9597 全面的 MyString
9598 继承自 string 的 MyString
3141 魔兽世界 2

****3141.cpp:

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <string>
using namespace std;
#define WARRIOR_NUM 5
#define WEAPON_NUM 3
#define MAX_WARRIORS 1000

enum { DRAGON,NINJA,ICEMAN,LION,WOLF };
/*
char            *            CWarrior::Names[WARRIOR_NUM]            =
{ "dragon","ninja","iceman","lion","wolf" };
红方司令部按照 iceman、lion、wolf、ninja、dragon 的顺序制造武士。
蓝方司令部按照 lion、dragon、ninja、iceman、wolf 的顺序制造武士。
*/
class CHeadquarter;
class CWeapon
{
    public:
        int nKindNo;
        int nForce;
        static int InitialForce[WEAPON_NUM];
        static const char * Names[WEAPON_NUM];
};
class CWarrior
{
    protected:
        CHeadquarter * pHeadquarter;
        int nNo;
    public:
        static const char * Names[WARRIOR_NUM];
        static int InitialLifeValue [WARRIOR_NUM];
        CWarrior( CHeadquarter * p,int nNo_);
        virtual void PrintResult(int nTime,int nKindNo);
        virtual void PrintResult(int nTime) = 0;
        virtual ~CWarrior() { }
```

```
};
class CDragon;
class CNinja;
class CIceman;
class CLion;
class CWolf;
class CHeadquarter
{
    private:
        int nTotalLifeValue;
        bool bStopped;
        int nColor;
        int nCurMakingSeqIdx;
        int anWarriorNum[WARRIOR_NUM];
        int nTotalWarriorNum;
        CWarrior * pWarriors[MAX_WARRIORS];
    public:
        friend class CWarrior;
        static int MakingSeq[2][WARRIOR_NUM];
        void Init(int nColor_, int lv);
        ~CHeadquarter () ;
        int Produce(int nTime);
        void GetColor( char * szColor);
        int GetTotalLifeValue() { return nTotalLifeValue; }

};

class CDragon:public CWarrior
{
    private:
        CWeapon wp;
        double fmorale;
    public:
        void Countmorale()
        {
            fmorale    =    pHeadquarter    ->    GetTotalLifeValue()
/(double)CWarrior::InitialLifeValue [0];
        }
        CDragon( CHeadquarter * p,int nNo_):
            CWarrior(p,nNo_) {
            wp.nKindNo = nNo % WEAPON_NUM;
            wp.nForce = CWeapon::InitialForce[wp.nKindNo ];
            Countmorale();
        }
```

```
        void PrintResult(int nTime)
        {
            CWarrior::PrintResult(nTime,DRAGON);
            printf("It has a %s,and it's morale is %.2f\n",
                CWeapon::Names[wp.nKindNo], fmorale);
        }
};
class CNinja:public CWarrior
{
    private:
        CWeapon wps[2];
    public:

        CNinja( CHeadquarter * p,int nNo_):
            CWarrior(p,nNo_) {
            wps[0].nKindNo = nNo % WEAPON_NUM;
            wps[0].nForce = CWeapon::InitialForce[wps[0].nKindNo];

            wps[1].nKindNo = ( nNo + 1) % WEAPON_NUM;
            wps[1].nForce = CWeapon::InitialForce[wps[1].nKindNo];
        }
        void PrintResult(int nTime)
        {
            CWarrior::PrintResult(nTime,NINJA);
            printf("It has a %s and a %s\n",
                CWeapon::Names[wps[0].nKindNo],
                CWeapon::Names[wps[1].nKindNo]);
        }
};
class CIceman:public CWarrior
{
    private:
        CWeapon wp;
    public:
        CIceman( CHeadquarter * p,int nNo_):
            CWarrior(p,nNo_)
        {
            wp.nKindNo = nNo % WEAPON_NUM;
            wp.nForce = CWeapon::InitialForce[ wp.nKindNo ];
        }
        void PrintResult(int nTime)
        {
            CWarrior::PrintResult(nTime,ICEMAN);
            printf("It has a %s\n",
```

