Data Structures and Algorithms (4)

Instructor: Ming Zhang

Textbook Authors: Ming Zhang, Tengjiao Wang and Haiyan Zhao

Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

https://courses.edx.org/courses/PekingX/04830050x/2T2014/

Outline

- The Basic Concept of Strings
- The Storage Structure of Strings
  - Sequential storage for string
  - The storage structure of class String
- The Implementation of Strings' Operations
  - The implementation of Strings' Operations
  - String class' implementation
- Pattern Matching for Strings
  - Naïve Algorithm
  - KMP Fast String Matching

Sequential storage for String

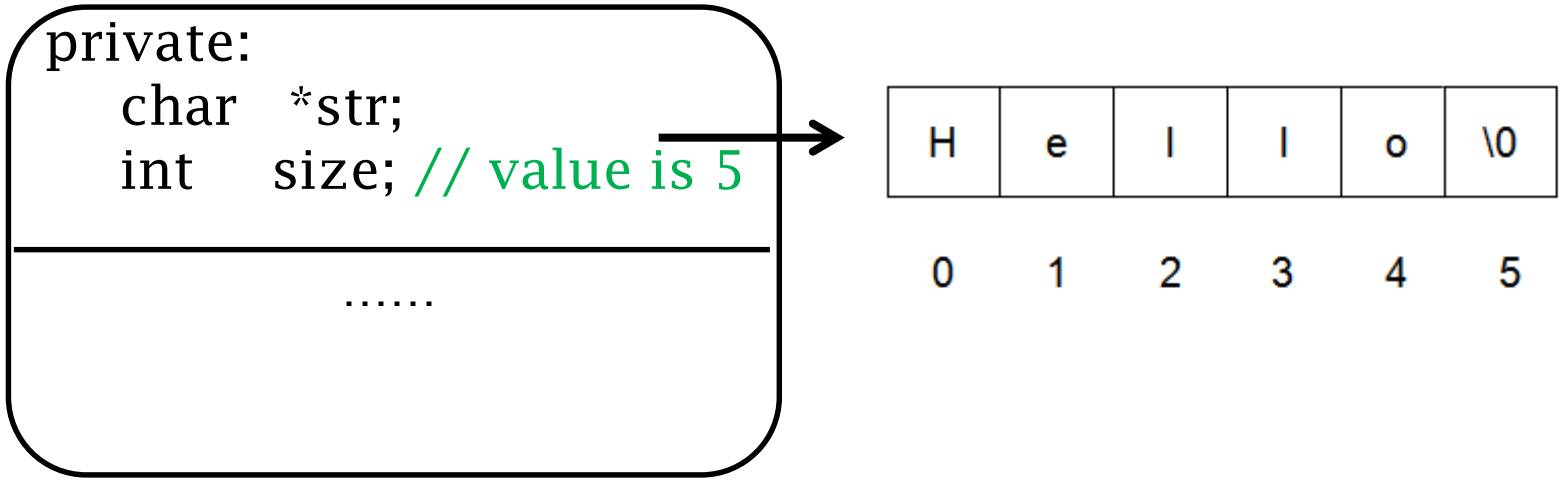
- For strings that has little change on the length, there are three processing scheme:

1. Use S[0] to record the strength of string(Pascal)
   - Disadvantage : limit the maximum length of string and make sure that it doesn't exceed 256.
2. To store the length of string, open up a new place
   - Disadvantage : The maximum length of string is static, not space dynamically applied
3. End with a special mark '\0' (C/C++)
   - For example : C/C++ string standard library (#include <string.h>)
   - '\0' ASCII char table numbered 0 , equivalent to the constant NULL、digit 0、constant false

The storage structure for String class

```
private:   //  implementation for string' storage structure
char *str; // data representation for string
int size;  // current length of string
```

For exmaple ,
String s1 = "Hello";



The implementation of string operation

- String Length
  - int strlen(char *s);
- String Copy
  - char *strcpy(char *s1, char*s2);
- String concatenation
  - char *strcat(char *s1, char *s2);
- String Comparison
  - int strcmp(char *s1, char *s2);

The implementation of string operation

```
// Get the length of string
int strlen(char d[ ]) {
   int i = 0;
   while (d[i] != '\0')
        i++;
   return i;
}
```

The implementation of string operation

```
// String copy
char *strcpy(char *d, char *s) {
    int i = 0;
    while (s[i] != '\0')   {
        d[i] = s[i];   i++;
    }
    d[i] = '\0';
    return d;
}
```

The implementation of string operation

```
// String comparison
int strcmp(const char *s1, const char *s2) {
    int i = 0;
    while (s2[i] != '\0'  &&  s1[i] != '\0' ) {
        if (s1[i] > s2[i])
            return 1;
        else if (s1[i] < s2[i])
            return -1;
        i++;
    }
    if (s1[i] == '\0' && s2[i] != '\0')
        return -1;
    else if s2[i] == '\0' && s1[i] != '\0')
        return 1;
    return 0;
}
```

More simple algorithm

```
int strcmp_1(char *s1, char *s2) {
    int i;
    for (i = 0; s1[i] == s2[i]; ++i ) {
        if(s1[i] == '\0' && s2[i] == '\0')
            return 0;              // if s1==s2
    }
    // if s1!=s2, then compare the first different
char
    return (s1[i]-s2[i]) / abs(s1[i]-s2[i]);
}
```

