Chương 05. Chồng hàm và chồng toán tử (function overloading and operator overloading)

I. Chồng hàm

II. Chồng toán tử

III. Các loại biến

I. Chồng hàm (function overloading)

1. Sự cần thiết phải chồng hàm
2. Trình biên dịch và các hàm chồng

1. Sự cần thiết phải chồng hàm

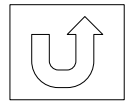
- *Bài tập 1:* Viết hàm tính trung bình cộng của một mảng int, long, float và double.
  - Với bài tập này, bình thường ta phải viết 4 hàm để tính trung bình cho 4 mảng khác nhau và khi gọi hàm ta phải nhớ 4 tên hàm này. Tuy nhiên, C++ cho phép nhiều hàm có tên giống nhau chỉ cần khác nhau về đối số. Việc sử dụng cùng một tên cho nhiều hàm gọi là chồng hàm. Chồng hàm giúp người sử dụng không phải nhớ nhiều tên hàm khác nhau.

2. Trình biên dịch và các hàm chồng

- Làm thế nào mà trình biên dịch có thể phân biệt được các hàm có cùng tên? Trình biên dịch sẽ tạo ra một tên mới cho mỗi hàm bằng cách kết hợp tên hàm với tên kiểu của các đối số.

  Ví dụ: tbc_int_int(), tbc_long_int()

- *Bài tập về nhà:*
  - Viết chương trình tính bình phương của một số int, long, float, double.
  - Làm thế nào để lấy địa chỉ của các hàm được chồng?



II. Chồng toán tử

2.1. Tại sao phải chồng toán tử?

2.2. Chồng các toán tử hai ngôi

2.3. Chồng các toán tử một ngôi

2.4. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

2.5. Chuyển đổi giữa các lớp

2.6. Chồng toán tử gán = và toán tử [ ]

2.7. Chồng toán tử nhập/xuất - Hàm bạn (friend function)

I. Tại sao phải chồng toán tử?

Chồng toán tử là sử dụng các toán tử có sẵn để tác động trên các toán hạng khác nhau, tức là ta có thể định nghĩa tác động của các toán tử trên các đối tượng lớp.

Chồng toán tử giúp chương trình dễ viết, dễ đọc và dễ hiểu. Ví dụ: giả sử ta muốn cộng hai đối tượng của lớp airtime rồi gán kết quả nhận được vào một đối tượng airtime khác. Khi đó, ta viết

at3=at1+at2

sẽ dễ hiểu hơn là viết

at3=at1.add(at2) hay at3.add(at1,at2)

Chương 14. Chồng toán tử

I. Tại sao phải chồng toán tử?

II. Chồng các toán tử hai ngôi

III. Chồng các toán tử một ngôi

IV. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

V. Chuyển đổi giữa các lớp

VI. Chồng toán tử gán = và toán tử [ ]

Chương 14. Chồng toán tử

II. Chồng các toán tử hai ngôi

II.1. Chồng các toán tử số học

II.2. Chồng các toán tử quan hệ

II.3. Chồng các toán tử gán phức hợp

II.1. Chồng các toán tử số học

Ví dụ 1: Viết chương trình cộng hai số phức nhập vào từ bàn phím bằng toán tử cộng +.

Bài về nhà 1: Xây dựng lớp đối tượng xâu ký tự để có thể dùng phép cộng ghép nhiều xâu ký tự thông thường thành một xâu.

II.1. Chồng các toán tử số học

Các toán tử có thể chồng là +, -, *, /

Để chồng một toán tử ta phải định nghĩa một hàm xác định phép toán mà toán tử đó sẽ thực hiện. Hàm chồng toán tử giống như các hàm bình thường, chỉ khác tên hàm là từ khóa operator kết hợp với toán tử: operatorX, trong đó X là toán tử. Ví dụ để chồng toán tử + ta có tên hàm là operator+

Lời gọi hàm chồng toán tử có thể dùng cú pháp giống như hàm bình thường. Ví dụ:

```
t3 = t1.operator+(t2);
```

Nhưng từ khóa operator, dấu chấm và cặp dấu ngoặc () là không cần thiết. Bởi vậy ta viết:

```
t3 = t1 + t2;
```

Chương 14. Chồng toán tử

II. Chồng các toán tử hai ngôi

II.1. Chồng các toán tử số học

II.2. Chồng các toán tử quan hệ

II.3. Chồng các toán tử gán phức hợp

II.2. Chồng các toán tử quan hệ

Ta có thể chồng tất cả các phép toán so sánh (quan hệ).

