BÀI 13: KIỂU XÂU KÝ TỰ

I. Kiểu xâu ký tự trong C (C-string)

1. Khái niệm

Là kiểu dữ liệu dùng để lưu một dãy các ký tự. Trong C có thể sử dụng kiểu mảng hay con trỏ kiểu char.

2. Khai báo

Có hai cách khai báo

a) Khai báo bằng mảng một chiều kiểu char

Cú pháp: **char tên_biến[kích thước];**

VD 1: `char st[20];`

Có thể khởi tạo giá trị cho chuỗi như các VD sau:

```
char str1[] = {'H', 'l','l','o'};
```

```
char str2[] = "Hello";
```

b) Khai báo bằng con trỏ kiểu char

Cú pháp: **char *<Tên_biến>;**

VD 2:

```
char *st;
```

Có thể khởi tạo giá trị cho chuỗi dạng con trỏ như VD sau:

```
char *st = "Hello";
```

VD 3: Xét đoạn chương trình sau:

```
void main(){
   char a[10];
   char *s1 = "10 spaces";
   char *s2;
   a= "Not OK";   //lỗi
   a[5] = 'A';
   s1[5]='B';
   s1="OK";
   *s2="Not OK";  //lỗi
   s2="OK";
   return 0;
}
```

3. Một số hàm trong C-string

a) Hàm gán chuỗi

Cú pháp:

```
char * strcpy(char * s1, const char * s2);
```

Chức năng: gán giá trị của s2 cho s1. độ dài s2 phải ≥ độ dài s1.

VD:

```
char *ss, *st;
ss=”hello”;
strcpy(st,ss);
```

b) Hàm so sánh chuỗi

```
int strcmp(const char s1, const char s2);
```

Chức năng: hàm trả về giá trị là số nguyên:

>0 : nếu s1 > s2

= 0 : nếu s1 = s2

< 0 : nếu s1<s2.

Một số hàm so sánh tương tự:

int strncmp(const char s1, const char s2, int n)

Có chức năng tương tự nhưng **strcmp()** chỉ so sánh n ký tự đầu.

Int stricmp(const char s1, const char s2)

Có chức năng tương tự nhưng **strcmp()** nhưng so sánh không phân biệt chữ hoa, hay chữ thường.

int strincmp(const char s1, const char s2, int n)

Có chức năng tương tự nhưng **strncmp()** nhưng so sánh không phân biệt chữ hoa, hay chữ thường.

c) Hàm tính chiều dài chuỗi

int strlen(const char s)

Hàm trả về độ dài chuỗi, không kể ký tự NULL

d) Hàm nối hai chuỗi

char * strcat(char * s1, const char * s2)

Bổ sung chuỗi s2 vào cuối chuỗi s1

e) Hàm tìm ký tự trong chuỗi

Char * strchr(char *s, int c)

Hàm trả về con trỏ, trỏ đến ký tự c đầu tiên xuất hiện trong xâu s. Nếu không tìm thấy c thì hàm trỏ vào ký tự NULL.

f) Hàm tìm chuỗi

char * strstr(char * s1, char * s2)

Hàm tìm vị trí xuất hiện đầu tiên của s2 trong s1. nếu không tìm thấy hàm trả về NULL.

II. Lớp string trong C++

Thư viện chuẩn STL (Standar Template Library) cung cấp một kiểu lớp **string** hỗ trợ tất cả hoạt động liên quan tới chuỗi và bổ sung thêm nhiều tính năng mới rất thuận tiện cho người lập trình.

1. Khai báo

Khai báo thư viện string

```
#include <string>
```

Khai báo biến string

```
string <biến xâu>;
```

Trong khai báo chúng ta cũng có thể khởi gán giá trị ban đầu cho biến xâu như những loại biến kiểu đơn giản khác.

VD:

```
string s = "Hello";
```

Với biến kiểu string, ta có thể sử dụng toán tử gán (=) để gán giá trị cho một biến xâu kiểu string.

2. Một số thao tác trên biến kiểu string

a. Nhập xuất chuỗi kiểu string

Đối với biến string tring C++ ta có thể sử dụng `cin >>, cout <<` để thực hiện thao tác nhập xuất.

VD:

```
string s;
cin >> s;
```

```
cout <<s;
```

Chú ý: Khi nhập chuỗi có chứa dấu cách chúng ta phải dùng hàm getline();

VD:

```
string s;
getline(cin, s);
```

Nếu dùng `cin >> s;` thì khi nhập xâu có dấu cách, phần xâu sau dấu cách đầu tiên sẽ bị mất.

b. Chuyển đổi biến string sang C-String

VD:

```
std::string s = "ABC"
char * s1 = s.c_str();
```

c. Các phép toán nối chuỗi

Đối với kiểu string chúng ta có thể dùng các phép toán +, += để nối hai hay nhiều chuỗi

VD:

```
string s="";
s="ABC";
```

`s=s+"DEF";` hoặc có thể viết `s+="DEF";`

d. Các phép toán so sánh hai chuỗi

Để so sánh hai chuỗi chúng ta có thể sử dụng các phép toán so sánh: >, <, ==, >=, <=, != hoặc hàm strcmp().

