C-String

- Inherited from C language
- An array of characters ends with a null character
  ```
  char cst[6] = "hello"; //c-string
  ```
  ![CharArray](image1)
  A c-string is not just an array of characters

  ```
  char cst[5] = "hello"; //c-characters
  ```
  ![CharArray](image2)
String

String is not primitive data type
It’s defined by the standard library
We may access each character in string as if it’s an array of `char`, although string is not an array of `char`.

Example:
```cpp
string st = "It is a String.";  // It is a string.
st[8] = 's';
```

- String has predefined functions contained within the class which we can use for our convenience to do string manipulations.

String Functions

- Two types of functions
  - Member functions
    Call on behave of a string data
  - Non-member functions
    Take a string as it’s argument
Member Function-[

We may see string as an array of characters.
string str1 = “hello”;

```
<table>
<thead>
<tr>
<th>h</th>
<th>e</th>
<th>l</th>
<th>l</th>
<th>o</th>
</tr>
</thead>
</table>
0 1 2 3 4
```

cout << str1[0]; //first character of str1
Non-member overloads

- operator>>
- operator<<
- operator+
- getline
#include <iostream>
using namespace std;
int main(){
    string str1, str2;
    cin >> str1;
    cout << str1;
    cout << str2 = str1 + str1; // concatenate 2 strings
    getline(cin, str1);
    return 0;
}
Member Functions

Call on behave of the string object
string_var.mem_fun()

operator[]
length()
insert()
substr()
find()
rfind()
c_str()
replace()
Member Function-length()

To identify the length of the string.
string st = "hello";

<table>
<thead>
<tr>
<th>h</th>
<th>e</th>
<th>l</th>
<th>l</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

cout << "Characters count: " << st.length();
Member Function - insert()

- String also allows us to insert some text into part of the string, instead of append the text to the end of a string. We use the insert function to manipulate the string.

Model: `str_var.insert(int index_pos, string text_tobe_insert);`

`index_pos`: the starting position in the `str_var` where you want the text to go, then push all the characters after the `index_pos` in the original `str_var` back after the inserted text.

`text_tobe_insert`: the text you want to insert into the `str_var` string

`st = “goodbye”;`

```
g o o d b y e
0 1 2 3 4 5 6
```

`st.insert(4, “ “);`

```
g o o d  b y e
0 1 2 3 4 5 6 7
```
Practice insert

How do we update string st “NY” to “New York”
string st = “NY”;

How do we update string st “NY” to “New York”
string st = “NY”;

st.insert(1, “ew ”);

//add to the end of st, instead of counting out the index manually
//we may ask the computer to compute
st.insert(st.length(), “ork”);
Member Function - substr()

- To get a sub-string from the original string.
- This function has 2 signature

Model 1: str_var.substr(int start_index);
Model 2: str_var.substr(int start_idx, int char_count);

string st = “good bye”;

```
g o o d b y e
0 1 2 3 4 5 6 7
```

cout << st.substr(3); // d bye
cout << st.substr(5, 2) // by
cout << st.substr(st.length() - 1, 1);
cout << st[st.length() - 1];
Practice substr

Update st to remove first character. Sample st = “BRYAN”;

//Get substring RYAN from st
//update st
st = st.substr(1);
If `st` has even number of characters, insert the middle 2 characters in the middle of the string.

```
// How to get the index of first middle character?
// `st.length() / 2 - 1`
// Where should we start insert?
// `st.length() / 2`
if (st.length() % 2 == 0)
    st.insert(st.length() / 2, st.substr(st.length() / 2 - 1, 2));
```
Member Function-find()

- Searches the string for the **first** occurrence of the sequence specified by its arguments.
- It’s **not** a boolean function.
- It returns the index of first match, -1 for not found.

Model 1: str_var.find(string sub_string_to_search);

```cpp
string st = "good bye";
```

<table>
<thead>
<tr>
<th>g</th>
<th>o</th>
<th>o</th>
<th>d</th>
<th>b</th>
<th>y</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

cout << "find first o at index: " << st.find("o");
```
```
//output: 1
Member Function- rfind()

- Searches the string for the **last** occurrence of the sequence specified by its arguments.
- It’s **not** a boolean function.
- It returns the index of first match, -1 for not found

Model 1: str_var.rfind(string sub_string_to_search);

String st = “good bye”;

<table>
<thead>
<tr>
<th>g</th>
<th>o</th>
<th>o</th>
<th>d</th>
<th>b</th>
<th>y</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

cout << “find last o at index: ” << st.rfind(“o”);
//output 2
Member Function-replace()

- Update string
- It takes 3 arguments

Model 1: str_var.replace(index, charCount, newString);

- index: of the first character to be replaced
- charCount: number of characters to replace
- newString: value copied to str_var at index

- Replace the portion of the string that begins at character index and spans charCount characters, by newString

String str1 = “hello”;

<table>
<thead>
<tr>
<th>h</th>
<th>e</th>
<th>l</th>
<th>l</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

str1.replace(1, 2, “a”);
//str1 becomes halo
Since C++ is inherited from C, there’re functions taking c-string instead of string.

str_var.c_str() \rightarrow returns str_var value as a c-string

File f;
string filename = “word.txt”;
f.open(filename.c_str());

char cst[] = “c-string data”;
string st = cst;
Function

// Write recursive function isPalindrome
// take the string it checks as parameter

bool isPalindrome(string st) {
    // terminate case, string st has 1 or 0 chars
    if (st.length() <= 1) return true;
    if (st[0] != st[st.length() - 1]) return false;
    return isPalindrome(st.substr(1, st.length() - 2));
}