第3章 练习

3.1节

3.1

略

3.2节

3.2

```
void Ex3_2 () {
    // 一次读取一行
//    string s;
//    while (getline(cin,s)) {
//        cout<<s<<endl;
//    }
// 一次读取一个词
    string s;
    while (cin>>s) {
        cout<<s<<endl;
    }
}
```

3.3

```
string s;
cin>>s;     // 遇到空白符就结束
getline(cin,s);     // 遇到回车符才会结束，能够读取一行的所有字符
```

3.4

```
void Ex_3_4() {
    string s1, s2;
    cin>>s1>>s2;
    if (s1 == s2) {
        cout<<"same"<<endl;
    } else {
```

```
        cout<<(s1 > s2 ? s1:s2)<<endl;
    }
    // 判断是否等长
    if ( s1.size() == s2.size() ) {
        cout<<"equal"<<endl;
    } else {
        cout<<(s1.size() > s2.size() ? s1 : s2)<<endl;
    }

}
```

3.5

```
void Ex3_5() {
    string s;
    string sum;
    while(cin >> s) {
      //  sum += s;
        sum = sum+ s+" ";
    }
    cout<<sum<<endl;
}
```

3.6

```
void Ex3_6() {
    string s = "123";
    for (auto &c : s) {     // 引用形式
        c = 'X';
    }
    cout<<s<<std::endl;
}
```

3.7

```
直接变成char,无法改变s内的字符
```

3.8

```
范围for更好，代码更简洁
```

3.9

```
不合法；  s是一个空字符，s[0]是未定义的
```

3.10

```
void Ex3_10() {
    string s;
    cin>>s;
    string s_deal;
    for (auto c : s) {
        if (!std::ispunct(c))
            s_deal += c;
    }
    cout<<s_deal<<endl;
}
```

3.11

```
合法；
c的类型是 const char c;
```

3.3节

3.12

```
1. 对
2. 错；  类型不匹配
3. 对；  10个"null"
```

3.13

```
1. 0个        2. 10个
3. 10个       4. 1个
5. 2个        6. 10个
7. 10个
```

3.14

```
vector<int> ivec;
int temp = 0;
while(cin >> temp) {
    ivec.push_back(temp);
}
```

3.15

```
vector<string> svec;
string temp;
while(cin >> temp) {
    ivec.push_back(temp);
}
```

3.16

```
3.11练习中，后续只要不修改就不会有问题
```

3.17

```
void Ex3_18() {
    vector<string> svec;
    string temp;
    while (cin>>temp) {
        svec.push_back(temp);
    }
    for (auto &s : svec) {      // 注意这里的引用类型
        for (auto &c : s) {
            c = toupper(c);
        }
    }
    for (auto s : svec)
        cout<<s<<" ";
    cout<<endl;
}
```

3.18

```
不合法;
ivec.push_back(42);
```

3.19

```
vector<int> ivec(10,42);    // 这个更好
vector<int> ivec{42,42,...};
vector<int> ivec={42,42,...};
```

3.20

```
void Ex3_20() {
```

```
    vector<int> ivec;
    int temp = 0;
    while (cin >> temp) {
        ivec.push_back(temp);
    }
    int sum_adjacent = 0;
    for (int i = 0; i < ivec.size(); ++i) {
        sum_adjacent += ivec[i];
        if ( i % 2 == 1) {
            cout<<sum_adjacent<<" ";
            sum_adjacent = 0;
        }
    }
    if ( ivec.size() % 2 == 1) {        // 如果长度是奇数
        cout<<ivec[ivec.size()-1];
    }

       /*   第二种情况
        int l = ivec.size() - 1;
    for ( int i = 0; i <= l/2; ++i) {
        if ( i == l - i) {          // 处理奇数个中间的
            cout<<ivec[i]<<" ";
            break;
        }
        cout<<(ivec[i] + ivec[l - i])<<" ";
    }
    */
}
```

3.4节

3.21

略

3.22

略

3.23

```
void EX3_23() {
    vector<int> ivec(10,1);
    for (auto it = ivec.begin(); it != ivec.end(); ++it) {
        *it *= 2;
    }
    cout<< *(ivec.begin())<<endl;
}
```

3.24

略

3.25

略

3.26

```
防止加法溢出
```

3.5节

3.27

```
1. 非法       2. 合法
3. 非法       4. 非法
```

3.28

```
sa  空字符串
ia  0
sa2 空字符串
ia2 未定义的值
```

3.29

```
数组需要显式指定大小，无法动态扩容；
数组内置操作少
```

3.30

```
ia[10]，超过了数组ia合法的下标
```

3.31

```
int ia[10]={0,};
for ( int i = 0; i < 10; ++i)
    ia[10] = i;
```

3.32

```
int ia2[10];
memcpy(ia2, ia, 10*sizeof(int));    // 直接复制内存，当然也可利用循环

//  如果时vector可以直接用=号  ia2 = ia;
```

3.33

```
未初始化会导致未定义的行为
```

3.34

```
p1 = p1 + (p2 - p1)   ---> p1 = p2
只要p1 p2合法，该语句就合法
```

3.35

```
int *p = ia;        // ia是一个整数数组
for (; p != end(ia); ++ia)
    *p = 0
```

3.36

```
// 比较数组
int l1 = get_array_length(ia1);     // 编写函数获得数组长度
int l2= get_array_length(ia2);
if ( l1 != l2) {
        cout<<"数组不相等";

} else {
    for (int i = 0; i < l1; ++i) {
        if (ia[i] != ia2[i]) {
            cout<<"数组不相等";
            break;
        }
    }
}

```

```
// 比较vector
if (ivec1 == ivec2 ) {
    cout<<"vector相等";
} else
    cout<<"vector相等";
```

3.37

```
程序有错误;
ca这种初始化形式需要显式的添加 \0
while的条件无法获取\0,会输出乱码
```

3.38

```
指针相加是两个地址相加，所以没有意义
```

3.39

```
string类型，直接用关系符号比较就行
C风格字符串需要使用strcmp(ca, ca2)比较
```

3.40

略

3.41 - 3.42

```
vector<int> ivec(begin(ia1), end(ia1));

int i = 0;
for(auot s : ivec)
    ia[++i] = s;
```

3.6节

3.43

```
int ia[3][4];

// 范围for
for(auto & i1 : ia)
    for(auto i2 : i1)
        cout<<i2<<" ";

```

```
// 下标运算符
for(int i = 0; i <= 3; ++i)
    for(int j =0; j <= 4; ++j)
        cout<<ia[i][j]<<" ";

// 指针
for (int (*p)[4] = begin(ia); p != end(ia); ++p)
    for(int *q = *p; q != end(*p); ++q)
        cout<<*q<<" ";
```

3.44

```
using int_array = int[4];
for (int_array *p = begin(ia); p != end(ia); ++p)
    for(int *q = *p; q != end(*p); ++q)
        cout<<*q<<" ";
```

3.45

```
for(auto p = begin(ia); p != end(ia); ++p)
    for(auto q = begin(*p); q != end(*p); ++q)
        cout<<*q<<" ";
```