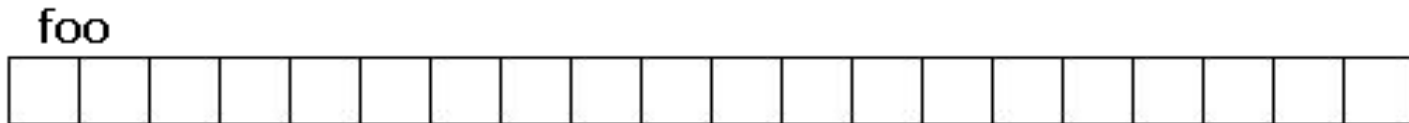


Mảng ký tự

Nguyễn Đức Thắng

Mảng ký tự

- Mảng ký tự (**C-style string**) là mảng một chiều mà kiểu dữ liệu của tất cả các phần tử trong mảng đều là ký tự.
- C++ hỗ trợ lớp **string** để làm việc với ký tự một cách hiệu quả.
- Cách khai báo: `char foo[20];`



Gán mảng ký tự

- Gán phần tử cho mảng ký tự:

```
foo[0] = 'H';  
foo[1] = 'e';  
foo[2] = 'l';  
foo[3] = 'l';  
foo[4] = 'o';
```



Xuất mảng ký tự

- Để xuất mảng ký tự, chúng ta **không cần dùng vòng lặp for** để xuất mảng ký tự mà chỉ cần gọi hàm **cout**.

Ví dụ: `cout<<foo;`

Kết quả: ???

- Cần thêm ký tự kết thúc mảng **'\0'**



Các cách khai báo mảng ký tự khác

- Khai báo nhưng không khởi tạo: `char foo[20];`

- Khai báo và khởi tạo như mảng 1 chiều bình thường (chú ý ký tự kết thúc mảng):

```
char foo[] = { 'L', 'e', ' ', 'T', 'r', 'a', 'n', ' ', 'D', 'a', 't', '\0' };
```

- Khai báo và khởi tạo bằng 1 chuỗi ký tự: (chương trình tự cấp phát bộ nhớ, tự thêm ký tự \0 vào cuối mảng)

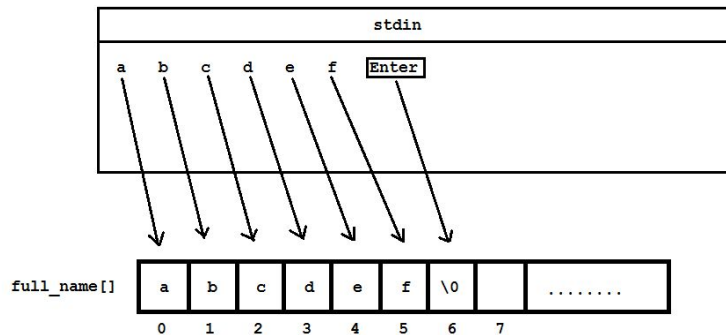
```
char foo[] = "Nguyen Duc Thang";
```



Nhập dữ liệu cho mảng ký tự

- Cũng tương tự như xuất mảng ký tự, chúng ta không cần dùng vòng for, có thể dùng **cin** để nhập dữ liệu cho mảng ký tự.

```
char full_name[50];  
  
cout << "Enter your full name: ";  
cin >> full_name;  
cout << "Your full name is " << full_name << endl;
```



- Vấn đề khi nhập chuỗi: Nguyen Duc Thang ???

Nhập dữ liệu cho mảng ký tự

```
char full_name[50];

cout << "Enter your full name: ";

cin.getline(full_name, 50);

cout << "Your full name is " << full_name << endl;
```

```
char full_name[50];

cout << "Enter your full name: ";

cin.getline(full_name, 50, '\n');

cout << "Your full name is " << full_name << endl;
```

Nhập dữ liệu cho mảng ký tự

- Hàm **gets** (trong visual studio theo chuẩn C++ 11 trở lên thì là **gets_s**) để nhập dữ liệu tương tự như **getline**.