```
                CWeapon::Names[wp.nKindNo]);
        }
};
class CLion:public CWarrior
{
    private:
        int nLoyalty;
    public:
        void CountLoyalty()
        {
            nLoyalty = pHeadquarter ->GetTotalLifeValue();
        }
        CLion( CHeadquarter * p,int nNo_):
            CWarrior(p,nNo_) {
            CountLoyalty();
        }
        void PrintResult(int nTime)
        {
            CWarrior::PrintResult(nTime,LION);
            CountLoyalty();
            printf("It's loyalty is %d\n",nLoyalty);
        }
};
class CWolf:public CWarrior
{
    public:

        CWolf( CHeadquarter * p,int nNo_):
            CWarrior(p,nNo_) { }
        void PrintResult(int nTime)
        {
            CWarrior::PrintResult(nTime,WOLF);
        }

};

CWarrior::CWarrior( CHeadquarter * p,int nNo_) {
    nNo = nNo_;
    pHeadquarter = p;
}
void CWarrior::PrintResult(int nTime,int nKindNo)
{
        char szColor[20];
        pHeadquarter->GetColor(szColor);
```

```
        printf("%03d %s %s %d born with strength %d,%d %s in %s
headquarter\n" ,
                nTime,       szColor,       Names[nKindNo],       nNo,
InitialLifeValue[nKindNo],

    pHeadquarter->anWarriorNum[nKindNo],Names[nKindNo],szColor);
}
void CHeadquarter::Init(int nColor_, int lv)
{
    nColor = nColor_;
    nTotalLifeValue = lv;
    bStopped = false;
    nCurMakingSeqIdx = 0;
    nTotalWarriorNum = 0;
    for( int i = 0;i < WARRIOR_NUM;i ++ )
        anWarriorNum[i] = 0;
}
CHeadquarter::~CHeadquarter () {
    int i;
    for( i = 0;i < nTotalWarriorNum; i ++ )
        delete pWarriors[i];
}
int CHeadquarter::Produce(int nTime)
{
    int nSearchingTimes = 0;
    if( bStopped )
        return 0;
    while( CWarrior::InitialLifeValue[MakingSeq[nColor][nCurMakingSeqI
dx]] > nTotalLifeValue &&
        nSearchingTimes < WARRIOR_NUM ) {
        nCurMakingSeqIdx = ( nCurMakingSeqIdx + 1 ) % WARRIOR_NUM ;
        nSearchingTimes ++;
    }
    int nKindNo = MakingSeq[nColor][nCurMakingSeqIdx];
    if( CWarrior::InitialLifeValue[nKindNo] > nTotalLifeValue ) {
        bStopped = true;
        if( nColor == 0)
            printf("%03d     red     headquarter     stops     making
warriors\n",nTime);
        else
            printf("%03d     blue     headquarter     stops     making
warriors\n",nTime);
        return 0;
    }
```

```
    nTotalLifeValue -= CWarrior::InitialLifeValue[nKindNo];
    nCurMakingSeqIdx = ( nCurMakingSeqIdx + 1 ) % WARRIOR_NUM ;
    int nTmp = anWarriorNum[nKindNo];
    anWarriorNum[nKindNo] ++;
    switch( nKindNo ) {
        case DRAGON:
            pWarriors[nTotalWarriorNum]             =             new
CDragon( this,nTotalWarriorNum+1);
            break;
        case NINJA:
            pWarriors[nTotalWarriorNum]             =             new
CNinja( this,nTotalWarriorNum+1);
            break;
        case ICEMAN:
            pWarriors[nTotalWarriorNum]             =             new
CIceman( this,nTotalWarriorNum+1);
            break;
        case LION:
            pWarriors[nTotalWarriorNum]             =             new
CLion( this,nTotalWarriorNum+1);
            break;
        case WOLF:
            pWarriors[nTotalWarriorNum]             =             new
CWolf( this,nTotalWarriorNum+1);
            break;