Bài tập 2: Viết hàm thành viên chồng toán tử so sánh nhỏ hơn (<) để so sánh hai đối tượng lớp airtime.

Chương 14. Chồng toán tử

II. Chồng các toán tử hai ngôi

II.1. Chồng các toán tử số học

II.2. Chồng các toán tử quan hệ

II.3. Chồng các toán tử gán phức hợp

II.3. Chồng các toán tử gán phức hợp

Có thể chồng các toán tử phức hợp sau:

+=, -=, *=, /=

Toán tử gán khác với các toán tử hai ngôi thông thường ở chỗ là chúng thay đổi đối tượng gọi chúng.

Mục đích chính của toán tử gán là thay đổi đối tượng nhưng chúng cũng thường được dùng để trả về giá trị.

Bài tập 3: Chồng toán tử gán += cho lớp airtime sao cho có thể dùng nó để gán các đối tượng airtime cho nhau.

II.3. Chồng các toán tử gán phức hợp (tiếp)

Khi trả về đối tượng ta nên dùng lệnh trả về đặc biệt sau:

Ví dụ: return airtime(hours,minutes);

Lệnh đặc biệt này tạo đối tượng trả về, hàm tạo sao chép không thực hiện.

Chương 14. Chồng toán tử

I. Tại sao phải chồng toán tử?

II. Chồng các toán tử hai ngôi

III. Chồng các toán tử một ngôi

IV. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

V. Chuyển đổi giữa các lớp

VI. Chồng toán tử gán = và toán tử [ ]

III. Chồng các toán tử một ngôi

Toán tử một ngôi là các toán tử chỉ có một toán hạng. Ví dụ: toán tử tăng ++, toán tử giảm --, toán tử dấu âm – và toán tử phủ định logic !. Hay dùng nhất là toán tử tăng giảm. Toán tử tăng và giảm có thể dùng ở trước hoặc sau toán hạng.

Bài tập 4: Chồng toán tử ++ cho lớp airtime để tăng đối tượng airtime lên 1 phút, toán tử ++ có thể đứng trước và sau đối tượng.

III. Chồng các toán tử một ngôi

Toán tử một ngôi là các toán tử chỉ có một toán hạng. Ví dụ: toán tử tăng ++, toán tử giảm --, toán tử dấu âm – và toán tử phủ định logic !. Hay dùng nhất là toán tử tăng giảm. Toán tử tăng và giảm có thể dùng ở trước hoặc sau toán hạng.

Bài tập 4: Chồng toán tử ++ cho lớp airtime để tăng đối tượng airtime lên 1 phút, toán tử ++ có thể đứng trước và sau đối tượng.

Chương 14. Chồng toán tử

I. Tại sao phải chồng toán tử?

II. Chồng các toán tử hai ngôi

III. Chồng các toán tử một ngôi

IV. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

V. Chuyển đổi giữa các lớp

VI. Chồng toán tử gán = và toán tử [ ]

IV. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

Việc chuyển đổi giữa các kiểu cơ bản được thực hiện tự động bởi vì các hàm chuyển đổi giữa các kiểu cơ bản đã có sẵn.

Khi ta tạo ra một lớp và muốn chuyển đổi giữa các đối tượng lớp và các kiểu dữ liệu cơ bản thì chúng ta phải viết hàm chuyển đổi.

Việc chuyển đổi từ các kiểu dữ liệu cơ bản sang các đối tượng được thực hiện bằng hàm tạo một đối số.

Việc chuyển đổi từ các đối tượng lớp sang các kiểu cơ bản được thực hiện bằng hàm chồng toán tử ép kiểu.

