第 5 课

I/O, STACK, QUEUE

薛浩

xuehao0618@outlook.com

阅读

- Programming Abstraction in C++ *Chapter 4&5*
- CS106B, Summer Quarter 2022
- C++ Tutorials - cplusplus.com

今日话题

- 程序交互
- Queue
- Stack

程序交互

获取用户输入

```cpp
string name;

cout << "Enter your name: ";
cin >> name;

// or
name = getLine("Enter your name: ")
```

获取用户输入

```
string name;

cout << "Enter your name: ";
cin >> name;

// or
name = getLine("Enter your name: ")
```

获取用户输入

```cpp
string name;

cout << "Enter your name: ";
cin >> name;

// or
name = getLine("Enter your name: ")
```

输出数据

```
string line = getLine("Enter: "); // cin  -> line
cout << line << endl;             // line -> cout
```

输出数据

```
string line = getLine("Enter: "); // cin  -> line
cout << line << endl;             // line -> cout
```

输出数据

```
string line = getLine("Enter: "); // cin  -> line
cout << line << endl;             // line -> cout
```

(**cin**) console	$\rightrightarrows$	data
(**cout**) console	$\leftleftarrows$	data

(cin) console $\rightrightarrows$ data

(**cout**) console $\leftleftarrows$ data

(cin) console	$\rightrightarrows$	data
(cout) console	$\leftleftarrows$	data

(cin) console $\rightrightarrows$ data

(cout) console $\leftleftarrows$ data

cin: console input

(cin) console	⇉	data
(cout) console	⇇	data

cin: console input

cout: console output

WHY NOT

file input → **fin** ??

file output → **fout** ??

WHY NOT

file input → **fin** ??

file output → **fout** ??

WHY NOT

file input → **fin** ??

file output → **fout** ??

HOW-TO

输入输出

- 本质上是一种称为流（stream）的数据结构
- cin 和 cout 是标准流，基于终端交互
- ifstream 和 ofstream 是文件流类，基于文件交互

使用文件流类需要声明文件流对象

```cpp
#include <fstream> // for ifstream/ofstream

ifstream fin("res/input.txt");    // counterpart cin
ofstream fout("res/output.txt"); // counterpart cout
```

使用文件流类需要声明文件流对象

```cpp
#include <fstream> // for ifstream/ofstream

ifstream fin("res/input.txt");    // counterpart cin
ofstream fout("res/output.txt"); // counterpart cout
```

使用文件流类需要声明文件流对象

```cpp
#include <fstream> // for ifstream/ofstream

ifstream fin("res/input.txt");    // counterpart cin
ofstream fout("res/output.txt"); // counterpart cout
```

使用文件流类需要声明文件流对象

```cpp
#include <fstream> // for ifstream/ofstream

ifstream fin("res/input.txt");    // counterpart cin
ofstream fout("res/output.txt"); // counterpart cout
```

使用文件流类需要声明文件流对象

```cpp
#include <fstream> // for ifstream/ofstream

ifstream fin("res/input.txt");    // counterpart cin
ofstream fout("res/output.txt"); // counterpart cout
```

通过文件流输入输出

```cpp
#include <fstream> // for ifstream/ofstream

ifstream fin("res/input.txt");    // counterpart cin
ofstream fout("res/output.txt"); // counterpart cout

string line;
getline(fin, line);    // fin  -> line
fout << line << endl; // line -> fout
```

通过文件流输入输出

```cpp
#include <fstream> // for ifstream/ofstream

ifstream fin("res/input.txt");    // counterpart cin
ofstream fout("res/output.txt"); // counterpart cout

string line;
getline(fin, line);    // fin  -> line
fout << line << endl; // line -> fout
```

通过文件流输入输出

```
#include <fstream> // for ifstream/ofstream

ifstream fin("res/input.txt");   // counterpart cin
ofstream fout("res/output.txt"); // counterpart cout

string line;
getline(fin, line);    // fin  -> line
fout << line << endl; // line -> fout
```

重复输入

```
while (true) {
    // cin  -> line
    line = getLine("Cin: ");
    if (line == "")
        break;
    // line -> cout
    cout << line << endl;
}
```

```
while (true) {
    // fin -> line
    getline(fin, line);
    if(fin.eof() == true)
        break;
    // line -> fout
    fout << line << endl;
}
```

BETTER

```
while (true) {
    if(getline(fin, line).eof() == true)
        break;
    fout << line << endl;
}
```

BETTER

```
while (true) {
    if(!getline(fin, line))
        break;
    fout << line << endl;
}
```