- Sử dụng:

```
char *gets_s( char *str );
```


```
char *gets_s( char *str, rsize_t n );
```

- Ví dụ:

```
char full_name[50];
```

```
gets_s(full_name);
```

```
gets_s(full_name, 20);
```



Các thao tác với mảng ký tự

- Thư viện **cstring** định nghĩa một tập các hàm xử lý với mảng ký tự, giúp chúng ta tiết kiệm thời gian và công sức viết code hơn.
- Tham khảo thêm: <https://www.cplusplus.com/reference/cstring/>



Thiết lập giá trị cho vùng nhớ thuộc mảng ký tự

- Hàm **memset**:

```
void* memset( void *ptr, int value, size_t num);
```

- Thiết lập **num** bytes ô nhớ đầu tiên bắt đầu từ địa chỉ **ptr** bằng giá trị **value**.

```
// Thiết lập 7 byte đầu là ký tự '-'  
char foo[] = "Almost every programmer should know memset!";  
cout << foo << endl;  
memset(foo, '-', 7);  
cout << foo << endl;
```

```
// Dùng hàm memset để khởi tạo toàn bộ giá trị mảng  
char foo[20];  
memset(foo, 'a', sizeof(foo));  
foo[sizeof(foo) - 1] = '\0';
```

Xem độ dài chuỗi ký tự

- Độ dài chuỗi ký tự được tính từ ký tự đầu tiên cho đến ký tự kết thúc chuỗi.

```
size_t strlen ( const char * str );
```

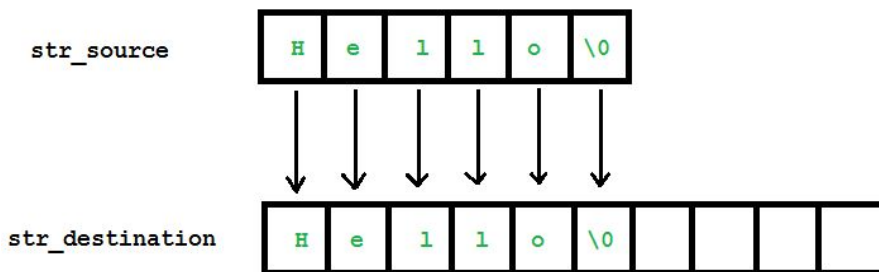
- Ví dụ:

```
char foo[50] = "C++ Programming language";  
cout << "Length of foo string: " << strlen(foo) << endl;
```



Sao chép mảng ký tự

- Có 2 cách để sao chép chuỗi ký tự từ một mảng sang mảng khác: dùng **strcpy** và **strncpy**
- Hàm **strcpy**: `char * strcpy (char * destination, const char * source);`
- Mảng đích phải được cấp phát đủ bộ nhớ.



Sao chép mảng ký tự

- Hàm **strncpy**: (trong visual là **strncpy_s**)

```
char * strncpy ( char * destination, const char * source, size_t num );
```

- Sao chép **num** ký tự từ mảng **source** sang mảng **destination**

```
char str_source[] = "This is source string";  
char str_destination[30];  
  
strncpy(str_destination, str_source, strlen(str_source) / 2);
```



So sánh 2 chuỗi ký tự

- Sử dụng hàm **strcmp**: `int strcmp (const char * str1, const char * str2);`
- Trả về:
 - 0: Nội dung 2 chuỗi ký tự giống nhau
 - Nhỏ hơn 0: $str1 < str2$ (so sánh theo ký tự bảng ASCII)
 - Lớn hơn 0: $str1 > str2$ (so sánh theo ký tự bảng ASCII)
- Hàm **strncmp**: So sánh n ký tự đầu tiên của 2 chuỗi
- Hàm **stricmp**: tương tự strcmp nhưng không phân biệt hoa thường.
- Tương tự với **strnicmp**