The implementation of string operation

```
// constructor
String::String(char *s)  {
    // Firstly, confirm the storage space needed for the new-built string,
    // the type of s is (char *)
    // As the initialized value of the new-built string.
    // Get the length of S by the standard function of string class.
     // strlen(s) get length of string
     size = strlen(s) ;
    // then, open up a space in the dynamic storage area
    // to store the initialized value of s, the end chars are also included.
     str  =  new  char [size + 1];
    // When opening up space  fails, it is abnormal, then quit
     assert(str != NULL);
    // use standard function strcpy to copy the whole string
    // to the storage space where the pointer of str points to
     strcpy(str, s);
}
```

The implementation of string operation

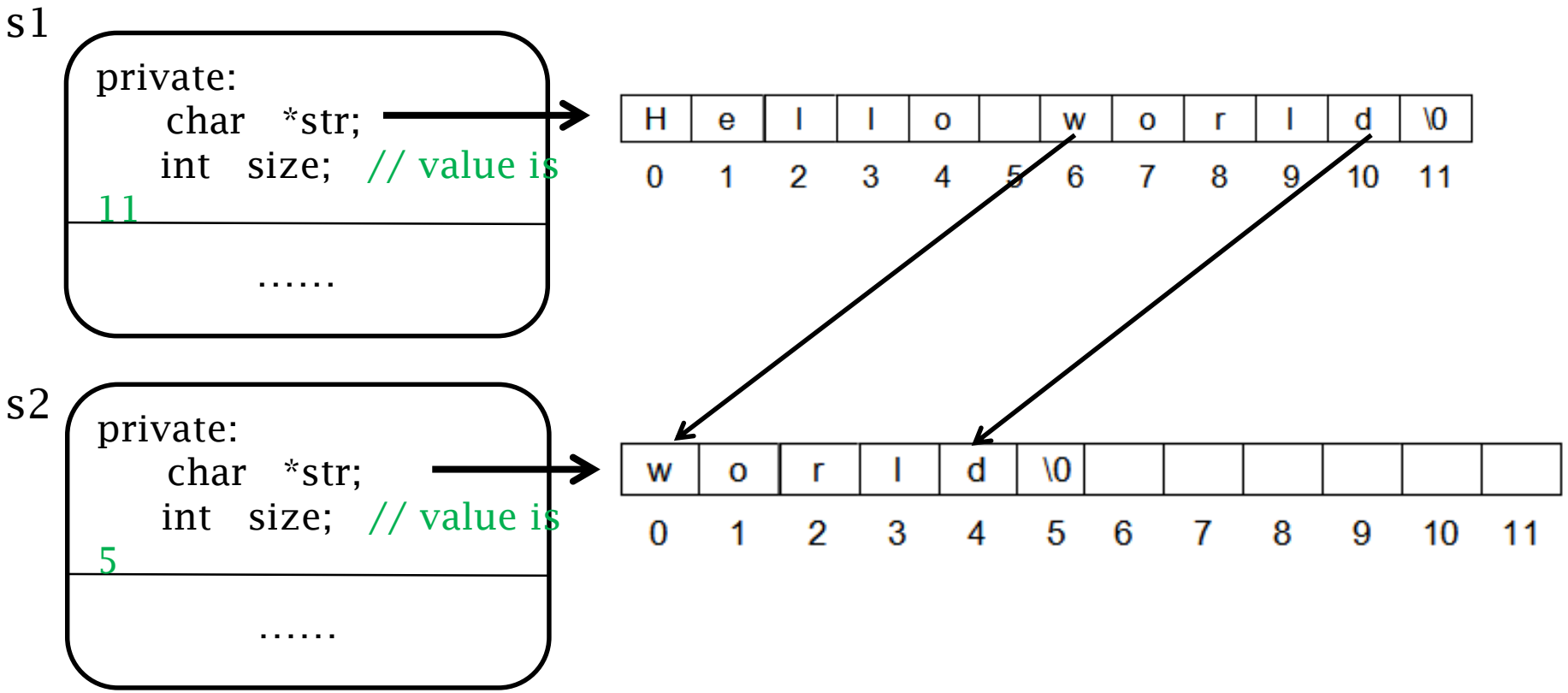
```
// Destructor
String::~String() {
   // Dynamic storage space must be released
   delete [] str;
}
```

The implementation of string operation

```
// Assign function
String  String::operator= (String&  s) {
  // String s will be assigned  to this string
  // if the length of this string is not equal to the length of s, release this
string
  //open up new spaces
  if (size != s.size)  {
    delete [] str ;       // release storage space
    str = new char [s.size+1];
    // if opening up the dynamic storage space fails, then quit.
    assert(str != NULL) ;
    size = s.size;
  }
  strcpy(str , s.str );
  //  return this instance as the instance of string class
  return  *this;
}
```

Substring extraction

- s2 = s1.Substr(6, 5) ;



Exercises

- Given strings S1 and S2, please list all possible conditions that S1+S2==S2+S1 ('+' is the concatenation operation)
- Design an algorithm to reverse the characters in a given string, without extra storage for an intermediate string.

Data Structures and Algorithms

Thanks

the National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)