    }
    pWarriors[nTotalWarriorNum]->PrintResult(nTime);
    nTotalWarriorNum ++;
    return 1;
}
void CHeadquarter::GetColor( char * szColor)
{
    if( nColor == 0)
        strcpy(szColor,"red");
    else
        strcpy(szColor,"blue");
}

const char * CWeapon::Names[WEAPON_NUM] = {"sword","bomb","arrow" };
int CWeapon::InitialForce[WEAPON_NUM];

const       char       *       CWarrior::Names[WARRIOR_NUM]       =
{ "dragon","ninja","iceman","lion","wolf" };
```

```
int CWarrior::InitialLifeValue [WARRIOR_NUM];
int              CHeadquarter::MakingSeq[2][WARRIOR_NUM]              =
{ { 2,3,4,1,0 },{3,0,1,2,4} };
int main()
{
    int t;
    int m;
    //freopen("war2.in","r",stdin);
    CHeadquarter RedHead,BlueHead;
    scanf("%d",&t);
    int nCaseNo = 1;
    while ( t -- ) {
        printf("Case:%d\n",nCaseNo++);
        scanf("%d",&m);
        int i;
        for(i = 0;i < WARRIOR_NUM;i ++ )
            scanf("%d", & CWarrior::InitialLifeValue[i]);
//      for(i = 0;i < WEAPON_NUM;i ++ )
//          scanf("%d", & CWeapon::InitialForce[i]);
        RedHead.Init(0,m);
        BlueHead.Init(1,m);
        int nTime = 0;
        while( true) {
            int tmp1 = RedHead.Produce(nTime);
            int tmp2 = BlueHead.Produce(nTime);
            if( tmp1 == 0 && tmp2 == 0)
                break;
            nTime ++;
        }
    }
    return 0;
}
```

****9597.cpp:

```
#include <cstdlib>
#include <iostream>
using namespace std;

int strlen(const char * s)
{   int i = 0;
    for(; s[i]; ++i);
    return i;
```

```
}
void strcpy(char * d,const char * s)
{
    int i = 0;
    for( i = 0; s[i]; ++i)
        d[i] = s[i];
    d[i] = 0;

}
int strcmp(const char * s1,const char * s2)
{
    for(int i = 0; s1[i] && s2[i] ; ++i) {
        if( s1[i] < s2[i] )
            return -1;
        else if( s1[i] > s2[i])
            return 1;
    }
    return 0;
}
void strcat(char * d,const char * s)
{
    int len = strlen(d);
    strcpy(d+len,s);
}
class MyString
{
//your code starts here
    private:
        char * str;
        int size;
    public:
        MyString(){
            str = new char[2]; //确保分配的是数组
            str[0] = 0;//既然是个字符串，里面起码也是个空串，不能让 str
== NULL
            size = 0;
        }
        MyString(const char * s) {
            //如果 s == NULL，就让它出错吧
            size = strlen(s);
            str = new char[size+1];
            strcpy(str,s);
        }
        MyString & operator=(const char * s ) {
```

```
        //如果 s == NULL，就让它出错吧
        int len = strlen(s);
        if( size < len ) {
            delete [] str;
            str = new char[len+1];
        }
        strcpy( str,s);
        size = len;
        return * this;
    }


    void duplicate(const MyString & s) {
        if( size < s.size ) { //否则就不用重新分配空间了
            delete [] str;
            str = new char[s.size+1];
        }
        strcpy(str,s.str);
        size = s.size;
    }
    MyString(const MyString & s):size(0),str(new char[1]) {
        duplicate(s);
    }
    MyString & operator=(const MyString & s) {
        if( str == s.str )
            return * this;
        duplicate(s);
        return * this;
    }

    bool operator==(const MyString & s) const {
        return strcmp(str,s.str ) == 0;
    }
    bool operator<(const MyString & s) const {
        return strcmp(str,s.str ) < 0;
    }
    bool operator>(const MyString & s) const {
        return strcmp(str,s.str ) > 0;
    }
    MyString operator + ( const MyString & s )  {
        char * tmp = new char[size + s.size + 2];//确保能分配一个数
组
        strcpy(tmp,str);
        strcat(tmp,s.str);
```

