**Ch1：**

P6-7：程序创建

P8：编译和链接

**Ch2：**

主函数：

#include <iostream> // 预处理器，头文件

using namespace std；// 名称空间

int main()// 主函数

{

*statements*;

cin.get();

return 0;

}

声明，赋值，函数原型

**Ch3：**

整形char, short, int, long, long long. unsigned

双下划线或下划线和大写字母打头的名称不推荐使用，因为要保留给编译器

#include <climits>

int n;

sizeof(int);

sizeof(n)

C++11的初始化列表,用{},可以省略=，内容为空时默认初始化为0。

int emus{7};

int rocs={};

wchar\_t bob=L’P’;

C++11新增的char16\_t和char32\_t，用前缀u和U指出：

char16\_t ch1=u’q’;

char32\_t ch2=U’\U0000222B’;

P45：8进制，10进制和16进制

P50：表3.2，转义序列

const限定符，在声明的时候初始化

浮点类型float, double, long double。E表示法：-8.3E-4

除法(/)，若操作数都是整数，则结果也是整数，小数部分被抛弃；若至少有一个浮点数，则结果也是浮点数。

类型转换。1)初始化或赋值时，转换为接受变量的类型，注意缩窄(narrowing); 2)C++11用初始化列表进行转换，不允许缩窄; 3)表达式中的转换; 4)传递参数时的转换; 5)强制类型转换，如将thorn从int转换为long的两种方法：

(long) thorn;

long (thorn);

四种强制类型转换符，如static\_cast<long> thorn，Ch15中介绍。

C++11中的auto声明。

**Ch4**

**数组**

数组声明：typeName arrayName[arraySize];其中arraySize必须为整形常数或整形表达式，不能是变量。数组索引从0开始编号。只有在定义的时候可以初始化数组，也不能将一个数组赋值给另一个数组，使用大括号初始化。可以用C++11的初始化列表。

字符串以空字符“\0”结尾

#include <cstring>

char name[15]=”C++owboy”;

sizeof(name); //数组的长度Byte

strlen(name); //字符串的长度,不包括空字符

cin使用空白(空格，制表符和换行符)来确定字符串的结束位置。cin.getline(name,lenth)读取一行数据，通过换行符来确定结尾，但不读取而是丢弃换行符，读取数据保存到变量name中，最长保存lenth-1个字符。cin.get()与getline类似，但get将换行符保留在输入列表中。不带参数的cin.get()可以读取下一个字符，包括换行符。数字和字符串混合输入时，用cin.get()来处理输入列表中的换行符。

#include <string>

using namespace std;

string str1,str2,str3;

str1 = str2; //赋值

str3 = str1+str2; //拼接

str1 += str2; //拼接

str1.size();

getline(cin, str1);

cin>>str1;

**结构**

2步创建结构：定义结构描述和创建结构变量。

定义描述：

struct structName

{

int a;

char c[20];

}

创建变量：

structName hat=

{

10,

”Liu kai”

}; //初始化，逗号间隔

structName hat {10,”Liu kai”}; //C++11初始化

在定义时创建变量：

struct structName

{

int a;

char c[20];

}hat=

{

10,

”Liu kai”

};

用hat.a访问结构成员a

初始化结构数组，在花括号中，用逗号隔断：

structName hat[2]=

{

{10,”Liu kai”},

{20,”Kai liu”}

};

**共用体**

关键字uion，与结构类似，但只能存储一种类型。用途之一为，当数据使用多种格式(但不会同时使用)时，可以节省空间。

**枚举**

关键字enum，提供了另一种创建符号常量的方法，可以代替const，如：

enum Num{one, two, three};

Num是枚举(类似于结构)，one, two, three为**枚举量**，是符号常量，默认为0~2, 可以通过显式地指定整数值来覆盖默认值。

Num temp;

在不进行强制类型转换的情况下，只能将枚举量(one)赋值给定义的枚举变量(temp)。枚举只有赋值运算，没有算数运算。

枚举量是整形，可被提升为int型，但int型不能自动转换为枚举类型。

int color = one; //valid

color = red+3; //valid, red转化为int

temp = 3；//invalid

通过强制类型转换，赋值给枚举变量

temp = Num(2);

设置枚举量：

enum Num{first, second=100, third};

first默认为0，second为100，third为前一个枚举值加一，所以为101。枚举量可以是相同的值。

枚举的取值范围：

上限是打于枚举量最大值的最小的2的幂减1。下限是，若枚举量的最小值不小于0，则下限为0；若最小值小于0，则为最大的小于最小枚举量的2的幂加1。如枚举：

enum scale{one=-5, two=1, three=13};

则下限为-7=(-23+1)，上限为15=(24-1)。

**指针**

定义数据：int data;

声明指针：int \* pt;

赋值： pt = &data;

\*pt和data等价，表示值；pt和&data等价，表示地址。

对每个指针都需要一个\*： int\* p1,p2; //表示指针p1和int变量p2

**注意**：C++再创建指针时只分配了存储地址的内存，而没有分配存储指针所指向的数据的内存，需要给数据提供空间，否则容易出错，一定要在对指针应用解除引用符(\*)之前，将指针初始化为一个确定的、适当的地址，这是使用指针的金科玉律。

**new和delete**

int \* pn = new int;

delete pn；

new和delete必须成对使用，但对空指针使用delete是安全的。

int \* pt=new int [10];//用new创建数组

delete [] pt;

指针和数组基本等价。

int p1[10];

int \*p2=&p1[0];

则p2[1]和\*(p2+1)等价。但sizeof(p1)表示数组的长度，sizeof(p2)为指针长度。

**字符串与指针**

在cout和多数C++的表达式中，char数组名、char指针和引号括起来的字符串常量都被解释成字符串第一个字符的地址。

char animal[20]=”bear”;

char \* ps;

ps=animal; //将ps指向字符串，但并没有复制字符串

ps=new char[strlen(animal)+1];//复制字符串的副本

strcpy(ps,animal);

delete [] ps;

**用new创建动态结构**

inflatable \* ps =new inflatable;

但访问成员时不能用点运算符(.)，而应该用箭头运算符(->):如ps->price;或(\*ps).price。

模板类vector和array(C++11)

#include <vector>

#include <array>

using namespace std;

vector<int> vi;

array<int,4> ai;

**Ch5:**

for(int i=0; i<5; i++)

{

}

1.初始化测试值。2.条件判断。3.执行循环。4.更新测试值。

a++使用a的当前值计算表达式，然后加1；++a先将a加1，再计算表达式。

P135递增递减运算符和指针re

strcmp(st1,st2)，测试st1和st2是否相同，相同返回0，不同返回1

P145

**Ch8：**

**内联函数：**

一般函数调用需要保存参数并跳转，但内联函数将编译代码与其他程序代码内联起来，而无需跳转。内联函数比一般函数稍快，但会占用更多内存。一般适用于代码量小但调用次数多的函数。内联函数需要：1. 在函数声明前加上关键字inline；2. 在函数定义前加上关键字inline。内联函数不能递归。

**引用变量：**

引用变量是已定义变量的别名，主要用途是作为函数的形参，函数将使用原始数据而非副本。这里用&符号来定义引用变量。必须在声明引用变量时初始化。

引用变量作为函数参数时，被调用函数可以访问调用函数的中的变量，称为按引用传递。与之功能对应的是C中的指针传递。C中常用的按值传递不能改变调用函数中的变量，只能使用调用函数的值的拷贝。

P260