▲

i=3 cout<<-i++ //输出的i是多少？

\*\* -3 //cout返回临时变量即-i=-3 此时i的值的4

▲

a&&b&&c :只有a为真时才判断b，b为真再判断c

int a=1，b=2，c=3，d=4；

int m=1，n=1；

（m=a>b）&&(n=c>d)://m和n的值是多少？

\*\*0和1 //m=0运算结束，后半部分不运算

▲

Inta，b,c;

C=(a=1,b=a+2,b+3);//a为1，最后c=6

▲

a=5,b=7,c=8

z=(a++%5)//(++b/7)//(++c-8) //求abcz

\*\*a=6 b=8 c=8 z=1

//A++先运算后在自增；++b先自增为8 结果为1；c不做运算

▲

输出字符的ascii码

\*\*

char a;

cin>>a;

cout<<a-0<<endl;

输出的a为97

▲

Int a=5,b=6;

B=++a // a=6,b=6

B=a++ // b=6,a=7

△乘方运算

#include<cmath>

c=pow(double(a),b);

▲

双引号：从当前安装目录（用户提供的头文件只能双引号）

尖括号：从库安装目录

▲**字符串函数**

**<string.h>**

**1.**strcpy(s1,s2):复制s2到s1

2.strncpy(s1,s2,n):复制s2的前n个字符到s1

3.strcat（s1，s2）：在s1后连接s2，s2不变化

4strncat（s1，s2，n）：在s1后连接s2的前n个字符

5.strcmp(s1,s2): 大于0 s1>s2 等于0 相等

6.strlen（str）:str 的长度

7.atof(“123.456”):字符串转为float

atoi（“123”）：字符转为int

<string>

1. str.c\_str():string 转换为c风格字符串，返回指针
2. str.copy(s1,n,pos) 把str中从pos开始的第n个字符复制到s1
3. str1=”12” str2=”AB” str1+str2=”12AB”
4. str1==str2
5. str1.size()/str1.length()///:获取长度
6. str1.empty() :检查是否为空
7. str.substr(2,4)//:返回下标2开始的4个字符
8. str.find(“DEF”,pos)//:从pos开始查找字符串“DEF”在str中的位置，返回D的下标
9. str.erase(3,5)//:从下标3开始往后删5个字符，
10. str.append(“12345”,1,3)//:在str末尾加上“12345”下标从1开始的3个字符，即“234”
11. str.replace(p0,n0,s1,n)//:删除从po开始的n0个字符，然后在po处插入字符串s1前n个字符

12.str.insert(p0,str2,pos,n)//:在p0位置插入字符串str2中pos开始的前n个字符

***指针变量***

定义： 基类型 \*指针变量名

指针变量只能指向和其基类型相同类型的变量，如基类型为int的指针变量只能用来指向整型变量，决不能指向实型变量

Int \*point\_1,\*point\_2,a

注意：指针变量名还是point\_1而不是\*point\_1

相关运算符：

（1）&：取地址运算符

（2）\*：指针运算符

▲**eg：**

Int arry[10]={1,2,3,4,5}

Int \*p=arry;

Cout<<\*(p+1); //输出结果：2

P++;

Cout<<\*p; //输出结果：2

Cout<<\*(arry+1);

Arry++; //输出结果：error

Cout<<\*(aryy+1); //输出结果：2

Cout<<\*(arry+5); //输出结果：0

**Vector**

* 是一个快速的动态分配内存的数组（容器）
* 动态数组，可以在运行阶段设置长度
* 具有数组的快速索引方式
* 可以插入和删除元素

**△定义和初始化**

**#Include<vector>**

Vector<double>vec1;

Vector<string>vec2(5);

Vector<int>vec3(20,998); //可以放20个元素，每个都是998

Vector<int>vec(n,n+10) ; n[10]={0} //可以通过数组向向量传值

**△常用操作**

Clear() //清空容器

Empty() //判断为空

Size() //返回元素个数

[index] //返回index位置

Erase(pos) //删除pos处数据

Earse(beg,end)//删除[beg，end）区间的数据

Front() //返回第一个元素

Insert(pos,elem)//在pos处插入一个元素

Pop\_back() //删除最后一个元素

Push\_back(elem)//在容器末尾插入elem

Resize(num) //重新设置容器大小

Begin(),end() //返回容器手尾元素迭代器

Reverse(vec.begin(),vec.end()) //翻转向量中的元素

#inclue<algorithm> -- sort(vec.begin(),vec.end()) //向量里的元素排序

**Eg,**

Vector<double> vecDouble ={98.5,67.9,43.6,32.9} ;

//向数组中插入数字

vecDouble.push\_back(100.8) ; //在数组尾部插入一个数字

// 遍历

Vector<int>: :iterator it;

For(it=vec.begin() ; it != vec.end() ; ++i )

Cout<<\*it<<endl;

//使用迭代器