> 由 KiOii (*_EM_Cpper*)整理。 (KiOii (*_EM_Cpper*) makes this note.)

Statements

Contents

5.1 Simple Statements

expression statement

- *define:* ***expression*** *+* ***;***
- *example:* `ival + 5;`

Null Statements

```
; // null statement
```

Beware of Missing or Extraneous Semicolons

```
while(iter != svec.end()); // extra semiclon(;)
    ++iter;
```

Compound Statement(Blocks)

```
while(cin >> s && s != sought)
{}
```

- *wraning:* **An empty block is equivalent to a null statement.**

5.2 Statement Scope

- *wraning:* we define variables inside the control structure of the if,swith,while,and for statement.
- *code-look:*

```
if(){/*...*/}
while(){/*...*/}
while(1)
    cout << ";"; // while control structure
cout << '\n';// not inside while control structure
//...
```

5.3 Conditional Statements

> C++ provides two statements that allow for conditional execution.

5.3.1 The if Statements

- *if:*

  if (**condition**)

  statement
- *if-else:*

  if (**condition**)

  statement

  else

  statement2
- *wraning:* **either or both statement and statement2 can be a block**

- *wraning:* **condition** can be an **expression** or an **initialized variable declaration**(can convertible to bool)
- *wraning:* each **else** is matched with the **closest** preceding **unmatched if**

5.3.2 The switch Statement

case label

- *wraning:* must be integral constant expressions

```
char ch = getVal();
int val = 42;
switch(ch){
    case 3.14:{}break; // error
    case val:{}break;  // error
    default:{}break;
}
```

- *wraning:* two case labels connot have the same value

5.4 Iterative Statements

> Iterative statements,commonlu called loops

5.4.1 The while Statement

- ***while*** (**condition**)

  statement

5.4.2 Traditional for Statement

- ***for*** (**init-statement; condition; expression**)

  statement

- *init-statement:* **declaration statement、expression statement、null statement**，is executed once only if the condition is true.
- *condition:* **null statement means true**

5.4.3 Range for Statement

- ***for*** (**declaration : expression**)

  statement

  - *expression:* **a sequence**
    - *such as:* **a braced intializer list、an array、an object of a type** such as vector or string that **has begin and end members that return iterators**
    - *declaration:* **auto &var** or **const auto &var** or ...

5.4.4 The do while Statement

- ***do***

  statement

  while **(condition);**

  - *condition:* **cannot be empty**， vatiables used in condition must be **defined outside** the body of the do while statement.

5.5 Jump Statements

> C++ offers four jumps: break,continue,goto,return

5.5.1 The break Statement

- *define:* terminates the nearest enclosing **while, do while, for, switch** statement

5.5.2 The continue Statement

- *define:* **terminates** the current iteration of the nearest enclosing loop and **immediately begins the next iteration.**

5.5.3 The goto Statement

- *define:* jump **from the goto** to another statement **in the same function**
- *use:* ***goto label;***

5.6 try Blocks and Exception Handling

Exception

- *define:* **run-time anomalies**

Exception Handing

- *use:* when program detects a problem that cannot resolve,it **having signaled what happened.**

throw expressions : detecting

- *use:* something it can't handle,we say that a **throw raises an exception**

try blocks : handing

- *use:* **starts with the keyword try** and **ends with one or more catch clauses**.

5.6.1 A throw Expression

- *wraning:* **the type of the expression** determines what kind of exception is thrown [1]

```
if(item.isbn() != item.isbn())
    throw runtime_error("Data must refer to same ISBN");
/*
*   throw an expression that is an object or type runtime_error,the object must be
*   initialized.
*/
```

- *wraning:* Throwing an exception **terminates the current function** and **transfers control to a handler** that will know how to handle this error.
- *runtime_error:* <**stdexcept**>

5.6.2 The try Block

- **try {**

  program-statements

  }catch (exception-declaration){

  handle-statements

  }catch (exception-declaration){

  handle-statements

  }// ...

- *wraning:* [2]

5.6.3 Standard Exceptions

Four Headers

- <**exception**>: class ***exception***
  - only an exception occurred but **provides no additional information**
- <**stdexcept**>:

Table 5.1. Standard Exception Classes Defined in <stdexcept>

exception	The most general kind of problem.
runtime_error	Problem that can be detected only at run time.
range_error	Run-time error: result generated outside the range of values that are meaningful.
overflow_error	Run-time error: computation that overflowed.
underflow_error	Run-time error: computation that underflowed.
logic_error	Error in the logic of the program.
domain_error	Logic error: argument for which no result exists.
invalid_argument	Logic error: inappropriate argument.
length_error	Logic error: attempt to create an object larger than the maximum size for that type.
out_of_range	Logic error: used a value outside the valid range.

- <**new**>: class ***bad_alloc***

- <**type_info**> : class ***bad_cast***
- *wraning:* we can only initialize ***exception,bad_alloc,and bad_cast*** objects.[3]
- *wraning:* The exception types **define only a single operation named what.** That function takes **no arguments** and **returns a const char* that points to a C-style character string**.

---

1. 表达式的类型就是抛出的异常类型↩

2. 当异常被抛出时，首先搜索抛出该异常的函数。如果没有找到匹配的catch语句，终止当前函数，并在调用该函数的函数中继续寻找合适的catch。依次下去。如果最终没有找到合适的catch，则程序转到terminate的标准库函数执行，执行非正常退出。对于没有try语句，但是发生了异常，则系统直接调用terminate函数。↩

3. 对于没有初始值的异常类型来说，what函数的返回内容取决于编译器↩