IV. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

Hàm chồng toán tử ép kiểu không có kiểu trả về, tên hàm bắt đầu bằng từ khoá operator sau đó là dấu cách rồi đến tên kiểu. Ví dụ: hàm chuyển đổi các đối tượng lớp sang kiểu long có dạng như sau:

```
operator long()
  {
   //Thuc hien chuyen doi o day
   return longvar;
  }
```

Mặc dù trong hàm có lệnh trả về nhưng hàm lại không có kiểu trả về, kiểu trả về ẩn trong tên hàm.

IV. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

Cách gọi hàm chồng toán tử ép kiểu:

Tên_kiểu(Đối tượng)

Ví dụ: long(doituong); ⇔ (long) doituong;

Hàm chồng toán tử ép kiểu được gọi tự động khi ta gán một đối tượng cho một biến kiểu cơ bản hoặc khi khởi tạo một biến kiểu cơ bản.

Bài tập 5: Xây dựng một lớp đối tượng chiều dài đo bằng đơn vị Anh: feet và inches. 1 foot = 12 inches, 1 meter = 3.280833 feet. Một chiều dài 6 feet 2 inches được viết là 6’-2”. Đặt tên lớp là English.

IV. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

Bài tập về nhà: Chuyển đối đối tượng xâu ký tự sang xâu ký tự thông thường.

Chương 14. Chồng toán tử

I. Tại sao phải chồng toán tử?

II. Chồng các toán tử hai ngôi

III. Chồng các toán tử một ngôi

IV. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

V. Chuyển đổi giữa các lớp

VI. Chồng toán tử gán = và toán tử [ ]

V. Chuyển đổi giữa các lớp

Trong nhiều trường hợp, việc chuyển đổi giữa các lớp là không có ý nghĩa.

Có 2 cách để chuyển đổi từ một lớp này sang một lớp khác:

- Dùng hàm tạo một đối số
- Dùng hàm chồng toán tử ép kiểu

Ví dụ: Viết 2 lớp alpha và beta cùng các hàm cần thiết để chuyển từ alpha sang beta và từ beta sang alpha.

Chương 14. Chồng toán tử

I. Tại sao phải chồng toán tử?

II. Chồng các toán tử hai ngôi

III. Chồng các toán tử một ngôi

IV. Chuyển đổi giữa các đối tượng và kiểu dữ liệu cơ bản

V. Chuyển đổi giữa các lớp

VI. Chồng toán tử gán = và toán tử [ ]

Chương 14. Chồng toán tử

VI. Chồng toán tử gán = và toán tử [ ]

VI.1. Chồng toán tử gán đơn giản =

VI.2. Chồng toán tử chỉ số []

VI.1. Chồng toán tử gán đơn giản =

Chúng ta có thể sử dụng toán tử gán để gán các đối tượng cho nhau mà không phải làm gì cả. Tuy nhiên, khi đối tượng sử dụng con trỏ hay làm những việc như đếm, đánh số thứ tự cho chính nó,… thì ta phải viết hàm chồng toán tử gán.

Hàm chồng toán tử gán và hàm tạo sao chép đều thực hiện sao chép dữ liệu từ đối tượng này sang đối tượng khác. Chỉ khác là hàm tạo sao chép tạo ra một đối tượng mới rồi sao chép dữ liệu của một đối tượng khác vào đối tượng mới này, còn hàm chồng toán tử gán chỉ sao chép dữ liệu tới một đối tượng đã có.

VI.1. Chồng toán tử gán đơn giản

Ví dụ: Tạo lớp omega gồm 2 mục dữ liệu: biến xâu name chứa xâu ký tự phản ánh tên đối tượng, biến nguyên snumber chứa số seri (thứ tự) của đối tượng. Chồng toán tử gán sao cho khi gán hai đối tượng thì nội dung biến xâu của đối tượng thay đổi còn số seri thì không thay đổi.

Chương 14. Chồng toán tử

VI. Chồng toán tử gán = và toán tử [ ]

VI.1. Chồng toán tử gán đơn giản =

VI.2. Chồng toán tử chỉ số []

VI.2. Chồng toán tử chỉ số []

Toán tử chỉ số thường được dùng để truy nhập các phần tử của mảng. Chồng toán tử chỉ số để có thể sử dụng ký hiệu [ ] truy nhập các phần tử của một đối tượng mảng.

Bài tập 6 (BTVN): Tạo một lớp mảng có sử dụng toán tử [ ] để nhập vào và đưa ra các phần tử của mảng.