VD:

```
string s1="ABCD", s2="ABDE";
cout << (s1>s2);    kết quả cho 0 (false)
cout << (s1<s2)     kết quả cho 1 (true)
```

Chú ý việc so sánh hai xâu, chương trình dựa vào thứ tự từ điển để so sánh. Xâu nào có thứ tự từ điển đứng trước sẽ cho kết quả nhỏ hơn.

e. Truy cập một ký tự trong chuỗi

Để truy cập đến một ký tự có vị trí *v* (*v*=0..lenth()-1)trong chuỗi ta ghi <tên biến chuỗi>[*v*] hoặc dùng phương thức at: <tên biến chuỗi>.at(*v*).

VD:

```
string s=”ABCD”;
```

```
cout << s[2]; cho kết quả là ký tự ‘C’.
```

Tương tự đối với lệnh `cout << s.at(2);`

f. Lấy độ dài của một biến chuỗi

Để lấy độ dài của một chuỗi ta sử dụng phương thức: lengh()

VD:

```
string s=”ABCD”;
```

```
cout << s.length();
```
Cho kết quả là 4

g. Lấy xâu con trong một xâu

Để lấy một xâu con trong một xâu, ta sử dụng phương thức **substr(int pos, int num)**. Trong đó **pos** là vị trí ký tự cần lấy và **num** là số ký tự cần lấy.

VD:

```
string s=”ABCDEF”;
```

```
cout << s.substr(1,3);
```
cho kết quả là “BCD”.

h. Chèn một chuỗi vào một chuỗi khác

Để chèn một chuỗi con vào một chuỗi khác, ta sử dụng phương thức insert(). Có ba dạng insert() như sau:

`s.insert(inp pos, char * str);` chèn chuỗi (C-string str) vào vị trí pos trong chuỗi s.

`s.insert(inp pos, string str);` chèn chuỗi (string str) vào vị trí pos trong chuỗi s.

`s.insert(int pos, int n, char ch);` chèn n ký tự ch vào chuỗi s tại vị trí pos.

i. Xóa một dãy các ký tự liên tiếp trong xâu

Để xóa một một dãy các ký tự liên tiếp trong xâu, ta sử dụng phương thức erase(). Có hai dạng như sau:

`s.erase(int pos, int n);` xóa n ký tự trong xâu s, bắt đầu từ ký tự ở vị trí pos.

`s.erase(int pos);` xóa các ký tự từ vị trí pos đến cuối xâu.

j. Tìm kiếm xâu con trong một xâu

Để tìm kiếm một xâu con trong xâu, ta sử dụng phương thức find(). Có ba dạng như sau:

`s.find(int ch, int pos = 0);` tìm ký tự ch kể từ pos đến cuối.

`s.find(char *s, int pos = 0);` tìm chuỗi kiểu char* từ pos đến cuối.

`s.find(string& s, int pos = 0);` tìm chuỗi s kể từ pos đến cuối.

Phương thức find(…) trả về số nguyên không chỉ vị trí xâu con được. Nếu không tìn thầy hàm trả về -1.

k. Thay thế một xâu con trong một xâu

Để thay thế một xâu con trong xâu s, ta sử dụng phương thức replace(). Có dạng như sau:

```
s.replace(int pos, int n, string st);
```

Thay thế n ký tự bắt đầu từ vị trí pos trong xâu s bằng xâu st

VD:

Đoạn chương trình sau sẽ biến đổi xâu s = “ABC XYZ 123” thành chuỗi s=”ABC HELLO 123”.

```
#include <iostream>
#include <cstdio>
#include <string>
using namespace std;
string s;
int main(){
     s = "ABC XYZ 123";
     s.replace(4,3,"HELLO");
     cout << s;     //Kết quả sẽ in ra “ABC HELLO 123”
     return 0;
}
```

h. Hàm chuyển đổi một số sang xâu số và ngược lại

Hàm đổi xâu số sang số

```
     stoi(string sn);
     atoi(char * sn);
```

Hàm đổi một số sang xâu số

```
     to_string(n);
```

Trong đó n có thể là kiểu số nguyên, số thực.

