第三部分　C中发展CPP的内容

/语法部分/

1. 函数定义

CPP中函数的定义与C无异　　注意代码规范性

2. 流程控制

CPP中与C基本无异

- 选择
  - if-else
  - switch-case
- 循环
  - for
  - while
  - do-while
- 跳转　go to　不推荐

注：C++11中，可以通过

for(类型　变量名：容器名){}

类型 → auto → 迭代器功能 iterator

来循环只读遍历容器。

不改写容器内容

3. 运算符

CPP中与C基本无异

但

单目运算符　增加

返回类型为指针

- new
- delete
- new[]
- delete[]

malloc

free

双目运算符　基本无异

其他运算符

. 成员选择符

.* 成员指针选择符

:: 范围指定 指defined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; ✗ int& a=10; ✗ 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⇒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

只声明和使用，不额外开空间

外部变量 extern 调用同一次且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

与宏定义#define[替换]不同

常量变量可取地址

与之类似的是内联函数

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 ⇒ 等号 =

② 数学符号 ⇒ 加(+)、减(-)、乘(*)、整除(/)、取余(%)

③ 关系运算 ⇒ 大于(>)、小于(<)、等于(==)、不等于(!=)

④ 逻辑运算 ⇒ 与(&&) 或(||) 非(!)

⑤ 三元运算 ⇒ 条件？真：假；

⑥ 一元运算 取地址(&) 取值引用(*、->)

自增(++) 自减(--)

开内存区域(new) 关内存区域(delete)

大小(sizeof)

9. 类型转换

cpp中 编译器会 自动进行如下类型转换

int → unsigned → long → unsigned long → double

↑

short, char

↑

float

(类型) 变量名

也可以由人工 进行 强制类型转换

10. 自定义类型

typedef 关键字 可进行 类型 的重命名

常用的 自定义架构：

struct 结构体　enum 枚举　class 类　union 联合体

居多

11. 数组与指针

与C一致，数组名为该数组的首地址的别名

可看为指向首地址的指针

但 可以使用指针 指向数组 来遍历或访问元素

12. 函数指针

函数名即为指向函数的地址

可以用一个指向函数的指针来调用该函数

格式：返回类型 (*函数指针)(参数列表)

```
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