Nối chuỗi ký tự

- Sử dụng hàm **strcat**: `char * strcat (char * destination, const char * source);`
- Nối vào sau chuỗi **destination** một bản copy chuỗi **source**.
- Mảng **destination** phải được cấp phát đủ bộ nhớ.
- Trong C++ 11 trở lên là **strcat_s**
- Hàm **strncat**: Nối n ký tự đầu tiên của chuỗi 2 vào chuỗi 1 - `strncat(s1, s2, 5)`



Tìm kiếm chuỗi ký tự trong chuỗi ký tự khác

- Để thực hiện tìm kiếm chuỗi **pattern** bên trong chuỗi **text** nào đó, ta sử dụng hàm

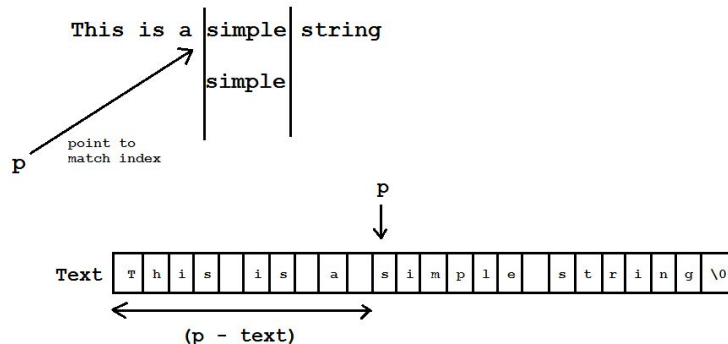
strstr: `const char * strstr (const char * text, const char * pattern);`

- Nếu không tìm thấy, trả về giá trị **NULL**. Hàm sẽ trả về **địa chỉ ô nhớ** của mảng ký tự **text** mà hàm tìm thấy sự trùng khớp với chuỗi ký tự **pattern**.

```
char text[] = "This is a simple string";
char pattern[] = "simple";

char *p = strstr(text, pattern);

if (p == NULL) {
    cout << "Could not find the pattern string in the text string"
    << endl;
}
else {
    int32_t match_index = (p - text) / sizeof(char);
    cout << "The pattern string match the text string at: " <<
    match_index << endl;
}
```



Tìm lần xuất hiện đầu tiên ký tự c trong chuỗi s

- Hàm **strchr**: `char *strchr(char s[], char c);`

- Trả về:

- Null: nếu không tìm thấy
- Địa chỉ c: nếu tìm thấy

```
char s[15];  
char *ptr, c = 'm';  
strcpy(s, "Vi du tim ky tu");  
ptr = strchr(s, c);  
if (ptr)  
    printf("Ky tu %c tai: %d", c, ptr-s);  
else  
    printf("Khong tim thay");
```

Tìm kiếm chuỗi xuất hiện trong chuỗi

- Hàm **strstr**: `char *strstr(char s1[], char s2[]);`

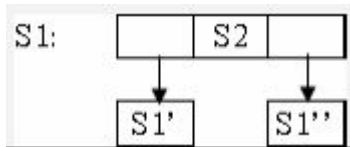
```
char *s1 = "Borland International";  
char *s2 = "nation", *ptr;  
ptr = strstr(s1, s2);  
printf("Chuoi con: %s\n", ptr);
```



Tách chuỗi

- Hàm **strtok**: `char *strtok(char s1[], char s2[]);`

- Nếu s2 có xuất hiện trong s1: Tách chuỗi s1 thành hai chuỗi: Chuỗi đầu là những ký tự cho đến khi gặp chuỗi s2 đầu tiên, chuỗi sau là những ký tự còn lại của s1 sau khi đã bỏ đi chuỗi s2 xuất hiện trong s1. Nếu s2 không xuất hiện trong chuỗi s1 thì kết quả chuỗi tách vẫn là s1.



```
char input[16] = "abc,d";
char *p;
// Lay chuỗi đầu
p = strtok(input, ",");
if (p) printf("S11: %s\n", p);
/*Lay chuỗi còn lại, tham số đầu là NULL*/
p = strtok(NULL, ",");
if (p) printf("S12: %s\n", p);
```

Bài tập

1. Viết chương trình nhập vào một chuỗi ký tự từ bàn phím, chuyển tất cả các ký tự trong chuỗi thành định dạng in hoa (hàm **toupper**).
2. Viết chương trình nhập vào một chuỗi ký tự **str** và một ký tự **ch** từ bàn phím, đếm trong chuỗi ký tự **str** có bao nhiêu lần xuất hiện ký tự **ch** mà bạn vừa nhập.



