C++primer 第五版 第三章 答案

3.1 答：用 using 声明就无需专门的前缀（形如命名空间::）也能使用所需的名字，using namespace std；就可以了。也不用像书上一样用到一个就用 using 声明一下。

3.2 答：

```
string line;
    while (getline(cin, line))
        cout << line << endl;//每次读入一整行，直到到达文件末尾
    string word;
    while (cin >> word)
        cout << word << endl;//逐个输出输入的单词，每个单词后紧跟一个换行符
```

3.3 答：string 读取规则：忽略开头的空白并从第一个真正的字符开始读取，直到遇见下一处空白位置。getline 不忽略开头的空白直到遇到换行符，换行符不读取。

3.4 答：

```
#include<iostream> #include<string> using namespace std;
int main()
{
    string s1, s2;
    cin >> s1 >> s2;
    if (s1 == s2)
        cout << "s1 等于 s2"<<endl;
    else
    {
        if (s1 > s2)
            cout << s1 << endl;
        else
            cout << s2 << endl;;
    }
    //改写比较长度
    if (s1.size() == s2.size())
        cout << "s1 的长度等于 s2 的长度" << endl;
    else
    {
        if (s1.size() > s2.size())
            cout << s1 << endl;
        else
            cout << s2 << endl;;
    }
}
```

3.5 答：

```
#include<iostream>  #include<string>    using namespace std;
int main()
{
    string s, word;
    while (cin>>word)
```

http://blog.csdn.net/chenxun_2010

```
    {
        s += word;
    }
    cout << s << endl;
}//按照要求只需把 s+=word 修改 s+=word+' '；即可
```
3.6 答：
```
#include<iostream>    #include<string>    using namespace std;
int main()
{
    string s("chen xun");
    for (auto &c:s)
    {
        if (!isspace(c))
        c = 'X';
    }
    cout << s << endl;
}
```
3.7 答：上题中使用 auto 关键字让编译器来决定变量 c 的类型，其实就是 char 型，我们可以直接设置为 char 型改成 char &c：s 即可。

3.8 答：
```
for (decltype(s.size()) index = 0; index != s.size(); ++index)
        {
            if (!isspace(s[index]))
            s[index] = 'X';
        }//当然是 for 范围语句更简单更加清晰
```
3.9 答： s 是一个空字符串，其长度为 0，输出第一个字符是不合法的。

3.10 答：
```
#include<iostream>    #include<string>    using namespace std;
int main()
{
    string s("chen, xun");
    for (auto c : s)
    {
        if (!ispunct(c))
            cout << c ;
    }
    cout << endl;
    return 0;
}
```
3.11 答：不合法，s 是 const string 类型字符串，是不允许修改其字符串的内容。

3.12 答：b 不合法，svec 是保存 vector<string>类型的对象，而 ivec 保存的是 int 型 vector 的 vector，元素类型不相同，所以是不合法的。

3.13 答：（a）0 个；（b）10 个；（c）10 个 42；（d）一个 10；（e）2 个元素分别是 10,42；（f）10 个元素；（g）10 个元素都是 hi；

3.14 答：
```
#include<iostream>    #include<vector>    using namespace std;
int main()
{
```

http://blog.csdn.net/chenxun_2010

```
    vector<int> ivec;
    int i;
    while (cin >> i)
    {
        ivec.push_back(i);
    }
    for (auto i : ivec)
        cout << i <<' ';
    cout<<endl;
    return 0;
}
```

3.15 答：

```
#include<iostream>    #include<vector>    #include<string>    using namespace std;
int main()
{
    vector<string> svec;
    string    word;
    while (cin >> word)
    {
        svec.push_back(word);
    }
    for (auto s : svec)
        cout << s << endl;
    return 0;
}
```

3.16 答：用 v.size 输出容量，不是空的 vector 就使用 for 语句输出 vector 中的对象。应该很简单吧，如果你做不到反复看 vector（从 87 页看起，没什么捷径）。

