字符串的定义与字符串类

```
// 析构函数，释放串的空间
String::~String(void){
	delete [] str;
}

//返回 str
char* String::get_str() const
{
	return str;
}

//返回 size
int String::get_size() const
{
	return size;
}

// 复制构造函数
String::String(const String &s)
{
	if(!s.get_str())
		str=0;
	else
	{
		size=s.get_size();
		str=new char [s.get_size()+1];
		strcpy(str, s.get_str());
	}
};

// 下标运算符重载
char& String:: operator [] (int n)
{
	if(n>size)
		n=size-1;
	return str[n];
};

// 赋值运算符重载--把一个 C++字符串赋值给当前对象
String & String:: operator = (const char *s)
{
	if(strcmp(s, str)==0)
		return *this;
	if(str)
		delete [] str;
	size=(unsigned int)strlen(s);
	str=new char [size+1];
	strcpy(str, s);
	return *this;
};

// 将当前对象与一个 C++字符串拼接
String & String:: operator + (const char *s) const
{
	// 创建一个临时串 temp
	String temp;
	// 释放创建 temp 时所申请的空间
	delete [] temp.str;
	// 为存放拼接串申请空间
	int len=size+(unsigned int)strlen(s);
	temp.str=new char [len+1];
	// 若申请空间失败，则退出
	if(temp.str==NULL)
	{
```

```
            cout<<"out of Memory"<<endl;
            exit(1);
        }
        temp.size=len;
        strcpy(temp.str, str);          // 把当前对象的私有项复制到 temp
        strcat(temp.str, s);            // 将 s 拼接到 temp.str 上
        return temp;                    // 返回
};

// 将一个 C++字符串与一个 String 串拼接
friend String operator + (char *str, const String &s)
{
        // 创建一个临时串 temp
        String temp;
        // 释放创建 temp 时所申请的空间
        delete [] temp.str;
        // 为存放拼接串申请空间
        int len=s.get_size()+(unsigned int)strlen(str);
        temp.str=new char [len+1];
        // 若申请空间失败，则退出
        if(temp.str==NULL)
        {
            cout<<"out of Memory"<<endl;
            exit(1);
        }
        temp.size=len;
        // 把当前对象的私有项复制到 temp
        strcpy(temp.str, s.get_str());
        // 将 str 拼接到 temp.str 上
        strcat(temp.str, str);
        // 返回
        return temp;
};

// 返回字符 c 最后出现的位置
int String:: FindLast(char c) const
{
        int i=size-1;
        while(i!=-1&&str[i]!=c) // 查找字符 c，则循环在串尾结束
            i--;
        return i; // 否则，返回字符 c 在当前对象中的位置
};

// 取子串函数，返回从 index 开始长度为 count 的子串
String String:: Substr ( int index, int count ) const{
        int charleft = size – index;    // 从 index 到串尾的字符数
        String temp;                    // 创建一个临时串 temp
        if ( index >= size )            // 若 index 大于串长
            return temp;                // 返回空串
        if ( count > charleft )         // 若 count 大于剩下的字符数，则子串取剩下的字符
            count = charleft;
        delete [] temp.str;             // 释放创建 temp 时产生的空串
        temp.str = new char[count + 1]; // 为子串动态申请空间
        if ( temp.str = = NULL)         // 若申请失败，则返回
            Error(ourofMemory);
        char *p = temp.str;             // 令字符指针 p 指向暂无内容的字符数组 temp.str 的首字符处
        char *q = &str[index];          // 令字符指针 q 指向当前对象的 str 数组下标 index 处
        for ( int i =0; i < count; i++)
            *p++ = *q++;                // 从 str 中复制 count 个字符到 temp.str
        *p= NULL;                       // 用 NULL 作为 temp.str 的结尾
        temp.size = count;
        return temp;
};
```

```
// 从 start 位置开始找字符 c 首次出现的位置，若找到，则返回该位置；否则返回-1.
int Find (char c, int start) const {
        if (start > = size ) // start 超过串长
                Error(outofMemory)
int i = start;
        while ( str[i] !=NULL &&str[i] != c ) // 查找字符 c，若当前对象中没有字符 c，则循环在串尾结束
                i ++;
        if ( str[i] = = NULL ) // 未找到字符 c，则返回-1
                return -1;
        return i; // 否则，返回字符 c 在当前对象中的位置
};
```