3. Ví dụ và bài tập

Bài 1: Nhập xâu ký tự bất kỳ và một ký tự bất kỳ.

a. Đếm số lần xuất hiện của ký tự đó trong xâu

b. Liệt kê số lần của tất cả các ký tự trong xâu.

```
#include <iostream>
#include <cstdio>
#include <string>
```

```
using namespace std;
string s;
unsigned char ch;
int dem=0;
int A[256]={0};
int main(){
     getline(cin,s);
     cin >> ch;
     for (int i=0;i<s.length();i++){
          if (s[i]==ch) dem+=1;
          A[s[i]]+=1;
     }
     //Kết quả câu a
     cout << ch << " xuat hien " << dem << " lan" << endl;
     //Kết quả câu b
     for (int i=0;i<256;i++){
          if (A[i]>0){
               cout << char(i) << " xuat hien " << A[i] << " lan";
          }
     }
     return 0;
}
```

Bài 2: Nhập xâu bất kỳ, biến đổi các chữ cái thường trong xâu thành chữ cái hoa

```
#include <iostream>
#include <cstdio>
#include <string>
using namespace std;

string s;
int main(){
     cout << "Nhap xau: ";
     getline(cin,s);
     for (int i=0;i<s.length();i++){
          if   ((int)s[i]   >=   97   &&   int(s[i])<=122)   s[i]=
char(int(s[i])-32);
     }
     cout << s;
     return 0;
}
```

Bài 3: Nhập xâu bất kỳ, biến đổi các chữ cái hoa trong xâu thành chữ cái thường.

```
#include <iostream>
#include <cstdio>
#include <string>
using namespace std;
string s;
int main(){
     cout << "Nhap xau: ";
```

```
     getline(cin,s);
     for (int i=0;i<s.length();i++){
          if   ((int)s[i]   >=   65   &&   int(s[i])<=97)   s[i]=
char(int(s[i])+32);
     }
     cout << s;
     return 0;
}
```

Bài 4: Nhập xâu bất kỳ, và in ra xâu đó theo thứ tự đảo ngược của các ký tự trong xâu.

```
#include <iostream>
#include <cstdio>
#include <string>
#include <algorithm>
using namespace std;

string s;
int main(){
     cout << "Nhap xau: ";
     getline(cin,s);
     int i,n=s.length()/2;
     for (i=0; i<n; i++){
          unsigned char tam = s[i];
          s[i]=s[s.length()-i-1];
          s[s.length()-i-1]=tam;
     }
     cout << s;
     return 0;
}
```

Bài 5: Nhập xâu bất kỳ và kiểm tra xem xâu ký tự có phải là xâu đối xứng.

```
#include <iostream>
#include <cstdio>
#include <string>
#include <algorithm>
using namespace std;

string s;
bool kiemtra(string a){
     int i,n=s.length()/2;
     for (i=0; i<n; i++){
          if (s[i] != s[s.length()-i-1]) return false;
     }
     return true;
}
int main(){
     cout << "Nhap xau: ";
     getline(cin,s);
     if (kiemtra(s)) cout << "xau doi xung";
```

```
     else cout << "xau khong doi xung";
     return 0;
}
```

Bài 6: Nhập xâu S1 gồm các ký tự bất kỳ. Tạo xâu S2 gồm các ký tự số trong xâu S1 (giữ nguyên thứ tự)

```
#include <iostream>
#include <cstdio>
#include <string>
using namespace std;
string s1,s2="";
int main(){
     getline(cin,s1);
     for (int i=0; i< s1.length();i++){
          if (s1[i]>=40 && s1[i]<=57) s2=s2+s1.substr(i,1);
     }
     cout << s2;
     return 0;
}
```

BÀI TẬP

Bài 7: Một từ là một dãy các ký tự được viết liền nhau không có dấu cách. Hãy viết chương trình thực hiện:

a. Nhập xâu bất kỳ rồi chuẩn hóa xâu sao cho giữa các từ cách nhau đúng một dấu cách, đầu và cuối xâu không có dấu cách.

b. Đếm số từ có trong xâu.

Bài 8: Một xâu ký tự nếu có nhiều ký tự giống nhau được viết liền nhau thì người ta nén bằng cách thay bằng số lượng ký tự và kế tiếp là ký tự đó. Ví dụ xâu “ABBCDDDE” được nén thành “A2BC3DE”. Hãy viết chương trình nhập một xâu rồi nén xâu và in ra màn hình.

Bài 9: Viết chương trình giải nén một xâu ký tự đã được nén ở bài 3.

Bài 10: Nhập một xâu có từ “ANH” và từ “EM”. Hãy tìm cách biến đổi từ “ANH” thành “EM” và ngược lại rồi in xâu ra màn hình.

Ví dụ: xâu “ANH CHO EM TAI CONG VIEN” biến đổi thành “EM CHO ANH TAI CONG VIEN”.