BETTER

```
while (getline(fin, line)) {
    fout << line << endl;
}
```

✏️ 小试牛刀

将 EnglishWords.txt 中所有的回文单词

输出到 palindrome.txt 文件中

今日话题

- ~~程序交互~~
- Queue
- Stack

QUEUE

QUEUE

- 遵循“先来先服务”或“先进先出”的顺序策略
- 常简写为 FIFO（First In First Out）
- 基本操作：入队 enqueue、出队 dequeue

斯坦福 QUEUE

操作示例

```
Queue<char> q;
q.enqueue('a');
q.enqueue('b');
q.enqueue('c');
while (!q.isEmpty()) {
    cout << q.dequeue() << endl;
}
// Output: ???
```

QUEUES AND STACKS

今日话题

- ~~程序交互~~
- ~~Queue~~
- Stack

STACK

STACK

- 遵循“后进先出”的逆序策略
- 常简写为 LIFO（Last In First Out）
- 基本操作：入栈 push、出栈 pop

斯坦福 STACK

操作示例

```cpp
Stack<char> s;
s.push('a');
s.push('b');
s.push('c');
while (!s.isEmpty()) {
    cout << s.pop() << endl;
}
// Output: ???
```

QUEUES AND STACKS

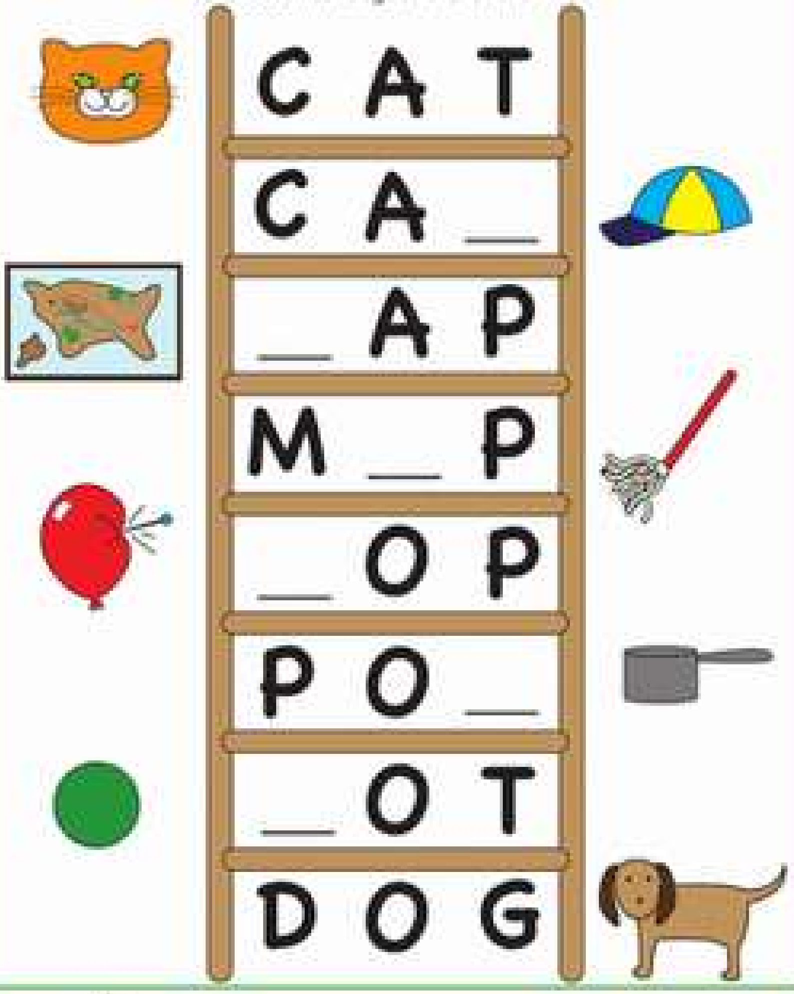
今日话题

- ~~程序交互~~
- ~~Queue~~
- ~~Stack~~

🎉 轻松一刻 🪁

The Word Ladder Challenge

How do you turn a CAT into a DOG? Have some fun with short vowels by completing this word ladder. Fill in the missing letter for each word and watch as "cat" becomes "dog" one letter at a time.



C A T

C A _

_ A P

M _ P

_ O P

P O _

_ O T

D O G

WORD LADDERS

✏️ 小试牛刀

编写一个 WordLadder 小游戏

ANNOUNCEMENT

作业 2 发布

下一次课

- 递归引入

THE END