

H1 CS205 C/ C++ Programming - Lab Assignment 5

Name: 邱煜 (Qiu Yu)

SID: 11611127

H2 Part1 - Analysis

This assignment is to create a class called `UTF8string` which knows "characters". I use the functions in `uf8.c` to implements these required functions.

H2 Part2 - Code

H3 UTF8string.cpp

```
1  #include <iostream>
2  #include <cstring>
3  #include <cstdlib>
4  #include <vector>
5  #include <algorithm>
6  #include "UTF8string.hpp"
7  #include "utf8.h"
8
9  using namespace std;
10
11 // empty constructor
12 UTF8string::UTF8string(){
13     str = "";
14     size = 0;
15     byte = 0;
16 }
17
18 // only string constructor
19 UTF8string::UTF8string(string rawStr) {
20     str = rawStr;
21     byte = rawStr.length();
22
23     int lenPtr = 0;
24     int readLen = 0;
25     int codePt;
26     unsigned char *pt = (unsigned char *)str.c_str();
27     size = 0;
28     while (readLen < byte)
```

```

29     {
30         lenPtr = 0;
31         codePt = utf8_to_codepoint(pt, &lenPtr);
32         pt += lenPtr;
33         readLen += lenPtr;
34         size++;
35         eachChar.push_back(codePt);
36         byteNum.push_back(lenPtr);
37     }
38 }
39
40 UTF8string::UTF8string(const char rawStr[]){
41     str = rawStr;
42     byte = str.length();
43
44     int lenPtr = 0;
45     int readLen = 0;
46     int codePt;
47     unsigned char *pt = (unsigned char *)str.c_str();
48     size = 0;
49     while (readLen < byte) {
50         lenPtr = 0;
51         codePt = utf8_to_codepoint(pt, &lenPtr);
52         pt += lenPtr;
53         readLen += lenPtr;
54         size++;
55         eachChar.push_back(codePt);
56         byteNum.push_back(lenPtr);
57     }
58 }
59
60 // partly constructor
61 UTF8string::UTF8string(string rawStr, int rawSize, int
rawByte){
62     str = rawStr;
63     size = rawSize;
64     byte = rawByte;
65
66     int lenPtr = 0;
67     int readLen = 0;
68     int codePt;
69     unsigned char *pt = (unsigned char *)str.c_str();
70     while (readLen < byte) {
71         lenPtr = 0;
72         utf8_to_codepoint(pt, &lenPtr);
73         pt += lenPtr;
74         readLen += lenPtr;

```

```

75         eachChar.push_back(codePt);
76         byteNum.push_back(lenPtr);
77     }
78 }
79
80 //full constructor
81 UTF8string::UTF8string(string rawStr, int rawSize, int
    rawByte, vector<int> rawByteNum, vector<int> rawEachChar){
82     str = rawStr;
83     size = rawSize;
84     byte = rawByte;
85     byteNum = rawByteNum;
86     eachChar = rawEachChar;
87 }
88
89 // overload <<
90 ostream &operator<<(ostream & os, const UTF8string &
    utf8string){
91     os << utf8string.str;
92     return os;
93 }
94
95 // overload +
96 UTF8string UTF8string::operator+(const UTF8string &a) {
97     UTF8string result(this->str+a.str, this->size+a.size,
    this->byte+a.byte);
98     result.byteNum.reserve(result.size+a.size+1);
99     result.byteNum.insert(result.byteNum.end(),
    a.byteNum.begin(), a.byteNum.end());
100     result.eachChar.reserve(result.size+a.size+1);
100     result.eachChar.insert(result.eachChar.end(),
    1 a.eachChar.begin(), a.eachChar.end());
100     return result;
100 }
100
100 // overload +=
100 void UTF8string::operator+=(const UTF8string &a){
100     *this = *this + a;
100 }
100
100 // overload *
100 UTF8string UTF8string::operator*(const int num) {
101     UTF8string result;
102     for (int i = 0; i < num; i++) {
103         result += *this;
104     }
105     return result;

```

```

16 }
17 UTF8string operator*(int num, const UTF8string &str) {
18     UTF8string result;
19     for (int i = 0; i < num; i++) {
20         result += str;
21     }
22     return result;
23 }
24
25 // overload !
26 UTF8string UTF8string::operator!(){
27     UTF8string result("", this->size, this->byte, this->byteNum, this->eachChar);
28     reverse(result.eachChar.begin(), result.eachChar.end());
29     reverse(result.byteNum.begin(), result.byteNum.end());
30     unsigned char *p = (unsigned char *)malloc(sizeof(unsigned char) * 5);
31     for (int i = 0; i < size; i++) {
32         result.str += (char *)codepoint_to_utf8(result.eachChar[i], p);
33     }
34     return result;
35 }
36
37 // return character length
38 int UTF8string::length() { return size; }
39
40 // return byte length
41 int UTF8string::bytes() { return byte; }
42
43 // return the character position where substr starts. -1 if not found
44 int UTF8string::find(string substr){
45     int charIdx = 0;
46     int nowByte = 0;
47     int byteIdx = str.find(substr);
48     if(byteIdx == -1){ return byteIdx; }
49     for(int i = 0; i < size; i++){
50         if(nowByte < byteIdx){
51             nowByte += byteNum[i];
52             charIdx++;
53         }else{ break; }
54     }
55     return charIdx;
56 }
57
58 // replace the to_remove with replacement