3.17 答：http://blog.csdn.net/chenxun_2010/article/details/31778891

```
#include<iostream>
#include<vector>
#include<string>
using namespace std;
int main()
{
    vector<string> svec;
    string    word;
    while (cin >> word)
    {
        svec.push_back(word);
    }
    for (auto s : svec)
    {
        for (auto &c : s)
        //for (decltype(s.size()) index = 0; index != s.size(); ++index)
        {
```

http://blog.csdn.net/chenxun_2010

```
            c = toupper(c);
            //s[index] = toupper(s[index]);
        }
        cout << s << endl;
    }
    return 0;
}
```

3.18 答：不合法：ivec 是一个空的 vector，根本不包含任何元素，是不能通过下标去访问任何元素，这个时候只能使用 push_back 向 vector 中添加元素。vector 下标运算符可以用于访问已经存在的元素，而不能用来添加元素。

3.19 答：vector<int> ivec(10,42); vector<int> ivec{42,42,……}; vector<int> ivec={42,42,……};
其实还可以另外已知的来初始化 ivec，也可以用添加元素的方法。

3.20 答：

```
#include<iostream>    #include<vector>  #include<string>   using namespace std;
int main()
{
    vector<int> ivec;
    int   i;
    while (cin >> i)
    {
        ivec.push_back(i);
    }
    for (decltype(ivec.size()) index = 0; index < ivec.size()-1; index=index + 2)
    {
        cout << ivec[index] + ivec[index + 1] << endl;
    }
    if (ivec.size() % 2 != 0)
        cout << "最后一个元素没有求和" << endl;
    return 0;
}
```

修改：

```
#include<iostream>  #include<vector>  #include<string>   using namespace std;
int main()
{
    vector<int> ivec;
    int   i;
    while (cin >> i)
    {
        ivec.push_back(i);
    }
    vector<int>::size_type fir = 0;
    vector<int>::size_type last = ivec.size() - 1;
    for (;fir<last; fir++, last--)
    {
        cout << ivec[fir] + ivec[last] << endl;
    }
```

http://blog.csdn.net/chenxun_2010

```
    if (ivec.size() % 2 != 0)
        cout << "中间一个元素没有求和" << endl;
    return 0;
}
```

3.21 答：使用迭代器输出 vector 的中对象，这里例举一个例子

```
#include<iostream>
#include<vector>
#include<iterator>
using namespace std;
int main()
{
    vector<int> ivec;
    int    i;
    while (cin >> i)
    {
        ivec.push_back(i);
    }
    for (auto it = ivec.begin(); it != ivec.end(); ++it)
        cout << *it << endl;
    return 0;
}
```

3.22 答：#include<iostream> #include<vector> #include<iterator> #include<string>

```
using namespace std;
int main()
{
    vector<string> text;
    string word;
    while (cin >> word)
    {
        text.push_back(word);
    }
    auto it = text.begin();
    {
    for (auto itr = (*it).begin(); itr != (*it).end(); itr++)
        *itr = toupper(*itr);
    }
    for (auto it = text.begin(); it != text.end(); ++it)
        cout << *it << endl;

    return 0;
}
```

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

```
int main()
{
```

http://blog.csdn.net/chenxun_2010

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

3.24 答：

```
#include<iostream>    #include<vector>    #include<iterator>  using namespace std;
int main()
{
    vector<int> ivec;
    int    i;
    while (cin >> i)
    {
        ivec.push_back(i);
    }
    for (auto it = ivec.begin(); it < ivec.end()-1; it=it+2)
    {
        cout << (*it) + (*it+1) << endl;
    }
    if (ivec.size() % 2 != 0)
        cout << "最后一个元素没有求和" << endl;
    return 0;
}
```

3.25 答：

```
#include<iostream>    #include<vector>    #include<iterator>  using namespace std;
int main()
{
    vector<unsigned> scores(11, 0);
    unsigned grade;
    vector<unsigned>::size_type n=0;
    while (cin >> grade)
    {
        n = grade / 10;
        vector<unsigned>::iterator it = scores.begin();
        it = it + n;
        (*it)++;
    }
    for (auto itr : scores)
    {
        cout << itr << ' ';
    }
}
```

