C++ 程序设计

冯哲

2020 年 12 月 28 日

目录

1 期中

1.1 数据类型

1.1.1 基本类型、导出类型

基本类型 `(short/long)int,float,double,char,void`

导出类型 数组、指针、结构体、共同体、类

1.1.2 基本数据类型与占用字节 Byte 数（32 位）

数据类型	占用字节数
`(signed)/unsigned char`	1
`(signed)/unsigned short(int)`	2
`(signed)/unsigned (long) int`	4
`float`	4
`(long)double`	8

1.1.3 常量和变量

整型常量

十进制 1 ∼ 9 开头

八进制 0 开头

十六进制 0X 或 0x 开头
L/l 结尾：`long int`
U/u 结尾：`unsigned int`
UL/ul/LU/lu 结尾：`unsigned long int`

实型常量

十进制小数 `0.123=.123;-56.`→ 必须有点

指数型 无要求数 + `E/e` + 整数

字符型常量 `char`、单引号

普通字符 可显示字符

转义字符 控制字符（见课本）;`ASCII` 码：`'\ddd'`→ 三位八进制数，`'\xhh'`→ 两位十六进制数
注意反斜杠\、单引号'、双引号" 的输出方法

字符串常量 两点说明:
`C++` 规定以字符`'\0'` 作为字符串结束的标志
双引号必须用转义字符表示

变量 标识符：字母、数字、下划线 _, 第一个字符不可是数字

1.2 运算符和表达式

一些要点

1.2.1 数据类型转换

自动 `char/short` → `int`,`int` → `unsigned` → `long` → `double`,`float` → `double`
`char` → `ASCII` 码

强制 （< 类型 >）< 表达式 > 或 < 类型 > （< 表达式 >）

转换对象在存储单元中内容和类型没有改变

1.2.2 自增减

`i++ i--` 先引用后自增/减

`++i --i` 先自增/减后引用

1.2.3 逻辑表达式

`&& || !`
当 `&&` 左边的表达式为 0 时，逻辑值为 0，右边表达式不计算
当 `||` 左边表达式为 1 时，值为 1，右边不计算

1.2.4 逗号表达式

最后一个表达式的值作为整个表达式的值

1.2.5 运算顺序

`()` → 单目（`!` ,± ,`++`,`--`, 类型转换）→ (`*`,`\`,`%` → `+`,`-`) → (`>`,`>=`,`<`,`<=` → `==`,`!=`) → `&&` → `||`) → (`=`) → ,

1.2.6 数据类型长度运算符

`sizeof`（< 类型/表达式 >）

1.3 分支语句

1.3.1 `if`

`if`

`if...elseif...else`

`if...else`

if 语句嵌套

1.3.2 条件表达式

```
<1>?<2>:<3>
```

1.3.3 switch

```
switch(<>)
{
case<1>:...;break;
⋮
defult:...;
}
```

1.4 循环语句

1.4.1 while（当型）

当条件成立时执行

1.4.2 do...while（直到型）

执行直到条件不成立 → 分号；至少执行一次

1.4.3 for

```
for(<1>;<2>;<3>) <...>
= <1>; while(<2>) { <...> <3>;}
```

1.5 控制执行顺序的语句

1.5.1 break;

跳出该层循环

1.5.2 continue;

该层循环中，结束本次，开始下次
(do...)while → 判段条件 或：
for → <3>

1.5.3 语句标号和 goto

< 语句标号 >:< 语句 > 语句标号与标识符规则相同

goto< 语句标号 >

限于本函数内

可从条件/循环 内 → 外 可 外 → 内

goto 和 if 可构成循环

<stdlib.h>：终止程序的运行

exit(...)；正常终止，做收尾工作

abort(...)；异常终止，不做收尾工作

1.6 数组

1.6.1 一维数组

可以不指定数组长度而由系统算出

把数组定义为全局变量或静态变量时，初值均为 0；定义为其它类型的局部变量时，初值随机

⟶ 冒泡法、选择法、擂台法

1.6.2 二维数组

行数可不指定，列数必须指定

1.6.3 字符数组

长度、储存单元：必须考虑 '\0'

初始化赋值

```
char str[12] = {'C','+','+','p',...,'\0'};

char str[] = {'C','+','+','p',...,'\0'};

char str[] = {"C++ program"};

char str[] = "C++ program";
```

输入、输出

仅给出字符数组名

输入空格或换行时，认为字符串结束，自动加'\0'

cin.getline(str,n) $\longrightarrow$ 输入一行（可带空格）

字符串处理函数 <string.h>

求字符串长度 strlen() 不包括'\0'