Lưu ý

Khi các hàm chồng toán tử cần trả về chính đối tượng gọi hàm thì ta nên dùng:

return *this

Khai báo kiểu trả về là tham chiếu

this là con trỏ có sẵn, chứa địa chỉ của đối tượng gọi hàm thành viên, do đó *this là đối tượng gọi hàm.

Nếu khai báo kiểu trả về là tham chiếu thì khi trả về đối tượng sẽ không tạo ra đối tượng trung gian.

2.7. Chồng toán tử nhập/ xuất - Hàm bạn

1. Giới thiệu về hàm bạn
2. Những thuận lợi khi dùng hàm bạn
3. Hàm bạn phá vỡ nguyên tắc bao gói thông tin

VII.1. Giới thiệu về hàm bạn

Hàm bạn (**friend function**) là một hàm thông thường, không phải là thành viên của một lớp nhưng có thể truy nhập được tới các thành viên **private** và **protected** của lớp đó.

Để cho một hàm thông thường là hàm bạn của một lớp, trong mô tả lớp ta viết khai báo hàm với từ khóa friend đứng trước.

Ví dụ: friend void show();

Chú ý: Trong mô tả lớp chỉ chứa khai báo hàm bạn, không chứa định nghĩa hàm bạn.

VII.1. Giới thiệu về hàm bạn

Khai báo hàm bạn có thể bất kỳ phần nào trong mô tả lớp. Tuy nhiên nên để ở phần public vì nó là phần giao diện của lớp, nghĩa là bất kỳ người sử dụng lớp nào cũng có thể gọi hàm bạn.

Theo nguyên tắc bao bọc và cất giấu dữ liệu trong LTHĐT, các hàm không phải là thành viên của lớp không thể truy nhập tới dữ liệu **private** và **protected** của một đối tượng. Tuy nhiên, trong một số trường hợp nguyên tắc này rất bất tiện. Các hàm bạn là một cách giải toả sự bất tiện này.

Một hàm có thể khai báo là bạn của nhiều lớp.

VII.2. Những thuận lợi khi dùng hàm bạn

Cho phép khi gọi hàm chồng toán tử thì bên trái toán tử không phải là đối tượng cũng được.

Ví dụ:

VII.2. Những thuận lợi khi dùng hàm bạn

Hàm bạn cho phép dùng ký hiệu hàm: Đôi khi một hàm bạn cho một cú pháp gọi hàm rõ ràng hơn hàm thành viên. Ví dụ, giả sử chúng ta muốn một hàm tính bình phương một đối tượng obj, khi đó cách viết sqr(obj) rõ ràng hơn cách viết obj.sqr()

VII.2. Những thuận lợi khi dùng hàm bạn

Hàm bạn như chiếc cầu nối giữa các lớp: Giả sử ta có một hàm tính toán trên các đối tượng của hai lớp khác nhau. Có thể hàm này có đối số là các đối tượng của hai lớp đó và tính toán trên dữ liệu private của chúng. Làm thế nào để có thể dùng trực tiếp dữ liệu private của hai lớp nếu chúng không có liên quan gì với nhau? Hàm bạn của hai lớp sẽ làm được điều này.

Ví dụ:

VII.3. Hàm bạn phá vỡ nguyên tắc bao gói thông tin

Khi đưa vào hàm bạn, một mặt nó thêm vào sự linh hoạt cho ngôn ngữ, mặt khác nó không còn giữ nguyên tắc là chỉ có các thành viên mới có thể truy nhập dữ liệu **private** của lớp.

Một hàm thông thường muốn là bạn của lớp phải được khai báo ở bên trong mô tả lớp đó. Thường thì người lập trình không truy nhập được mã nguồn của các lớp nên không thể chuyển một hàm thành một hàm bạn của lớp. Ở khía cạnh này tính toàn vẹn của lớp vẫn còn được giữ.

VII.3. Hàm bạn phá vỡ nguyên tắc bao gói thông tin

Mặc dù vậy, hạm bạn vẫn gây ra sự lộn xộn trong tư tưởng LTHĐT.

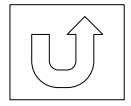
Tóm lại, luôn sử dụng hàm thành viên trừ khi có một lý do bắt buộc phải sử dụng hàm bạn.