3.26 答：用第二种方法的话，当 beg 和 end 都是是指针或者迭代器的话是无法编译通过的，因为指针和迭代器运算不支持相加运算，却支持相减运算。而第一种会确保不会溢出即 mid

http://blog.csdn.net/chenxun_2010

范围不会超过 end，比较安全。所以第一种方法是可靠的。

3.27 答：（a）非法；（b）合法；4*7-14 是个常量表达式；（c）非法；txt_size 函数调用不能代替常量表达式；（d）非法，个数不符合；“fundamental “有 12 个元素；

3.28 答：sa 和 sa2 被初始化为空字符串，这是因为会调用 string 的默认构造函数，在后面将会学习到的；ia 在函数体外会被默认初始化为 0，而 ia2 在函数体内是局部变量，其值是未知的，访问的行为必定是未定义的行为。

3.29 答：数组的是有维度的而且维度必须是明确已知的，不提供 size（）和 pusk_back（）等 vector 具有的操作，不能随意向数组中添加元素，无法改变已知数组的维度。数组对于某些特性的应用来说程序的运行时性能较好，但是相应的也损失了一些灵活性。

3.30 答：这种行为就是最常见的数组下标越界行为，我们知道 c/c++遵循一个伟大的传统，从 0 开始，数组的下标也是从 0 开始，本题中数组下标是从 0 到 9，所以当 ix=array_size 时就越界了，这里把条件改成 ix<array_size 即可。

3.31 答：#include<iostream>　　using namespace std;

```
int main()
{
    int a[10];
    for (int i = 0; i != 10; ++i)
        a[i] = i;
    for (auto k : a)
        cout << k<<' ';
    cout<< endl;
    return 0;
}
```

3.32 答：#include<iostream>　　using namespace std;

```
int main()
{
    const size_t sz = 10;
    int a1[sz];
    int a2[sz];
    for (size_t i = 0; i != 10; ++i)
        a1[i] = i;
    for (size_t i = 0; i != 10; ++i)
        a2[i] = a1[i];
    for (size_t i = 0; i != 10; ++i)
        cout << a1[i] << endl;
}
```

改成 vector：

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    vector<int> ivec1;
    vector<int> ivec2;
```

http://blog.csdn.net/chenxun_2010

```
    for (int index = 0; index != 10; index++)
        ivec1.push_back(index);
    for (auto it = ivec1.begin(); it != ivec1.end(); it++)
        ivec2.push_back(*it);
    for (auto it = ivec2.begin(); it != ivec2.end(); it++)
        cout << *it << endl;
    return 0;
}
```

3.33 答：

```
#include<iostream>    #include<vector>    using namespace std;
int main()
{
    unsigned scores[11];
    unsigned grade;
    while (cin >> grade)
    {
        if (grade <= 100)
            ++scores[grade / 10];
    }
    for (int i = 0; i != 11; ++i)
        cout << scores[i] << endl;
    return 0;
}
```

运行上面的程序，虽然能运行，但是 scores 没有初始化，所以其内容是我们无法预知的。

3.34 答：参与运算的两个指针必须指向同一个数组当中的元素才是合法的行为。两个指针相减的结果的类型是一种名为 ptrdiff_t 的标准库类型，和 size_t 一样，ptrdiff_t 也是一种定义在 cstddef 头文件中机器相关的类型，因为差值可能是负数，所以 ptrdiff_t 是一种带符号类型。

3.35 答：

```
#include<iostream>    #include<vector>    using namespace std;
int main()
{
    const int sz = 5;
    int a[sz] = { 1, 2, 3, 4, 5 };
//  for (int *pbeg = begin(a),*pend = end(a); pbeg != pend; pbeg++)
//      *pbeg = 0;
//  for (int *pbeg = begin(a), *pend = end(a); pbeg != pend; pbeg++)
//      cout << *pbeg << endl;
    for (int *p = a; p != a + sz; ++p)
        *p = 0;
    for (auto i : a)
        cout << i << endl;
    return 0;
}
```

3.36 答：比较字符可以用字符比较函数，比较数组可以比较数组的元素个数是否相同，然后再比较每个下标相同的元素。下面例子比较简单

http://blog.csdn.net/chenxun_2010

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	int a1[5] = { 1, 2, 3, 4, 5 };
	int a2[5] = { 1, 2, 4, 4, 5 };
	bool index ;
	for (int i = 0; i != 5; ++i)
	{
		if (a1[i] != a2[i])
		{
			cout << "两个数组不相等" << endl;
		}
	}
}
```

