



CSE/CS102

计算机组成与系统



第 0 讲

From C++ To C

- 薛浩
- stickmind.com

抽象数据类型

- C++ 的底层涉及到很多 C 的内容，用好 C 可以帮助我们更好地理解 C++ 中的一些概念
- 在深入数据表示的内部细节之前，我们先以**抽象数据类型**观点来学习 C 程序

```
typedef char* string;
```

```
typedef FILE* stream;
```

- 以上将字符指针/文件指针定义成了 string 和 stream 两个数据类型
- 以独立的逻辑单元来处理字符串和文件，关注抽象层的行为，避免沉溺于细节

字符串

- 字符串在 C 中就是一个字符序列（数组）
- 接口 `strlib.h` 提供了一个抽象层，可以在避免考虑细节的情况下处理字符串
- 接口 `strlib.h` 本质是对 `string.h` 的封装，`string.h` 是 C 标准库接口
 - 用好 `string.h` 需要关注很多细节和其他概念，对初学者不太友好
 - 后续课程会专门介绍 `string.h` 的用法

`strlib.h`

`string.h`

语言级

机器级

Ex1. 获取字符串并打印

- 接口 `genlib.h` 定义了 `string` 类型, 此时可以像 `int` 类型一样创建字符串变量
- 接口 `simpio.h` 提供了一些获取用户输入的函数, 获取字符串可以使用 `GetLine` 函数

```
#include "genlib.h"
#include "simpio.h"

int main(int argc, string args[]) {
    printf("Enter a string: ");
    string str = GetLine();
    printf("%s\n", str);
}
```

Ex2. 确定字符串的长度

- 字符串是一个字符序列，其包含的字符总数称为字符串的**长度**（length）
- 使用 `StringLength(str)` 函数调用可以计算出字符串 `str` 的长度
- 例如，`StringLength("test\n")` 的结果是 5，注意 `\n` 是转义字符

```
#include "genlib.h"
#include "simpio.h"
#include "strlib.h"

int main(int argc, string args[]) {
    printf("Enter a string: ");
    string str = GetLine();
    printf("The length is %d.\n", StringLength(str));
}
```

Ex3. 从字符串中索引字符

- 字符串的位置编号从 0 开始，例如 “hello” 的每个字符编号如图所示

h	e	l	l	o
0	1	2	3	4

- 字符串每个字符下方的数字称为字符串的**索引** (index)
- 使用 `IthChar(str, 3)` 可以返回字符串 `str` 索引位置为 3 的字符，以上案例将返回 `'l'`
- 字符的数据类型是 `char`，本质上依然是整型值（有整型特性的类型统称为标量类型）

```
char FirstChar(string str) {  
    return IthChar(str, 0);  
}
```

Ex4. 连接字符串

- 使用 `ConcatString(str1, str2)` 可以将两个字符串首尾相连，这样的操作称为**连接** (concatenation)
- 一个有用的需求是输出一些装饰字符，比如输出一行 * 号可以使用 `ConcatNCopies(10, “*”)`

```
string ConcatNCopies(int n, string str) {  
    string result = "";  
    for (int i = 0; i < n; i++) {  
        result = ConcatString(result, str);  
    }  
    return (result);  
}
```


Ex5. 分割字符串

- 从字符串中提取一部分**子串** (substring) , 可以使用 `SubString(str, p1, p2)` 函数, 该接口将从 `str` 中提取索引区间为 `[p1,p2]` 的子串
- 需要注意一些特殊情况, 有了这些保护措施, 即便不理解字符串底层表示, 也能够放心地操作字符串:
 - 如果 `p1` 为负数, 则自动转为 `0`
 - 如果 `p2` 大于字符串最后一个字符索引, 则自动转为 `StringLength(str) - 1`
 - 如果 `p1` 大于 `p2`, 则返回空字符串

```
string SecondHalf(string str) {  
    int len = StringLength(str);  
    return (SubString(str, len / 2, len - 1));  
}
```

Ex6. 比较字符串

- 字符是标量类型，所以支持比较操作；两个字符串可以根据字母顺序，从左到右依次比较每个字符
- 使用 `StringCompare(str1, str2)` 可以比较两个字符串，并返回一个整数
 - 如果 `str1` 字符在 `str2` 前面，则返回负整数
 - 如果 `str1` 字符在 `str2` 后面，则返回正整数
 - 如果两个字符串完全相等，则返回 0
- 对于判断两个字符串是否相等的情况，还可以直接使用 `StringEqual(str1, str2)`