III. Các loại biến

1. Sự khác nhau giữa khai báo và định nghĩa
2. Thời gian tồn tại và phạm vi hoạt động của các loại biến

1. Sự khác nhau giữa khai báo và định nghĩa

- Một khai báo (declaration) chỉ xác định tên và kiểu dữ liệu. Nhiệm vụ của khai báo là cung cấp thông tin cho trình biên dịch, nó không yêu cầu trình biên dịch làm bất cứ việc gì.
- Trái lại, một định nghĩa (definition) yêu cầu trình biên dịch phải cấp phát bộ nhớ cho biến.
- Trong một số trường hợp khai báo cũng yêu cầu trình biên dịch cấp phát bộ nhớ, chẳng hạn như khai báo biến. Tuy nhiên, với định nghĩa thì trong bất kỳ trường hợp nào cũng yêu cầu cấp phát bộ nhớ.



2. Thời gian tồn tại và phạm vi hoạt động của các loại biến

- ✧ Các loại biến có hai đặc tính chính là phạm vi hoạt động và thời gian tồn tại. Phạm vi hoạt động liên quan đến phần chương trình nào có thể truy nhập (sử dụng) biến. Thời gian tồn tại là khoảng thời gian trong đó biến tồn tại. Phạm vi hoạt động của biến có thể là trong một lớp, một hàm, một file hay một số file. Thời gian tồn tại của một biến có thể trùng với một đối tượng, một hàm hay toàn bộ chương trình.
- ✧ Có các loại biến sau: biến tự động, biến thanh ghi, biến trong khối lệnh, biến ngoài, biến tĩnh và đối tượng.

a) Các biến tự động (automatic variable)

- Các biến tự động là các biến được khai báo trong một hàm. Sở dĩ gọi chúng là các biến tự động bởi vì chúng được tự động tạo khi hàm được gọi và bị hủy khi hàm kết thúc.
  - Biến tự động có phạm vi hoạt động trong một hàm. Do đó, một biến i được khai báo trong một hàm hoàn toàn khác với một biến i được khai báo trong một hàm khác.
  - Mặc định các biến tự động không được khởi tạo, bởi vậy ngay sau khi chúng được hình thành chúng sẽ có một giá trị vô nghĩa.

b) Các biến thanh ghi (register variable)

- ✧ Biến thanh ghi là một loại biến tự động đặc biệt. Nó được đặt trong các thanh ghi của CPU chứ không phải trong bộ nhớ. Việc truy nhập các biến thanh ghi nhanh hơn các biến thông thường. Biến thanh ghi có lợi nhất khi được dùng làm biến điều khiển cho lệnh lặp bên trong nhất trong các lệnh lặp lồng nhau. Ta chỉ nên dùng một đến hai biến thanh ghi trong một hàm.
- ✧ Để khai báo biến thanh ghi ta dùng từ khóa register trước khai báo biến thông thường.

  Ví dụ: register int a;

c) Các biến trong khối lệnh

- Các biến tự động có thể được khai báo ở bất kỳ đâu trong một hàm hoặc trong một khối lệnh. Khối lệnh là phần chương trình nằm giữa hai dấu ngoặc { và }, chẳng hạn như thân lệnh if hay thân lệnh lặp. Các biến được khai báo trong một khối lệnh có phạm vi hoạt động chỉ trong khối lệnh đó.

d) Các biến ngoài (external variable)

- Các biến ngoài là các biến được khai báo ở bên ngoài tất cả các hàm và các lớp. Các biến ngoài có phạm vi hoạt động từ vị trí khai báo đến cuối file khai báo chúng. Thời gian tồn tại của các biến ngoài là thời gian tồn tại của chương trình, tức là khi chương trình kết thúc thì các biến ngoài mới bị hủy. Khác với các biến tự động, các biến ngoài được tự động khởi tạo bằng 0 nếu ta không khởi tạo.

d) Các biến ngoài (*tiếp*)

```
//Bat dau file
int a;  //a la bien ngoai
.....
class aclass
  {
    .......
  };
void afunc();
.......
//Cuoi file
```

d) Các biến ngoài (*tiếp*)

- ✧ Nếu chương trình được chia thành nhiều file thì các biến ngoài chỉ có thể dùng được trong file khai báo chúng, không dùng được trong các file khác. Để các file khác có thể sử dụng một biến ngoài đã được định nghĩa ở một file ta phải khai báo biến đó dùng từ khóa extern.
- ✧ Để các biến ngoài chỉ truy nhập được trong file khai báo chúng, không truy nhập được từ file khác ta dùng từ khóa static. Trong ngữ cảnh này, từ khóa static có nghĩa là hạn chế phạm vi hoạt động của biến.

