基础

位运算

- 与`&`：均为1得1，否则0
  - 用于将某些位数清零
  - 用于检查特定位数是否为1
- 或`|`：均为0得0，否则1
  - 用于将某些位数变1
  - 用于检查特定位数是否为0
- 取反`~`：1变0，0变1
  - 用于将特定位数取反

函数

空间回收

函数调用完毕后空间将被回收，因此不能返回指向栈内存的指针

- 返回过程是先清空函数的本地内存空间，保留返回空间至调用语句执行完；返回空间部分是值拷贝，即若是指针则保留其本身内容（指向地址），但指向空间内部的地址已经被回收了
- 返回局部变量，实际上是返回局部变量的值拷贝，返回完毕后空间回收，地址中的内容可能被清除
- 字符串常量储存在只读数据段，可以被返回
- 全局变量和静态局部变量可以返回

```
int FUN1(int a){
  return a+1;
}

int * FUN2(int a){
  int b=a+1;
  return &b;
}

int & FUN3(int a){
  int b=a+1;
  return b;
}

char * FUN4(){
  static char s[]="Hello Rin";
  return s;
}

int main(){
  int a=1,b;
  char * s;
  //正确 返回值（拷贝）
  b=FUN1(a);
  //错误 局部变量的值可能被清除
  b=*(FUN2(a));
  b=FUN3(a);
  //正确 返回静态局部变量 函数退出不会回收 数组同理
  s=FUN4();
  return 0;
}
```

指针和引用

引用是特殊的指针，不需要`*`来区别访问地址还是内容

```
type & referencename=variationname
```

用途

函数参数传递和返回结果

- 参数传递：类似于指针传递，传递地址，避免了值的复制；实参需要是变量
- 返回值传递：需要返回值是一下类型之一，以保证函数调用完毕空间释放后返回值仍然能够

- 全局变量或`static`类型的静态局部变量
- 引用类型的参数
- 对象或其成员

函数指针

- 定义

```
Type (*pFuncName)(Type1 a,Type2 b,...)=Func;
//pFuncName=Func;
//pFuncName(...); 直接调用
```

- 使用
  - 作为函数参数时

```
void Func(int (*pF)(int *)){}
//template<class T>
//class Func{public:void operator()(T pF){};};

void F(int *a){}

int main(){
  int n=2;
  Func(F(n));
  //Func<int (*)(int *)>(F);
  return 0;
}
```

字符串

字符数组和字符指针

- 字符串类型可以被强制复制为字符指针，但仍然储存在只读字符段，内容不能改变

```
//s1和s2都可以作为指针操作 但s1是强行赋值的字符串常量
//s1储存在只读空间 不能修改内容 地址也不会被函数调用后释放掉
char * s1="HELLO RIN";
char s2[]="HELLO RIN";
```

关键字

`const`关键字

- `const`表示分配储存空间后立即初始化，并且不再允许修改
- 对于指针，可以先不赋值；但后续只能用非`const`指针赋值`const`指针而不能相反
    - 指针内容不能改变，但指针可以改变

```
char * p1="Alice";
char * p2="Bob";
const char * p=p1;
p=p2;//合法
```

- 常引用可以引用非`const`值，但不能修改引用，只能修改原值

  常值只能赋值给常引用
- 不改变变量时尽量使用`const`，因为运算符重载，例如`<<`和`=`右侧都要求对象为`const`，特别是返回值为临时对象时，为了识别赋值，必须声明形参为`const`

`static`关键字

- `static`相当于全局变量
- 类的`static`属于同一类共有
- 在类中使用`static`需要在类外像全局变量一样声明或初始化
    - `static`的初始化需要在类外部进行