字符串复制 strcpy(<1>,<2>) <2>→<1>
<1> 必须是数组名，<2> 可以是字符串常量/数组名
<1> 应该足够大
'\0' 也被复制

字符串连接 strcat(<1>,<2>) <1> 之后的'\0' 取消

字符串比较 strmcp<1>,<2> 比较 ASCII 码，第一个不同字符的比较结果作为结果

str1 = str2 → 0;

str1 > str2 → +;

str1 < str2 → -;

大写 → 小写：strlwr，小写 → 大写：strupr

1.7 函数

1.7.1 定义和调用

函数原型说明中可省略形参名

实参与形参仅是值传送，占用不同的内存单元

1.7.2 数组作为函数参数

数组元素 值传送

数组名

一维数组 传地址

定义函数时，数组 [长度] 可有可无；调用时，只写数组名

实参数组与形参数组类型应一致，大小可不一致

二维数组 传地址

行数可不指定，列数必须指定

1.7.3 内联函数

定义时前面加 inline

空间换时间

1.7.4 具有默认参数值的函数

定义时指定默认值，调用时若给出实参值，则使用；未给出实参值则使用默认值

原型说明可简写为 `int add{int =5,int =6,int =7};`

定义时，有默认值的参数必须位于右侧

调用时，省略某实参，则后面都应省略

1.7.5 函数的重载

定义时形参必须不同

若函数名、形参均相同，但返回值不同，则不能定义

1.8 变量的存储类型

1.8.1 作用域

块作用域 不同作用域变量允许同名，局部优先

文件作用域 函数外定义/`extern` 说明，定义位置到源程序文件结束

函数原型作用域 原型说明与定义说明中变量可不同

函数作用域

类作用域

1.8.2 局部/全局变量

局部变量 函数或块内定义

全局变量 函数外定义，有文件作用域

可同名，同名时局部优先，但用`::`引用全局变量

1.8.3 动态/静态变量

动态变量 生存期仅在作用域内

静态变量 生存期为整个程序的执行期

1.8.4 存储类型

自动类型 `auto`

`auto` 可省略，省略即默认 `auto`

不赋初值则初值不定

寄存器类型 `register`

常用作循环控制变量

处理方式随系统变化

静态类型 `static`

静态局部变量
不回收储存空间
默认初值为 0
只能由定义它的函数引用
保存函数的运行结果

静态全局变量 `static` 可省略

外部类型 `extern`

同一文件中使用 `extern` 说明，扩展全局变量作用域

同一程序不同文件中使用 `extern` 说明，不同文件之间可引用相同的全局变量或函数

变量类型	全局/局部变量	作用域	静/动态变量	存储区
自动变量	局部变量	块作用域	动态变量	动态存储区
寄存器变量	局部变量	块作用域	动态变量	**CPU** 寄存器
静态局部变量	局部变量	块作用域	静态变量	静态存储区
静态全部变量	全局变量	文件作用域	静态变量	静态存储区
外部变量	全局变量	文件作用域	静态变量	静态存储区

1.9 编译预处理

几个要点

1.9.1 宏定义

不作正确性检查

不检验类型

只做字符替换，不分配内存空间，不进行值的传递处理

带参数时，应用括号括起来

终止宏名作用域 `#undef N`

1.9.2 条件编译

格式 1	格式 2	格式 3
`#ifdef` 标识符	`#ifndef` 标识符	`#if` 表达式（只能包含常量）
程序段 1	程序段 1	程序段 1
(`#else` 程序段 2)	(`#else` 程序段 2)	`#else` 程序段 2
`#endif`	`#endif`	`#endif`

1.10 指针（部分）

1.10.1 基础

类型是所指储存单元中内容的类型

赋值 p=&a p=p1 p=0

加 n、减 n，是指向前/后 n 个元素，其地址 ± n × sizeof

关系运算 对地址进行比较

	b=*p++;	⇔	b=*p;p++;
	b=*++p;	⇔	++p;b=*p;
运算顺序	b=(*p)++;	⇔	b=(*p);(*p)++;
	b=*(p++);	⇔	b=*p;p++;(与第一行等价)
	b=++*p;	⇔	b=++(*p);

1.10.2 一维数组

数组指针 数组 a 的指针 = a = &a[0]

p=a ⇔ p=&a[0]

	for(p=a;p<a+N;p++)	cout << *p << endl;
访问	for(i=0;i<N;i++)	cout << *(p+i) << endl 或 cout << *(a+i) << endl;
	for(i=0;i<N;i++)	cout << p[i] << endl;

地址 p+i a+i &a[i]

值 *(p+i) *(a+i) p[i] a[i]