  Ví dụ: (*trang sau*)

d) Các biến ngoài (*tiếp*)

Ví dụ 1: Truy nhập biến ngoài trên nhiều file

```
//Bat dau file 1
int a;    //a la bien ngoai
//Cuoi file 1

//Bat dau file 2
extern int a;     //khai bao su dung bien ngoai a o file 1
          //Trong file 2 co the truy nhap bien a
//Cuoi file 2

//Bat dau file 3
//Khong khai bao su dung bien ngoai a nen trong file 3
// khong the truy nhap bien a
//Cuoi file 3
```

d) Các biến ngoài (*tiếp*)

Ví dụ 2: Hạn chế việc truy nhập biến ngoài

```
//Bat dau file 1
static int a;   //dinh nghia bien ngoai a
                //bien a chi truy nhap duoc trong file nay
//Cuoi file 1

//Bat dau file 2
extern int a; //Khong dung duoc khai bao nay
//Cuoi file 2
```

d) Các biến ngoài (*tiếp*)

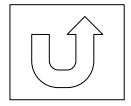
- Có hai vấn đề khi sử dụng biến ngoài:
  - Vì biến ngoài có thể truy nhập được từ bất kỳ hàm nào trong chương trình nên rất dễ bị thay đổi làm mất dữ liệu.
  - Vì các biến ngoài có phạm vi hoạt động ở mọi nơi trong chương trình nên ta phải quan tâm đến vấn đề kiểm soát tên biến để sao cho không có hai biến nào trùng tên.

e) Các biến tĩnh cục bộ (local static)

- Các biến tĩnh cục bộ được sử dụng khi ta muốn duy trì giá trị của một biến khai báo trong hàm giữa các lời gọi hàm. Tức là khi hàm kết thúc biến tĩnh vẫn còn và vẫn chứa giá trị, khi hàm được gọi lần 2 lại có thể sử dụng giá trị này. Phạm vi hoạt động của biến tĩnh cục bộ là trong hàm nhưng thời gian tồn tại của nó là suốt thời gian chương trình chạy.
- *Trong lập trình hướng đối tượng người ta không dùng biến tĩnh một mình mà dùng trong lớp đối tượng.*

f) Đối tượng

- Đối tượng được C++ đối xử như các biến. Các đối tượng có thể được tạo ở dạng biến tự động, biến ngoài,… những không tạo được ở dạng biến thanh ghi.
- Phạm vi hoạt động của thành viên dữ liệu riêng (được khai báo private) của lớp đối tượng là chỉ trong các hàm thành viên của lớp. Còn phạm vi hoạt động của các hàm thành viên (được khai báo public) là tất cả các hàm và các lớp trong chương trình.
- Thời gian tồn tại của dữ liệu riêng là (dù là private hay public) là thời gian tồn tại của đối tượng.



Bài tập chương 5

Bài 1. Viết chương trình nhập vào 2 số phức. Tính tổng, hiệu và tích của 2 số phức đã nhập. Yêu cầu sử dụng các toán tử +, -, * cho số phức.

Bài 2. Viết chương trình sử dụng đối tượng ngăn xếp để đưa ra màn hình số nhị phân của một số nguyên dương n.

Bài tập chương 14

Bài 3. Viết chương trình sử dụng đối tượng ngăn xếp để tìm và đưa ra màn hình tất cả các số nguyên tố nhỏ hơn một số nguyên dương n nhập vào từ bàn phím theo thứ tự giảm dần.

Bài 4. Viết chương trình nhập vào danh sách n tên. Sắp xếp danh sách tên theo vần ABC. Sử dụng đối tượng xâu tự tạo để chứa tên.