Chapter 9. 스트링

박 종 혁 교수
UCS Lab
(http://www.parkjonghyuk.net)

Tel: 970-6702
Email: jhpark1@snut.ac.kr

Learning Objectives

- 스트링을 위한 배열 형식
  - C-스트링
- 문자 조작 도구
  - 문자 I/O
  - get, put 멤버 함수
  - putback, peek, ignore
- 표준 스트링 클래스
  - 스트링 프로세스

개요

- 두 개의 스트링 형식:
- C-스트링
  - 문자형 배열
  - 배열의 끝을 마킹 → null, "\0"
  - C언어에서 기인된 전통적인 방법
- 스트링 클래스
  - 템플릿 사용

C-스트링

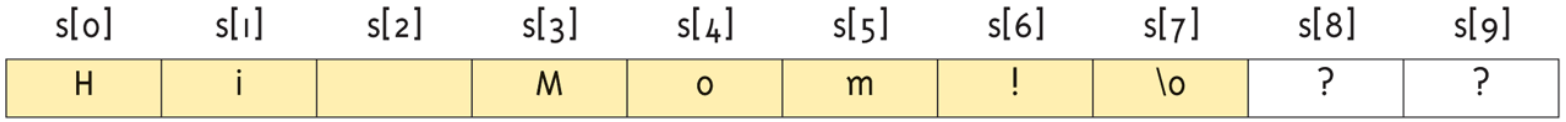
- 문자형 배열
  - 인덱스 변수당 하나의 문자
  - extra character: "\0"
    - 널 문자
    - 문자열의 끝을 표시 : 종료표시자 (End Marker)
- C언어부터 사용해 오던 방식
  - “Hello” : C-스트링으로 저장된 문자 상수

C-스트링 변수

- 문자형 배열:
  char s[10];
  - 9개 문자를 위한 C-스트링 변수 선언
  - 널 문자 추가 됨
- 일반적으로 부분적으로 채워진 배열
  - 최대 스트링 크기를 여유 있게 선언
  - 스트링의 끝은 널 문자로 마킹
- 일반 배열과 다른점:
  - 반드시 널 문자를 포함해야 함

C-스트링 저장

- char s[10];
  - "Hi Mom!" 의 저장:



C-스트링 초기화

- C-스트링 초기화:
  char myMessage[20] = "Hi there.";
  - 모든 배열 요소를 채울 필요 없음
  - 배열의 마지막에 "\0" 위치
- 배열의 크기를 명시하지 않을 경우:
  char shortString[] = "abc";
  - 자동으로 문자열보다 하나 큰 크기로 설정됨
  - 다음의 예와 다름:
    char shortString[] = {‘a’, ‘b’, ‘c’};

C-스트링 인덱스

- C-스트링은 배열
- 인덱스 변수로 접근이 가능:
  char ourString[5] = "Hi";
  - ourString[0] → "H"
  - ourString[1] → "i"
  - ourString[2] → "\0"
  - ourString[3] → unknown
  - ourString[4] → unknown

C-스트링 인덱스 조작

- 인덱스 변수 단위의 조작이 가능
  char happyString[7] = "DoBeDo";
  happyString[6] = "Z";
  - 주의!
  - 여기서, '\0'(null) 이 'Z'로 대치됨
- 널 문자가 유지되지 못하면 더 이상 C-스트링으로 동작하지 않음!
  - 예상치 못한 결과!

라이브러리

- C-스트링 선언
  - C++ 라이브러리가 필요 없음
  - 표준 C++에 Built in 되어있음
- 조작
  - <cstring> 라이브러리 필요
  - #include <cstring>
  - 전역 네임스페이스에 정의되어 있다 - 별도의 using 문장이 필요 없다.

C-스트링의 할당(=)과 비교(==)

- C-스트링은 다른 변수와 다름
  - 할당과 비교 연산이 불가:
    ```
    char aString[10];
    aString = "Hello";          // ILLEGAL!
    ```
    - C-스트링 선언 할 때 “=” 사용 가능! : 변수의 초기화 구문
- 할당을 위해 라이브러리 함수를 사용 해야 함:
  ```
  strcpy(aString, "Hello");
  ```
  - <cstring>의 라이브러리 함수
  - aString에 "Hello“가 할당
  - 크기 체크하지 않음!
    - 다른 배열과 마찬가지로 프로그래머의 몫!
    - strcpy함수의 세번째 인자로 복사되는 문자열의 크기 지정.

C-스트링의 비교

- == 연산자 사용 불가

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

  - aString == anotherString; // NOT allowed!

