C++primer 第五版 第四章 答案

4.1 答：105

4.2 答：（a）*（vec.begin（））；（b）*（vec.begin（））+1

4.3 答：这可以接受。因为，操作数的求解次序通常对结果没什么影响。只有当二元操作符的两个操作数涉及同一对象，并改变该对象的值时，操作数的求解次序才会影响计算结果；后一种情况只会在部分（甚至是少数）程序中出现。在实际使用中，这种“潜在的缺陷”可以通过程序员的努力得到弥补，但“实现效率”的提高却能使所有使用该编译器的程序受益，因此利大于弊。

4.4 答：(((12 / 3) * 4) + (5 15))+ ((24 % 4) /2)；

4.5 答：-86、-18、0、-2；

4.6 答：ival%2==0；判断结果是否为 true，若为 true 则为偶数，否则为奇数；

4.7 答：2^16+1；10000*10000；3276*3276；

4.8 答：采用短路求值的策略，逻辑与、逻辑或这两种运算符都是先求左值，当且仅当左侧运算对象为真时才对右侧运算对象求值。相等性运算符(==)优先级别高于其他逻辑运算符。

4.9 答：当指针 cp 不为空指针且 cp 指向的字符不为空字符就能输出字符；

4.10 答：a>b&&b>c&&c>d;

4.11 答: i != (j<k)；

4.13 答：（a）d=i=3；（b）i=3，d=3.5；

4.14 答：if（42=i）这个非法，42=i 是个赋值表达式，42 是个字面值不能作为作为左操作数；if（i=42）这表达式合法， i=42 的结果永远为真。

4.15 答：pi 是一个指针变量，得到 0 值，表示的是地址，不能赋值给 int 型 ival，类型不一样时可以执行隐式转换，但是这里不能自动把指针类型转换成 int 型，可以把表达式修改成，dval=ival=0； pi=0；

4.16 答：（a）该表达式的意图是 getPtr()赋值给 p，然后判断 p 是否为 0，=优先级比!=优先级低，先把 getptr()是否为 0 的结果赋值给 p，p 得到的结果不是 true 就是 false。应该修改成为 if((p=getPtr()) ! = 0);（b）该表达是想要判断 i 是否等于 1024，而表达式的意思是把 1024 赋值给 i，然后判断 i=1024 整个是否为真。应该修改成 if(i==1024);

4.17 答：前置版本是首先将运算对象加 1（或减 1），然后将改变后的对象作为求值结果，后置版本也会将对象加 1（或减 1），但是求值结果是运算对象改变之前的那个值得副本。

4.18 答：写成前置的形式是：*++pbeg；vector 中的第一个元素（对象）没有输出，而且还试图解引用尾后迭代器，这是未定义的行为，将引发错误。

4.19 答：（a）ptr 指针不为空指针且*ptr++为真时整个表达结果才为真。（b）非法，改成 ival&&ival++；（c）非法，vec[ival++]<=vec[ival--];

4.20 答：（a）合法；（b）合法；（c）非法，指针没有 empty()的成员，改成(*iter).empty()；（d）合法；（e）合法；（f）合法；

4.21 答：#include<iostream>　　#include<vector>　　using namespace std;

```
int main()
{
    vector<int> ivec{0,1,2,3,4,5,6,7,8,9};
    for (vector<int>::iterator itr = ivec.begin(); itr != ivec.end(); ++itr)
        *itr = ((*itr % 2 == 0) ? *itr : *itr * 2);
    for (auto itr = ivec.begin(); itr != ivec.end(); ++itr)
        cout << *itr << ' ';
```

```
    cout << endl;
    return 0;
}
```

4.22 答：

```
#include <iostream>    #include <string>    #include <vector>
using namespace std;
int main()
{
    vector<unsigned> grades;
    unsigned i;
    while (cin >> i)
        grades.push_back(i);
    for (vector<unsigned>::const_iterator grade = grades.begin();
        grade != grades.end(); ++grade)
    {
        string finalgrade = (*grade < 60) ? "fail" : "pass";
        finalgrade = (*grade > 90) ? "high pass"
            : (*grade < 60) ? "fail" : "pass";
        cout << *grade << " " + finalgrade << endl;
    }
    return 0;
}
```

4.23 答：

```
string p1 = s + ((s[s.size() - 1] == 's') ? "" : "s");
```

4.24 答：先判断成绩是否小于 60，再判断成绩是否大于 90；

4.25 答：10000000

4.26 答：如果在 32 位系统倒是没有什么问题，在某些机器上并不确定 int 占 32 位，unsigned long 在任何机器上都将至少拥有 32 位。

4.27 答：结果分别为：3、7、true、true；

4.28 答：

```
#include <iostream>
using namespace std;
int main()
{
    cout << sizeof(char) << endl;
    cout << sizeof(short) << endl;
    cout << sizeof(unsigned) << endl;
    cout << sizeof(int) << endl;
    cout << sizeof(unsigned int) << endl;
    cout << sizeof(long) << endl;
    cout << sizeof(unsigned long) << endl;
    cout << sizeof(float) << endl;
    cout << sizeof(double) << endl;
    cout << sizeof(void *) << endl;
}
```

4.29 答：

```
#include <iostream>    #include <string>    #include <vector>
using namespace std;
```

```
int main()
{
	int x[10],  *p = x;
	cout << sizeof(x) << endl;//40
	cout << sizeof(*x) << endl;//4
	cout << sizeof(p) << endl;//4
	cout << sizeof(*p) << endl;//4
	cout << sizeof(x) / sizeof(*x) << endl;//10
	cout << sizeof(p) / sizeof(*p) << endl;//1
}
```

4.30 答：sizeof(x + y); sizeof(p->men[i]); sizeof(a) < b; sizeof(f());

4.31 答：

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
int main()
{
	vector<int> ivec;
	int cnt = 10;
	while (cnt > 0)
		ivec.push_back(cnt--);
	vector<int>::const_iterator iter = ivec.begin();
	while (iter != ivec.end())
		cout << *iter++ << endl;
	vector<int> vec2(10, 0);
	cnt = vec2.size();
	for (vector<int>::size_type ix = 0;ix != vec2.size(); ix++, cnt--)
		vec2[ix] = cnt;
	iter = vec2.begin();
	while (iter != vec2.end())
		cout << *iter++ << endl;
	return 0;
}
```

4.32 答：利用指针和下标操作数组中的元素

4.33 答：（someVal？++x），（++y：x），（--y）；

4.34 答：（a）判断 fval 值是否等于 0；（b）先求和，再把求和的值赋值给 dval；（c）先求积，再把所得值与 dval 相加；

4.35 答：（a）（b）、（c）、（d）都会发生隐式转换；测试如下；

```
#include <iostream>
using namespace std;
int main()
{
	char c;
	int ival = 2;
```

```
    float fval = 2.1;
    double dval = 2.3;
    unsigned int ui = 1;
    cout << ival*1.0 << endl;
    cout << (fval = ui - ival*1.0 )<< endl;
    c = ival + fval + dval;
    cout << c << endl;
    c = 'a' + 3;
    cout << c << endl;
}
```

4.36 答：int i*=static_cast<int>(d);

4.37 答：pv 是 double*类型，（d）中是把 double*转换成 char*型

（a）pv=cosnt_cast<void *>(ps);（b）i=static_cast<int>(*pc);

（c）double pv=static_cast<double *>(&d);（d）pc=reinterpret_cast<char*>(pv);

4.38 答：把表达式的求值结果转换成 double 型赋值给 slope；