而比较两个 vector 同 string 一样，两个 vector 对象相等话他们当所含的个数相同，而且对应位置的元素值也相同。关系运算符依照字典顺序进行比较：如果两个 vector 对象的容量不相同，但是在相同的位置上的元素值都一样，则元素较少的 vector 对象小于较多的 vector 对象；如果元素的值有区别，则 vector 对象的大小关系由第一对相异的元素值的大小关系决定。

3.37 答：输出数组的每一个字符，每输出一个字符换行。

3.38 答：两个指针相加是没有意义的，也没有定义两个指针相加，就好像两个学生一个学生考了 70 分，一个考 80，在没有说明情况下，他们相加是没有意义的，而相减就能表示他们之间的距离，指针相减就是表两个指针之间的距离（两个指向的同一个数组）。

3.40 答：比较两个 string

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	string s1 = "chen xun";
	string s2 = "chen xun chen";
	if (s1 > s2)
		cout << s1 << endl;
	else
		cout << s2 << endl;;
}
```

比较两个 c 风格的字符串：调用库函数 strcmp

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
```

http://blog.csdn.net/chenxun_2010

```
	const char s1[] = "chen xun";
	const char s2[] = "chen xun chen ";
	if (strcmp(s1, s2)>0)
		cout << s1 << endl;
	else
		cout << s2 << endl;
	return 0;
}
```

3.40 答：

```
#include<iostream>	#include<cstring>	using namespace std;
int main()
{
	const char *s1 = "chen xun ";
	const char *s2 = "is a good man";
	char *s3=(char*)malloc(strlen(s1)+strlen(s2)+1);
	strcpy(s3, s1);
	strcat(s3, s2);
	cout << s3 << endl;
	free(s3);
	s3 = NULL;
	return 0;
}//如果你碰到 error：4996 请你 Google，相信你能解决，呵呵。
```

3.41 答：

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
	int a[5] = { 1, 2, 3, 4 };
//	vector<int> ivec(a, a + 5);
	vector<int> ivec(begin(a), end(a));
	for (auto i : ivec)
		cout << i << ' ';
	cout << endl;
	return 0;
}
```

3.42 答：

```
#include<vector>
using namespace std;
int main()
{
	vector<int> ivec;
	int i;
	while (cin >> i)
	{
```

http://blog.csdn.net/chenxun_2010

```
            ivec.push_back(i);
        }
        int *a=new int[ivec.size()];
        for (auto itr = ivec.begin(); itr != ivec.end(); ++itr)
        {
            *a++= *itr;
        }
        for (auto i : ivec)
            cout << i << ' ';
        cout << endl;
```

3.43 答：

版本 1：

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    int a[3][4] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
    for (int (&row)[4] : a)
    {
        for (int col : row)
            cout << col<< ' ' ;
        cout << endl;
    }
}
```

版本 2：

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    int a[3][4] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
    for (int i = 0; i != 3; ++i)
    {
        for (int j = 0; j != 4; ++j)
            cout << a[i][j] << ' ';
        cout << endl;
    }
}
```

版本 3：

```
#include<iostream>
#include<vector>
using namespace std;
int main()
```

http://blog.csdn.net/chenxun_2010

```
{
    int a[3][4] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
    using int_array = int[4];
    for (int_array *p = a; p != (a + 3); ++p)
    {
        for (int *q = *p; q != *p+4; ++q)
            cout << *q << ' ';
        cout << endl;
    }
}
```

3.44 答：

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    int a[3][4] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
    //using int_array = int[4];
    typedef int int_array[4];
    for (int_array *p = a; p != (a + 3); ++p)
    {
        for (int *q = *p; q != *p+4; ++q)
            cout << *q << ' ';
        cout << endl;
    }
}
```

3.45 答：

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    int a[3][4] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
    for (auto *p = a; p != a + 3; ++p)
    {
        for (auto *q = *p; q != *p+4; ++q)
            cout << *q << ' ';
        cout << endl;
    }
}
```

http://blog.csdn.net/chenxun_2010

http://blog.csdn.net/chenxun_2010