   switch (user_command) {
   case 'b':
      if (empty())
         cout << " Warning: empty buffer " << endl;
      else
         while (previous_line() == success);
      break;
   case 'c':
```


CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
      if (empty())
         cout << " Warning: Empty file" << endl;
      else if (change_line() != success)
         cout << " Error: Substitution failed " << endl;
      break;

   case 'd':
      if (remove(current_position, temp_string) != success)
         cout << " Error: Deletion failed " << endl;
      break;

   case 'e':
      if (empty())
         cout << " Warning: empty buffer " << endl;
      else
         while (next_line() == success)
            ;
      break;

   case 'f':
      if (empty())
         cout << " Warning: Empty file" << endl;
      else
         find_string();
      break;

   case 'g':
      if (goto_line() != success)
         cout << " Warning: No such line" << endl;
      break;

   case '?':
   case 'h':
      cout  << "Valid commands are: b(egin) c(hange) d(el) e(nd)"
            << endl
            << "f(ind) g(o) h(elp) i(nsert) l(ength) n(ext) p(rior) "
            << endl
            << "q(uit) r(ead) s(ubstitute) v(iew) w(rite) " << endl;

   case 'i':
      if (insert_line() != success)
         cout << " Error: Insertion failed " << endl;
      break;

   case 'l':
      cout << "There are " << size() << " lines in the file." << endl;
      if (!empty())
         cout << "Current line length is "
              << strlen((current->entry).c_str()) << endl;
      break;

   case 'n':
      if (next_line() != success)
         cout << " Warning: at end of buffer" << endl;
      break;
   case 'p':
      if (previous_line() != success)
         cout << " Warning: at start of buffer" << endl;
      break;
```


CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
   case 'r':
      read_file();
      break;

   case 's':
      if (substitute_line() != success)
         cout << " Error: Substitution failed " << endl;
      break;

   case 'v':
      traverse(write);
      break;

   case 'w':
      if (empty())
         cout << " Warning: Empty file" << endl;
      else
         write_file();
      break;

   default :
      cout << "Press h or ? for help or enter a valid command: ";
   }
}
```

16.2.5. Đọc và ghi tập tin

Tập tin sẽ được đọc và ghi đè lên vùng đệm. Nếu vùng đệm không rỗng, cần có thông báo hỏi lại người sử dụng trước khi thực hiện lệnh.

```
void Editor::read_file()
/*
pre:  Nếu Editor không rỗng thì người sử dụng sẽ trả lời có chép đè nội dung mới lên hay
      không.
post: Đọc tập tin đọc vào Editor. Nội dung cũ nếu có trong Editor sẽ bị chép đè.
uses: Các phương thức và các hàm của String và Editor.
*/

{
   bool proceed = true;
   if (!empty()) {
      cout << "Buffer is not empty; the read will destroy it." << endl;
      cout << " OK to proceed? " << endl;
      if (proceed = user_says_yes()) clear();
   }

  int line_number = 0, terminal_char;
  while (proceed) {
      String in_string = read_in(*infile, terminal_char);
      if (terminal_char == EOF) {
         proceed = false;
         if (strlen(in_string.c_str()) > 0)
              insert(line_number , in_string);
      }
      else insert(line_number++, in_string);
   }
}
```


CuuDuongThanCong.com https://fb.com/tailieudientucntt

16.2.6. Chèn một hàng

Để chèn một dòng mới, trước hết chúng ta đọc một chuỗi ký tự nhờ phương thức `read_in` của lớp `String` trong phần 5.3. Sau đó chuỗi ký tự được chèn vào `List` nhờ phương thức insert của `List`. Chúng ta không cần kiểm tra vùng đệm đã đầy hay chưa do các tác vụ của `List` đã chịu trách nhiệm về việc này.