2 期末

2.1 指针（续）

2.1.1 二维数组

第 i 行行地址	a+i、&a[i]
第 i 行行首地址（第 i 行第 0 列地址）	a[i]、*(a+i)、&a[i][0]
元素 a[i][j] 的地址	a[i]+j、*(a+i)+j、&a[i][0]+j、&a[i][j]
第 i 行第 j 列元素值	*(a[i]+j)、*(*(a+i)j)、*(&a[i][0]+j)、a[i][j]

2.1.2 字符串数组

指针变量可以作为复制函数的参数

字符型指针变量与字符数组的区别

分配内存方式 前者存内容，后者仅存地址

初始化赋值含义

赋值方式 前者只能对元素逐个赋值

输入/输出方式 前者可以直接 `cin/cout`, 后者只有在被赋字符数组地址之后才可以

值的改变 前者表示的起始地址不可改变

2.1.3 指针变量与数组作为函数参数

指针变量作为函数参数，为传地址方式, 相当于有多个返回值

数组与指针作为函数参数，实参、形参、数组名、指针变量共有四种组合，效果相同

2.1.4 指针数组和指向一维数组的指针

指针数组 < 类型 >*< 数组名 >[< 数组长度 >]
一个指针可以指向一个字符串，则一组指针指向一组字符串

指向一维数组的指针 < 类型 >(*< 指针变量名 >)[< 二维数组列数 >]
用于表示二维数组中某行的各元素值
赋值：`p=&a[`二维数组第某行`]`

2.1.5 返回指针值的函数和函数指针变量

返回指针值的函数 < 类型 >*< 函数名 >(< 形参 >) {函数体}

函数指针变量 < 类型 >(*< 变量名 >)(< 参数表 >)
函数指针仅仅是定义了一个函数的入口地址，在使用之前必须将函数名（即函数入口地址）赋给函数指针变量
作用是用函数指针变量来代替多个函数
调用：(*< 指针变量 >)(< 实参表 >) 或 < 指针变量 >(< 实参表 >)

2.1.6 new 和 delete 运算符

new 定义格式：
```
< 指针变量 > = new< 类型 >;
< 指针变量 > = new< 类型 >(value);        能直接赋上初值
< 指针变量 > = new< 类型 >[< 表达式 >];   分配指定类型的数组空间
```

delete 定义格式：
```
delete< 指针变量 >;
delete[]< 指针变量 >;   将所指的一维数组内存空间归还
```

注意事项

new 时不能赋初值

new 出来指针变量值为 0 表示分配失败，应终止执行程序

new 之后记得 delete

2.1.7 引用类型变量和 const 类型的指针

引用类型变量

定义 < 类型 >&< 引用变量名 >=< 变量名 >;
必须初始化，初始化变量必须同类型，初始化值不能是常数

```
float &rx = * new float;
...
delete &rx;
```

可用动态分配内存空间来初始化一个引用变量：

作为函数参数 传地址

const 类型变量

const 型常量 注意与 #define 的比较

const 型指针 定义方式：
```
const < 类型 > *< 指针变量名 >;
< 类型 > *const < 指针变量名 >;
const < 类型 > *const < 指针变量名 >;
```

2.2 枚举类型

2.2.1 枚举类型

定义 enum < 枚举类型名 > {< 枚举类型表 >};

序号 可默认，可全指定，可部分指定；允许两个不同元素取相同整数值，只是无意义

2.2.2 枚举类型变量

定义 三种定义方式：
```
（前面已定义枚举类型）< 枚举类型名 > < 变量 >;
enum < 枚举类型名 > {< 枚举元素表 >}; < 变量 >;
enum {< 枚举元素表 >} < 变量 >;
```

引用
赋值：枚举变量 = 枚举元素/枚举变量
输入/输出：不能 cin >>，cout << 输出的是序号值而不是元素值
比较：比较的是序号值

2.3 结构体

2.3.1 结构体类型

定义时记得最后的分号！

2.3.2 结构体变量

定义 三种定义方式同枚举类型，定义时即可初始化

引用 < 结构体变量 >.< 成员名 >
结构体变量的成员可以输入/输出
结构体变量不可以输入输出，但是可以相互赋值

结构体变量与结构体变量数组作为函数参数 结构体变量作参数为值传送，结构体变量数组为地址传送

2.4 类和对象

2.4.1 类

定义 默认 `private`, 记得加逗号

说明

在类体外定义成员函数 < 类型 >< 类名 >::< 成员函数名 >(形参表){函数体}

定义数据成员时不能赋初值，因为没有分配内存单元

2.4.2 对象

定义 三种定义方式同枚举类型

存储空间的分配 系统为数据成员分配不同内存空间，不用对象的成员函数则共享同一内存空间

引用 引用方式同结构体，注意在类内引用本类的数据成员或调用本类的成员函数时，不能加对象名