Thư viện string trong C++

- Với Cstring, chúng ta phải quan tâm xem khai báo bao nhiêu ô nhớ cho đủ, nhiều vấn đề khác.
- **string** là lớp chuẩn mô tả về chuỗi ký tự, cung cấp khả năng xử lý và lưu trữ chuỗi ký tự, có thể thay đổi kích thước vùng nhớ linh hoạt.

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

//.....
string my_string;
```



Các cách khai báo string

```
string empty_string();
```

```
string my_string = "Learning C++ is easy";
```

```
string another_string("Don't give it up");
```

```
string temp_string("What the hell is going on?");
```

```
string one_more = temp_string;
```

```
string it_just_began(temp_string);
```



Nhập và xuất string

- Xuất: dùng hàm **cout**
- Nhập: dùng hàm **cin**, **getline(cin, s)**, **getline(cin, s, '!')**



Các phương thức với string

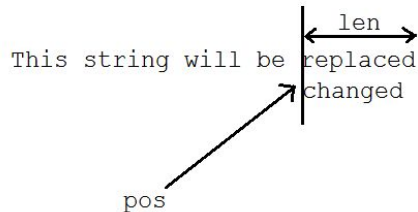
- Tính độ dài chuỗi ký tự: `s.length()`, `s.size()`
- Kiểm tra string rỗng hay không: `s.empty()`
- Xóa dữ liệu string: `s.clear()`
- Truy cập phần tử trong string: `s[1]`, `s.at(2)`
- Truy xuất nhanh phần tử đầu tiên và cuối cùng: `s.front()`, `s.back()`
- Nối thêm 1 ký tự vào string: `s.push_back('')`



Các phương thức với string

- Xóa phần tử cuối cùng của string: `s.pop_back()`
- Nối chuỗi ký tự sau string: `s1.append(s2)`, `s1 += s2`
- Chèn 1 string vào vị trí bất kỳ trong string: `s1.insert(0, s2)`
- Thay thế một phần string: `string& replace (size_t pos, size_t len, const string& str);`

```
string my_string = "This string will be replaced";  
cout << my_string << endl;  
  
my_string.replace(20, 8, "changed");  
cout << my_string << endl;
```



Các phương thức với string

- Tìm kiếm: `size_t find (const string& str, size_t pos = 0) const;`

```
string name;
cout << "Enter a name: ";
getline(cin, name);

int32_t search_index = name_list.find(name);

if(search_index == -1)
    cout << name << " is not exist in name_list" << endl;
else
    cout << "Found at: " << search_index << endl;
```

- Trả về chỉ số mà **name** được tìm thấy trong **name_list**, nếu không tìm thấy trả về -1



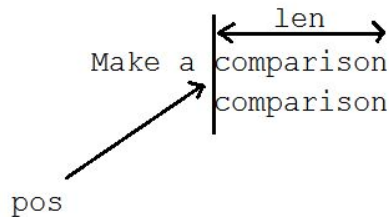
Các phương thức với string

- So sánh: `int compare (const string& str) const;`

```
string str1 = "This is a string";
string str2 = "This is a string";

if(str1.compare(str2) == 0)    {
    cout << "str1 and str2 are equal" << endl;
}
else {
    cout << "str1 and str2 are not equal" << endl;
}
```

```
// so sánh từ vị trí số 7, lấy ra 10 ký tự liên tiếp
nhau
string my_string = "Make a comparison";
int comparison = my_string.compare(7, 10,
"comparison");
cout << "Result of the comparison: " << comparison
<< endl;
```





Thank you!