```
Error_code Editor::insert_line()
/*
post: Mộ dòng do người sử dụng gõ vào sẽ được chèn vào vị trí theo yêu cầu.
uses: Các phương thức và các hàm của String và Editor.
*/
{
   int line_number;
   cout << " Insert what line number? " << flush;
   cin  >> line_number;
   while (cin.get() != '\n');
   cout << " What is the new line to insert? " << flush;
   String to_insert = read_in(cin);
   return insert(line_number, to_insert);
}
```

16.2.7. Tìm một chuỗi ký tự

Đây là một công việc tương đối khó. Việc tìm một chuỗi ký tự do người sử dụng yêu cầu được thực hiện trên toàn vùng đệm. Hàm `strstr` của `String` sẽ tìm chuỗi ký tự được yêu cầu trong dòng hiện tại trước, nếu không có sẽ tìm tiếp ở các dòng kế tiếp trong vùng đệm. Nếu tìm thấy, dòng có chuỗi ký tự cần tìm sẽ được hiển thị và trở thành dòng hiện tại, dấu '^' sẽ chỉ ra vị trí chuỗi ký tự được tìm thấy.

```
void Editor::find_string()
/*
pre:  Editor đang chứa văn bản.
post: Chuỗi ký tự được yêu cầu sẽ được tìm từ dòng hiện tại trở đi. Nếu tìm thấy thì hiện dòng
      chứa chuỗi ký tự đó, đồng thời chỉ ra chuỗi ký tự.
uses: Các phương thức và các hàm của String và Editor.
*/
{
   int index;
   cout << "Enter string to search for:" << endl;
   String search_string = read_in(cin);
   while ((index = strstr(current->entry, search_string)) == -1)
      if (next_line() != success) break;
   if (index == -1) cout << "String was not found.";
   else {
      cout << (current->entry).c_str() << endl;
      for (int i = 0; i < index; i++)
         cout << " ";
      for (int j = 0; j < strlen(search_string.c_str()); j++)
         cout << "^";
   }
   cout << endl;
}
```


CuuDuongThanCong.com https://fb.com/tailieudientucntt

16.2.8. Biến đổi chuỗi ký tự

Hàm `change_line` nhận một chuỗi ký tự cần thay thế từ người sử dụng. Khi chuỗi ký tự này được tìm thấy trong dòng hiện tại, người sử dụng sẽ được yêu cầu nhập chuỗi ký tự để thay thế. Cuối cùng là một loạt các tác vụ của `String` và `C-String` được thực hiện để loại chuỗi ký tự cũ và thế chuỗi ký tự mới vào.

```
Error_code Editor::change_line()
/*
pre:  Editor đang chứa văn bản.
post: Nếu chuỗi ký tự được yêu cầu được tìm thấy trong dòng hiện tại thì sẽ được thay thế bởi
      chuỗi ký tự khác (cũng do người sử dụng gõ vào), rả về success; ngược lại trả về fail.
uses: Các phương thức và các hàm của String và Editor.
*/
{
   Error_code result = success;
   cout << " What text segment do you want to replace? " << flush;
   String old_text = read_in(cin);
   cout << " What new text segment do you want to add in? " << flush;
   String new_text = read_in(cin);

   int index = strstr(current->entry, old_text);
   if (index == -1) result = fail;
   else {
      String new_line;
      strncpy(new_line, current->entry, index);
      strcat(new_line, new_text);
      const char *old_line = (current->entry).c_str();
            strcat(new_line, (String)(old_line + index +                    +
                                                  strlen(old_text.c_str())));
      current->entry = new_line;
   }
   return result;
}
```

Lệnh `strcat(new_line, (String)(old_line+index+strlen(old_text.c_str()))` tìm con trỏ tạm chỉ vị trí bắt đầu phần còn lại ngay sau chuỗi ký tự được thay thế trong dòng cũ, con trỏ này sẽ được chuyển đổi thành đối tượng `String` và nối với `new_line`.

CuuDuongThanCong.com https://fb.com/tailieudientucntt