CS205 C/ C++ Programming Assignment 5

Name: 傅伟堡(Weibao Fu)

SID:11812202

Part 1-Analysis

The problem is to write some functions of UTF8string. Since the string is not a normal string, we can use some characters of UTF8 to implement the function. On the other side, we need to override the operator to implement some operators. Such that, we may need to know the knowledge of overrider.

Part 2-Code

UTF8string.cpp

```
#include"UTF8string.hpp"
#include"utf8.h"
#include<string>
using namespace std;

UTF8string::UTF8string(){

}

UTF8string::UTF8string(const char*s){
    str.clear(); //reset
    unsigned char *p = (unsigned char*)s;
    while(*p){
        str+=p[0];
        p++;
    }
    strlen = utf8_charlen((unsigned char*)s);
}

UTF8string::UTF8string(string s){
    str = s;
    strlen = utf8_charlen((unsigned char*)s.c_str());
}
```

```
UTF8string::~UTF8string(){
    str.clear();
}

string UTF8string::getstring(){
    return str;
}

unsigned long long UTF8string::length()const{
    return strlen;
}

unsigned long long UTF8string::bytes()const{
    return str.length();
}

unsigned long long UTF8string::find(const string substr)const{
    unsigned long long count = 0;
    unsigned long long index = str.find(substr);
    if(index<0) return -1;  //cannot find
    unsigned char*p = (unsigned char*)str.c_str();
    while(p!=(unsigned char*)&str[index]){
        count++;
        _utf8_incr(p);
    }
    return count;
}

void UTF8string::replace(UTF8string to_move,UTF8string to_replace){
    unsigned long long start = 0;
    unsigned char*p;
    unsigned int codepoint;
    int bytes_in_char;
    while(str.find(to_move.getstring(),start)!=string::npos){ //find target in
str
        bool judge = true;
        start = str.find(to_move.getstring(),start);
        string prestring = str.substr(0,start);      //prefix string
        p = (unsigned char*)prestring.c_str();
        //judge whether the prefix string is a valid UTF8string
        while (*p){
            codepoint = utf8_to_codepoint(p, &bytes_in_char);
            if (codepoint) {
                _utf8_incr(p);
            }else{
                judge = false;
                break;
            }
        }
        /*
        if prefix string is a valid UTF8string, we replace the string
        else we set start+=1, and continue find the next string
        */
        if(judge){
            string poststring = str.substr(start+to_move.bytes());
            str = prestring + to_replace.getstring() + poststring;
            strlen = strlen - to_move.length() + to_replace.length();
```

```
            start += to_replace.bytes();
        }else{
            start+=1;
        }
    }
}

ostream& operator<<(ostream &out,UTF8string utfstr){
    out << utfstr.getstring();
    return out;
}

UTF8string& operator+(UTF8string str1,const string str2){
    UTF8string *utf = new UTF8string(str1.getstring()+str2);
    return *utf;
}

UTF8string& operator+(UTF8string str1,UTF8string str2){
    UTF8string *utf = new UTF8string(str1.getstring()+str2.getstring());
    return *utf;
}

UTF8string& operator+(const string str1,UTF8string str2){
    UTF8string *utf = new UTF8string(str1+str2.getstring());
    return *utf;
}

void UTF8string::operator+=(const string s){
    str+=s;
    unsigned char*p = (unsigned char *)s.c_str();
    while(*p){
        strlen++;
        _utf8_incr(p);
    }
}

void UTF8string::operator+=(UTF8string s){
    str+=s.getstring();
    strlen+=s.length();
}

string operator*(UTF8string str1,const unsigned long long n){
    string s;
    for(unsigned long long i=0;i<n;i++){
        s+=str1.getstring();
    }
    return s;
}

string operator*(const unsigned long long n,UTF8string str1){
    string s;
    for(unsigned long long i=0;i<n;i++){
        s+=str1.getstring();
    }
    return s;
}

string UTF8string::operator!()const{
```

```
    string result;
    unsigned char*p = (unsigned char *)str.c_str();
    unsigned char*begin_location[length()];
    unsigned char*begin;
    unsigned char*end;
    unsigned long long cnt=0;
    // find the begin pointer location of every character
    while(*p){
        begin_location[cnt++]=p;
        _utf8_incr(p);
    }
    //swap
    for(unsigned long long i=0;i<cnt/2;i++){
        unsigned char*temp = begin_location[i];
        begin_location[i] = begin_location[cnt-i-1];
        begin_location[cnt-i-1] = temp;
    }
    //traverse and add every bytes into our string
    for(int i=0;i<cnt;i++){
        begin = begin_location[i];
        _utf8_incr(begin_location[i]);
        end = begin_location[i];
        while(begin!=end){
            result+=*begin;
            begin++;
        }
    }
    return result;
}
```

