Chapter 5

Strings

Learning Objectives

- An Array Type for Strings
  - C-Strings
- Character Manipulation Tools
  - Character I/O
  - get, put member functions
  - putback, peek, ignore
- Standard Class string
  - String processing

Introduction

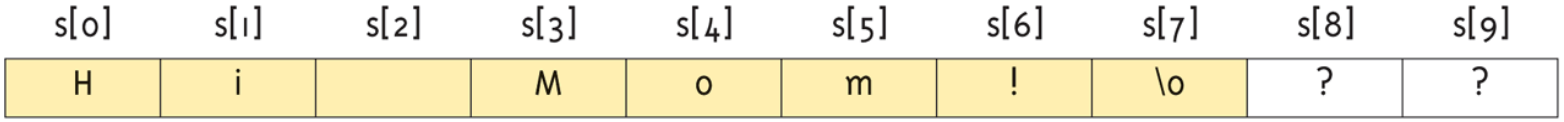
- Two string types:
- C-strings
  - Array with base type char
  - End of string marked with null, "\0"
  - "Older" method inherited from C
- String class
  - Uses templates

C-Strings

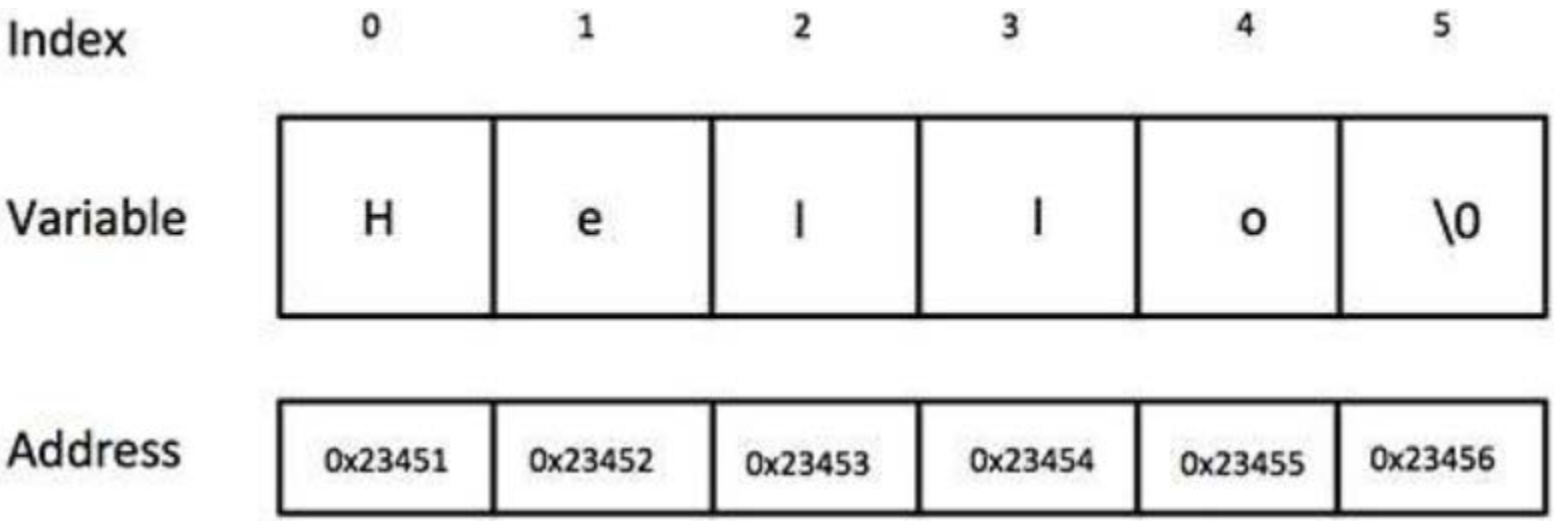
- Array with base type *char*
  - One character per indexed variable
  - One extra character: "\0"
    - Called "null character"
    - End marker
- We've used c-strings
  - Literal "Hello" stored as c-string

C-String Storage

- A standard array:
  char s[10];
  - If s contains string "Hi Mom!", stored as:



- char greeting[] = "Hello";