```

```

10 int UTF8string::replace(UTF8string to_remove, UTF8string
0 replacement){
16     int byteIdx = str.find(to_remove.str());
16     if(byteIdx == -1){ return byteIdx; }
10     str = str.replace(byteIdx, to_remove.str.length(),
3 replacement.str);
16
14     size -= to_remove.length();
15     size += replacement.length();
16     byte -= to_remove.bytes();
10     byte += replacement.bytes();
18
19     int lenPtr = 0;
10     int readLen = 0;
17     int codePt;
12     unsigned char *pt = (unsigned char *)str.c_str();
13     while (readLen < byte) {
14         lenPtr = 0;
13         codePt = utf8_to_codepoint(pt, &lenPtr);
10         pt += lenPtr;
17         readLen += lenPtr;
18         eachChar.push_back(codePt);
19         byteNum.push_back(lenPtr);
10     }
18     return 1;
18 }

```

H3 UTF8string.hpp

```

1  #ifndef UTF8string_hpp
2  #define UTF8string_hpp
3
4  #include <vector>
5
6  class UTF8string {
7      public:
8          std::string str;
9          int size;
10         int byte;
11         std::vector<int> byteNum;
12         std::vector<int> eachChar;
13
14         // constructor and destructor
15         UTF8string();
16         UTF8string(std::string);

```

```

17     UTF8string(const char[]);
18     UTF8string(std::string, int, int);
19     UTF8string(std::string, int, int, std::vector<int>,
std::vector<int>);
20     ~UTF8string(){};
21
22     // overload operator
23     friend std::ostream &operator<<(std::ostream &, const
UTF8string &);
24     UTF8string operator+(const UTF8string &);
25     void operator+=(const UTF8string &);
26     UTF8string operator*(const int);
27     friend UTF8string operator*(int, const UTF8string &);
28     UTF8string operator!();
29
30     // member functions
31     int length();
32     int bytes();
33     int find(std::string);
34     int replace(UTF8string, UTF8string);
35 };
36
37 #endif

```

H2 Part 3 - Result & Verification

H3 Test case

Test case #1:

```

1  Input:
2      testUTF8string.cpp
3
4  Output:
5      test contains: Mais où sont les neiges d'antan?
6      length in bytes of test: 33
7      number of characters (one 2-byte character): 32
8      position of "sont": 8
9      test2 before replacement: Всѐ хорошó, что хорошó
кончáется
10     test2 after replacement: Всѐ просто, что хорошó
кончáется
11     test + test2: Mais où sont les neiges d'antan?Всѐ
просто, что хорошó кончáется

```

```

12     Appending !!! to test
13     Result: Mais où sont les neiges d'antan?!!!
14     Testing operator *: hip hip hip hurray
15     Testing operator !: Никола́й Васи́льевич Го́голь -> ьлоґоҔ
    чивеѣ́лисаВ ѣалокиН
16
17 Verification:
18 test contains: Mais où sont les neiges d'antan?
19 length in bytes of test: 33
20 number of characters (one 2-byte character): 32
21 position of "sont": 8
22 test2 before replacement: Всѣ хорошó, что хорошó конча́ется
23 test2 after replacement: Всѣ просто, что просто конча́ется
24 test + test2: Mais où sont les neiges d'antan?Всѣ просто,
    что просто конча́ется
25 Appending !!! to test
26 Result: Mais où sont les neiges d'antan?!!!
27 Testing operator *: hip hip hip hurray
28 Testing operator !: Никола́й Васи́льевич Го́голь -> ьлоґоҔ
    чивеѣ́лисаВ ѣалокиН

```

```

errors generated.
Personal@Coreys: ~/Desktop/C : C++/Workspace/assign_5 ➤ g++ testUTF8string.cpp UTF8string.cpp -x c utf8.c -o testU
Personal@Coreys: ~/Desktop/C : C++/Workspace/assign_5 ➤ ./testU
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: Всѣ хорошó, что хорошó конча́ется
test2 after replacement: Всѣ просто, что просто конча́ется
test + test2: Mais où sont les neiges d'antan?Всѣ просто, что хорошó конча́ется
Appending !!! to test
Result: Mais où sont les neiges d'antan?!!!
Testing operator *: hip hip hip hurray
Testing operator !: Никола́й Васи́льевич Го́голь -> ьлоґоҔ чивеѣ́лисаВ ѣалокиН
Personal@Coreys: ~/Desktop/C : C++/Workspace/assign_5 ➤

```

H2 Part 4 - Difficulties & Solutions

1. We need to store the byte information of the UTF8string. I use two `vector` to implements that. One stores the byte length of each character of the string, one stores each the code point of each character.