UTF8string.hpp

```
#ifndef UTF8STRING_HPP
#define UTF8STRING_HPP
#include<string>

class UTF8string{
    private:
        unsigned long long strlen;
        std::string str;

    public:
        //constructor&destructor
        UTF8string();
        UTF8string(const char *s);
        UTF8string(const std::string);
        ~UTF8string();
        std::string getstring();
        //function
        unsigned long long length()const;
        unsigned long long bytes()const;
        unsigned long long find(const std::string substr)const;
        void replace(const UTF8string to_move,const UTF8string to_replace);
        //overload +=
        void operator+=(const UTF8string str);
        void operator+=(const std::string str);
        //overload !
```

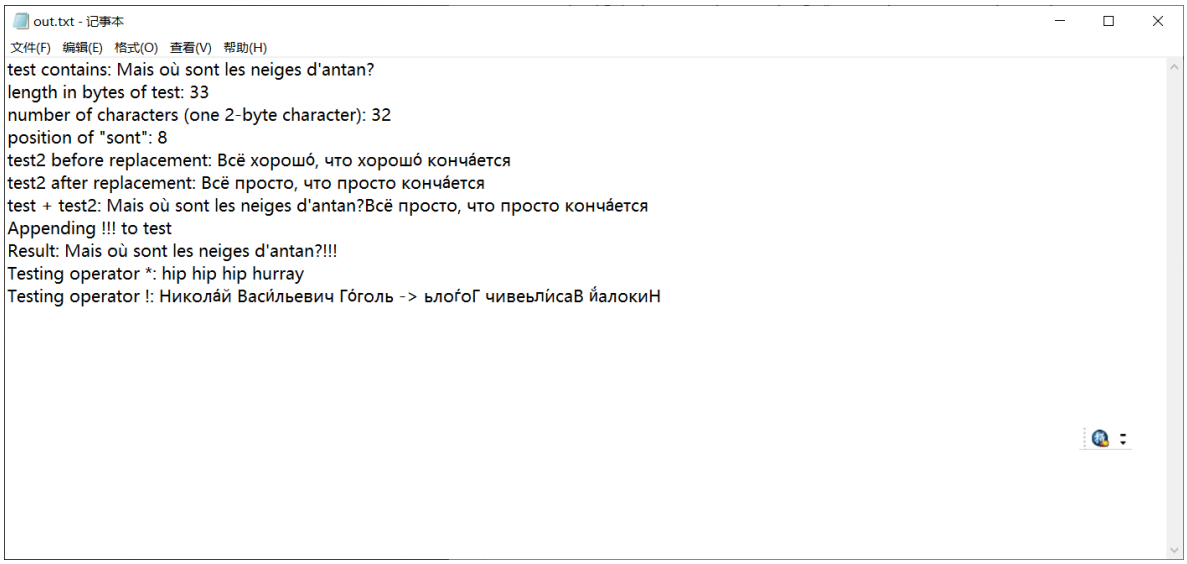
```
        std::string operator!()const;
};
        //overload <<
        std::ostream& operator<<(std::ostream& out, const UTF8string str);
        //overload +
        UTF8string& operator+(const UTF8string str1,const UTF8string str2);
        UTF8string& operator+(const std::string str1,const UTF8string str2);
        UTF8string& operator+(const UTF8string str1,const std::string str2);
        //overload *
        std::string operator*(const unsigned long long n,const UTF8string str1);
        std::string operator*(const UTF8string str1,const unsigned long long n);

#endif
```

Part 3-Result & Verification

Test case #1

```
Input: g++ UTF8string.cpp utf8.c testUTF8string.cpp -o demo
       ./demo>out.txt
Output: test contains: Mais où sont les neiges d'antan?
        length in bytes of test: 33
        number of characters (one 2-byte character): 32
        position of "sont": 8
        test2 before replacement: Всё хорошо́, что хорошо́ конча́ется
        test2 after replacement: Всё просто, что просто конча́ется
        test + test2: Mais où sont les neiges d'antan?Всё просто, что просто
конча́ется
        Appending !!! to test
        Result: Mais où sont les neiges d'antan?!!!
        Testing operator *: hip hip hip hurray
        Testing operator !: Никола́й Васи́льевич Го́голь -> ьлоґоГ чивеьли́саВ
йалокиН
```



Part 4-Difficulties & Solutions

1.We set three forms to implement constructors, such that we can support parameter `string` and `pointer` to construct.

2.Since we may need to visit private member, we set a function `getstring`, and `length()` and `bytes()` to implement it.

3.Since we need to find a string in given UTF8string and give the position, we use `find()` to find the position in bytes and then use `_utf8_incr()` to find the position in character.

4.Since we need to implement `replace()` , first we need to use `find()` to find the string. Then we need to check if the string is a valid UTF8string. If the string is a valid UTF8string, then we replace it, else we do nothing.

5.When we need to override operator, we just need to override it in normal way.