C-String Initialization

- Can initialize c-string:
  char myMessage[20] = "Hi there.";
  - Needn't fill entire array
  - Initialization places "\0" at end
- Can omit array-size:
  char shortString[] = "abc";
  - Automatically makes size one more than length of quoted string
  - NOT same as:
    char shortString[] = {'a', 'b', 'c'};

C-String Indexes

- A c-string IS an array
- Can access indexed variables of:
  char ourString[5] = "Hi";
  - ourString[0] is "H"
  - ourString[1] is "i"
  - ourString[2] is "\0"
  - ourString[3] is unknown
  - ourString[4] is unknown

Library

- Declaring c-strings
  - Requires no C++ library
  - Built into standard C++
- Manipulations
  - Require library <cstring>
  - Typically included when using c-strings
    - Normally want to do "fun" things with them

= and == with C-strings

- C-strings are not like other variables
  - Cannot assign or compare:

```
char aString[10];
aString = "Hello";          // ILLEGAL!
```

    - Can ONLY use "=" at declaration of c-string!
- Must use library function for assignment:
  strcpy(aString, "Hello");
  - Built-in function (in <cstring>)
  - Sets value of aString equal to "Hello"
  - NO checks for size!
    - Up to programmer, just like other arrays!

Comparing C-strings

- Also cannot use operator ==

```
char aString[10] = "Hello";
char bString[10] = "Goodbye";
```

  - aString == bString; // NOT allowed! No error message!

- Must use library function again:

```
if (strcmp(aString, bString))
      cout << "Strings NOT same.";
else
      cout << "Strings are same.";
```

  - *The comparison is true if the strings do not match.*

C-string Functions: strlen()

- "String length"
- Often useful to know string length:
  char myString[10] = “guesswhat";
  cout << strlen(myString);
  - Returns number of characters
    - Not including null
  - Result here:
    6

C-string Functions: strcat()

- strcat()
- "String concatenate":
  char stringVar[20] = "The rain";
  strcat(stringVar, "in Spain");
  - Note result:
    stringVar now contains "The rainin Spain"
  - Be careful!
  - Incorporate spaces as needed!

C-string Arguments and Parameters

- Recall: c-string is an array
- So c-string parameter is array parameter
  - C-strings passed to functions can be changed by receiving function!
- Like all arrays, typical to send size as well
  - Function "could" also use "\0" to find end
  - So size not necessary if function won't change c-string parameter
  - Use "const" modifier to protect c-string arguments

C-String Input Example

- char a[80], b[80];
  cout << "Enter input: ";
  cin >> a >> b;
  cout << a << b << "END OF OUTPUT\n";
- Dialogue offered:

  Enter input: Today Friday Fun soon

  TodayFridayEND OF OUTPUT

  – Note: Underlined portion typed at keyboard
- C-string *a* receives: “Today"
- C-string *b* receives: “Friday"

C-String Line Input

- Can receive *entire line* into a c-string
- Use getline(), a predefined member function:

```
char a[80];
cout << "Enter input: ";
cin.getline(a, 80);
cout << a << "END OF OUTPUT\n";
```

  - Dialogue:
    Enter input: Today Friday Fun soon
  - Today Tuesday FunEND OF INPUT

Standard Class string

- Defined in library:
  #include <string>
  using namespace std;
- String variables and expressions
  - Treated much like simple types
- Can assign, compare, add:
  string s1, s2, s3;
  s3 = s1 + s2; //Concatenation
  s3 = "Hello Mom!" //Assignment
  - Note c-string "Hello Mom!" automatically converted to string type!

Program Using the Class string

Display 9.4 Program Using the Class string

```
1  //Demonstrates the standard class string.
2  #include <iostream>
3  #include <string>
4  using namespace std;

5  int main( )
6  {
7       string phrase;
8       string adjective("fried"), noun("ants");
9       string wish = "Bon appetite!";

10      phrase = "I love " + adjective + " " + noun + "!";
11      cout << phrase << endl
12           << wish << endl;

13      return 0;
14 }
```

Initialized to the empty string.

Two equivalent ways of initializing a string variable

SAMPLE DIALOGUE

I love fried ants!
Bon appetite!

Class string Processing

- Same operations available as c-strings
- And more!
  - Over 100 members of standard string class
- Some member functions:
  - .length()
    - Returns length of string variable
  - .at(i)
    - Returns reference to char at position i