切记：字符串非标量类型，不能直接进行比较操作

Ex7. 字符串搜索

- 确定一个字符串是否包含一个字符或字符串在程序设计中非常有用，使用 `FindChar` 和 `FindString` 两个函数可以实现这个目的
- `FindChar('w', "hello world", 2)` 该调用将从索引 2 的位置搜索 “hello world”，如果发现字符 ‘a’ 则返回当前索引；否则，返回 -1
- `FindString("wor", "hello world", 0)` 该调用将从索引 0 开始搜索，如果发现 “wor”，则返回首字符索引；否则，返回 -1

Ex8. 字符串赋值操作

- 对于基本类型，比如 `int`，可以将一个变量 `a` 的值直接赋值给另一个变量 `b`

```
int a = 3;
```

```
int b = a;
```

- 对于字符串类型，如果使用同样的操作赋值，将会发生意想不到的问题（后续课程会进行解密）
- 字符串的赋值可以使用 `CopyString(str)` 操作完成

```
string str = "hello";
```

```
string copy = CopyString(str);
```

Ex9. 字符串转换操作

- 大小写转换可以使用 `ConvertToLowerCase(s)` 和 `ConvertToUpperCase(s)`
- 字符转换成字符串可以使用 `CharToString(ch)`
- 整型和字符串的转换可以使用 `IntegerToString(n)` 和 `StringToInteger(s)`
- 浮点型和字符串转换可以使用 `RealToString(d)` 和 `StringToReal(s)`



问题？

抽象数据类型

- C++ 的底层涉及到很多 C 的内容，用好 C 可以帮助我们更好地理解 C++ 中的一些概念
- 在深入数据表示的内部细节之前，我们先以**抽象数据类型**观点来学习 C 程序

```
typedef char* string;
```

```
typedef FILE* stream;
```

- 以上将字符指针/文件指针定义成了 string 和 stream 两个数据类型
- 以独立的逻辑单元来处理字符串和文件，关注抽象层的行为，避免沉溺于细节

文件流

- 文本文件在程序开发中有很多用途，这类文件通常都包含文本，可以将其看作存储在硬盘上的字符序列
- 例如，文本文件 `dialogue.txt` 包含如下文本

```
How are you doing?  
Not bad.
```

- 更准确的说，该文本将以如下形式存储，每一行由转义字符 `\n` 进行换行：

```
How are you doing?\nNot bad.EOF
```

- 特别注意的是，文本文件结尾是一个特殊字符 `EOF`，可以通过比较判断是否到达文件末尾

```
if(ch == EOF) {  
    ...  
}
```

C++ 文件流的使用

```
int main() {  
    string filename = getLine("Input file: ");  
    ifstream infile;  
    infile.open(filename.c_str());  
    if (infile.fail()) {  
        cout << "Cannot open " << filename << endl;  
        return 1;  
    }  
    char ch;  
    while ((ch = infile.get()) != EOF) {  
        cout.put(ch);  
    }  
    infile.close();  
    return 0;  
}
```

C 文件流的使用

```
int main(int argc, string args[]) {  
    printf("Input file name: ");  
    string filename = GetLine();  
    stream infile = fopen(filename, "r");  
    if (infile != NULL) {  
        printf("Can't open the file %s. Try again.\n", filename);  
        return 1;  
    }  
    char ch;  
    while ((ch = getc(infile)) != EOF) {  
        putc(ch, stdout);  
    }  
    fclose(infile);  
    return 0;  
}
```




问题？

TranslateLine

```
void TranslateLine(string line) {
    TokenScanner scanner;
    scanner.setInput(line);
    while (scanner.hasMoreTokens()) {
        string token = scanner.next_token();
        if (IsLegalWord(token))
            token = TranslateWord(token);
        cout << token;
    }
    cout << endl;
}
```

```
void TranslateLine(string line) {
    InitScanner(line);
    while (!AtEndOfLine()) {
        string token = GetNextToken();
        if (IsLegalWord(token))
            word = TranslateWord(token);
        printf("%s", token);
    }
    printf("\n");
}
```



总结

- 不管是 C 还是 C++ 接口，都必须能够实现公共和私有函数/变量的分离
- C++ 通过类实现封装；C 通过文件管理内部信息
- C 模块的内部状态由**全局变量**保存，模块中的所有函数都可以访问全局变量
- 由 .h 接口文件导出的函数/变量属于公共接口，可供用户使用
- 在 .c 实现文件中，使用 `static` 标记全局变量或函数，表明其为该模块私有