

第三部分

C 中发展 C++ 的内容 /语法部分/

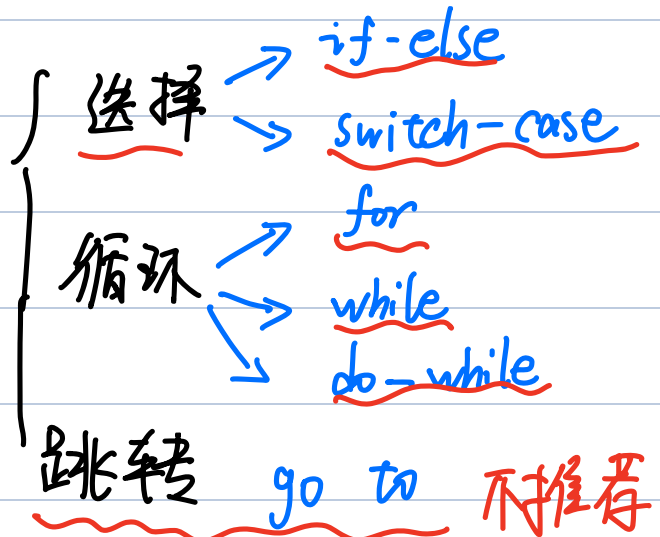
1. 函数定义

C++ 中函数的定义与 C 无异

※ 应注意代码规范性

2. 流程控制

C++ 中与 C 基本无异



※ C++ 中, 可以通过

for (类型 变量名 : 容器名) { }

↓ auto
迭代器 功能
iterator

来循环 只读 遍历 容器。
不读写容器内容

3. 运算符

C++ 中与 C 基本无异

但

单目运算符 增加

双目运算符

基本无异

返回类型为 指针

new

delete

new[]

delete[]

malloc

free

其他运算符

. 成员选择符

* 成员指针选择符

:: 范围指定 指定类或命名空间

sizeof() 大小

4. 数据类型

① 固有类型

int float char ^{空类型}void bool
long/short double 非0T/F为0

② 指针

与C基本无异，指向地址

5. 参数传递

① 值传递

会造成内存冗余

本质是某个变量值复制给函数中的变量进行操作
不改变原变量值。

② 地址传递

是把变量的地址传给函数变量进行操作

会改变原变量值

③ 引用传递 reference 有效防止了野指针 & 指针越界
类似于指针传递, 但 安全性更高
引用传递允许函数对传递来的外部变量进行修改
会改变原变量值。

※ 注意: 类型 & 变量名 指向变量的地址

引用变量 相当于对已有变量的 特殊别名

因此 初始化引用变量时应当给定初始变量

`int & a;`

X

`int & a = 10;`

X

`int & a = A;`

常量也不行

给A别名a

6. 变量作用域

函数/类/全局

与C无异, 一个变量只在其最近的作用域内有效

7. 变量的内存分配

当前线程存活 \Rightarrow 全局变量存活

全局变量

全局变量区域

[与线程的存活相关]

局部变量

系统栈

外部变量

`extern`

只声明和使用, 不额外开空间

调用 同一项目下不同文件的同一变量

常量变量 `constant`

`const` 类型 变量名

也是变量, 但其值不能改变

与之类似的是

内联函数

与宏定义 `#define` [替换] 不同

常量变量可取地址

※也可以在函数参数中应用, 使参数只读调用.

`const` 类型 & 节省空间且只读.

8. 运算符的应用

① 赋值 \Rightarrow 等号 =

② 数学符号 \Rightarrow 加⁺、减⁻、乘^{*}、整除[/]、取余[%]

③ 关系运算 \Rightarrow 大于[>]、小于[<]、等于⁼⁼、不等于^{!=}

④ 逻辑运算 \Rightarrow 与^{&&}、或^{||}、非[!]

⑤ 三元运算 \Rightarrow 条件? 真: 假;

⑥ 一元运算

| | |
|-------------------------|--------------------------------------|
| ^{&} 取地址 | [*] 、 \rightarrow 取值引用 |
| ⁺⁺ 自增 | ⁻⁻ 自减 |
| ^{new} 开内存区域 | ^{delete} 关内存区域 |
| 大小 <code>sizeof</code> | |


```
int foo();  
double goo();  
int hoo(int x);
```

// 给函数指针赋值

```
int (*funcPtr1)() = foo; // 可以
```

```
int (*funcPtr2)() = goo; // 错误! 返回值不匹配!
```

```
double (*funcPtr4)() = goo; // 可以
```

funcPtr1 = hoo; // 错误, 因为参数不匹配, funcPtr1只能指向不含参数的函数, 而hoo含有int型的参

```
int (*funcPtr3)(int) = hoo; // 可以, 所以应该这么写
```

```
int foo(){  
    return 5;  
}  
int main()  
{  
    int (*funcPtr1)() = foo;  
    int (*funcPtr2)() = &foo; // c++会隐式得把foo转换成&foo, 所以你无需再加入&  
    std::cout << funcPtr1() << std::endl;  
    std::cout << funcPtr2() << std::endl;  
}
```

结果:

5

5

↓
通过函数指针调用函数