```
            MyString os(tmp);
            delete [] tmp;
            return os;
        }
        MyString & operator += ( const MyString & s) {
            char * tmp = new char [size + s.size + 2];
            strcpy( tmp,str);
            strcat( tmp,s.str);
            size += s.size;
            delete [] str;
            str = tmp;
            return * this;
        }

        char & operator[](int i) const {
            return str[i];
        }
        MyString operator()(int start,int len) const {
            char * tmp = new char[len + 1];
            for( int i = 0;i < len ; ++i)
                tmp[i] = str[start+i];
            tmp[len] = 0;
            MyString s(tmp);
            delete [] tmp;
            return s;
        }
        ~MyString() { delete [] str; }

    friend ostream & operator << ( ostream & o,const MyString & s)
    {
        o << s.str ;
        return o;
    }
    friend MyString operator +( const char * s1,const MyString & s2)
    {
        MyString tmp(s1);
        tmp+= s2;
        return tmp;
    }


//your code ends here
};
```

```
int CompareString( const void * e1, const void * e2)
{
    MyString * s1 = (MyString * ) e1;
    MyString * s2 = (MyString * ) e2;
    if( * s1 < *s2 )
    return -1;
    else if( *s1 == *s2)
    return 0;
    else if( *s1 > *s2 )
    return 1;
}
int main()
{
    MyString s1("abcd-"),s2,s3("efgh-"),s4(s1);
    MyString SArray[4] = {"big","me","about","take"};
    cout << "1. " << s1 << s2 << s3<< s4<< endl;
    s4 = s3;
    s3 = s1 + s3;
    cout << "2. " << s1 << endl;
    cout << "3. " << s2 << endl;
    cout << "4. " << s3 << endl;
    cout << "5. " << s4 << endl;
    cout << "6. " << s1[2] << endl;
    s2 = s1;
    s1 = "ijkl-";
    s1[2] = 'A' ;
    cout << "7. " << s2 << endl;
    cout << "8. " << s1 << endl;
    s1 += "mnop";
    cout << "9. " << s1 << endl;
    s4 = "qrst-" + s2;
    cout << "10. " << s4 << endl;
    s1 = s2 + s4 + " uvw " + "xyz";
    cout << "11. " << s1 << endl;
    qsort(SArray,4,sizeof(MyString),CompareString);
    for( int i = 0;i < 4;i ++ )
    cout << SArray[i] << endl;
    //s1 的从下标 0 开始长度为 4 的子串
    cout << s1(0,4) << endl;
    //s1 的从下标 5 开始长度为 10 的子串
    cout << s1(5,10) << endl;
    return 0;
}
```

****9598.cpp:

```
#include <cstdlib>
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
class MyString:public string
{
//your code starts here
    public:
    MyString():string() {};
    MyString( const char * s):string(s){};
    MyString( const string & s ): string(s){};
    MyString operator() ( int s, int l)
    {
        return substr(s,l);
    };


//your code ends here
};

int main()
{
    MyString s1("abcd-"),s2,s3("efgh-"),s4(s1);
    MyString SArray[4] = {"big","me","about","take"};
    cout << "1. " << s1 << s2 << s3<< s4<< endl;
    s4 = s3;
    s3 = s1 + s3;
    cout << "2. " << s1 << endl;
    cout << "3. " << s2 << endl;
    cout << "4. " << s3 << endl;
    cout << "5. " << s4 << endl;
    cout << "6. " << s1[2] << endl;
    s2 = s1;
    s1 = "ijkl-";
    s1[2] = 'A' ;
    cout << "7. " << s2 << endl;
    cout << "8. " << s1 << endl;
    s1 += "mnop";
```

```
    cout << "9. " << s1 << endl;
    s4 = "qrst-" + s2;
    cout << "10. " << s4 << endl;
    s1 = s2 + s4 + " uvw " + "xyz";
    cout << "11. " << s1 << endl;
    sort(SArray,SArray+4);
    for( int i = 0;i < 4;i ++ )
    cout << SArray[i] << endl;
    //s1 的从下标 0 开始长度为 4 的子串
    cout << s1(0,4) << endl;
    //s1 的从下标 5 开始长度为 10 的子串
    cout << s1(5,10) << endl;
    return 0;
}
```