- 라이브러리 함수를 사용해야 함:

```
if (strcmp(aString, anotherString))
        cout << "Strings NOT same.";  //true
else
        cout << "Strings are same.";  //false
```

<cstring> 라이브러리: 디스플레이 9.1 <cstring>에서 사전 정의된 C-스트링 함수 (1/2)

- Full of string manipulation functions

Display 9.1 Some Predefined C-String Functions in <cstring>

FUNCTION	DESCRIPTION	CAUTIONS
`strcpy(`*Target_String_Var*`,` *Src_String*`)`	Copies the C-string value *Src_String* into the C-string variable *Target_String_Var*.	Does not check to make sure *Target_String_Var* is large enough to hold the value *Src_String*.
`strcpy(`*Target_String_Var*`,` *Src_String*`,` *Limit*`)`	The same as the two-argument `strcpy` except that at most *Limit* characters are copied.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcpy`. Not implemented in all versions of C++.
`strcat(`*Target_String_Var*`,` *Src_String*`)`	Concatenates the C-string value *Src_String* onto the end of the C-string in the C-string variable *Target_String_Var*.	Does not check to see that *Target_String_Var* is large enough to hold the result of the concatenation.

(continued)

<cstring> 라이브러리: 디스플레이 9.1 <cstring>에서 사전 정의된 C-스트링 함수 (2/2)

Display 9.1 Some Predefined C-String Functions in <cstring>

FUNCTION	DESCRIPTION	CAUTIONS
`strcat(`*Target_String_Var*`,` *Src_String, Limit*`)`	The same as the two argument `strcat` except that at most *Limit* characters are appended.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcat`. Not implemented in all versions of C++.
`strlen(`*Src_String*`)`	Returns an integer equal to the length of *Src_String*. (The null character, `'\0'`, is not counted in the length.)	
`strcmp(`*String_1*`,` *String_2*`)`	Returns 0 if *String_1* and *String_2* are the same. Returns a value < 0 if *String_1* is less than *String_2*. Returns a value > 0 if *String_1* is greater than *String_2* (that is, returns a nonzero value if *String_1* and *String_2* are different). The order is lexicographic.	If *String_1* equals *String_2*, this function returns 0, which converts to `false`. Note that this is the reverse of what you might expect it to return when the strings are equal.
`strcmp(`*String_1*`,` *String_2, Limit*`)`	The same as the two-argument `strcat` except that at most *Limit* characters are compared.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcmp`. Not implemented in all versions of C++.

C-스트링 함수: strlen()

- 스트링의 길이를 리턴:

```
char myString[10] = "dobedo";
cout << strlen(myString);
```

  - 문자의 수를 리턴 함
    - 널 문자는 포함되지 않음
  - 결과 → 6

C-스트링 함수: strcat()

- strcat()
- 스트링 연결:

  char stringVar[20] = "The rain";
  strcat(stringVar, "in Spain");

  – 결과:
  "The rainin Spain"
  – 주의! → 문자열을 모두 수용할 수 있는 공간이 필요!

C-스트링 인자와 매개변수

- C-스트링은 배열
- 따라서, C-스트링 매개변수는 배열 매개변수와 동일
  - 함수에 전달된 C-스트링은 함수에서 수정이 가능!
- 다른 배열과 동일하게 일반적으로 배열크기 전달
  - 마지막을 찾기 위해 함수내부에서 널 문자 이용
  - 함수내부에서 C-스트링 매개변수의 변경이 없다면 크기는 필요 없음
  - C-스트링 매개변수 보호 → const

C-스트링 출력

- << 이용
- 이미 사용:
  cout << news << " Wow.\n";
  - 여기서 news는 C-스트링 변수
- 가능한 이유 → << 는 이미 C-스트링을 위해 오버로딩 되어있음!

C-스트링 입력

- >> 이용
  - 이슈가 존재
- 공백문자 → delimiter(구획 문자)
  - Tab, space, line break
  - Delimiter 에서 읽기 중단
- C-스트링 크기 주의
  - 스트링을 위해 충분한 공간을 확보해야 함!
  - C++에서 공간에 대한 경고가 없음!

C-스트링 입력 예제

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

  Enter input: Do be do to you!
  DobeEND OF OUTPUT
- C-스트링 *a* ← "Do"
- C-스트링 *b* ← "be"

C-스트링 라인 입력

- C-스트링에 라인 단위의 입력이 가능
- getline() 사용, 사전 정의된 멤버 함수:

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

  - 실행결과:

```
Enter input: Do be do to you!
Do be do to you!END OF INPUT
```

예 : 명령행 인자

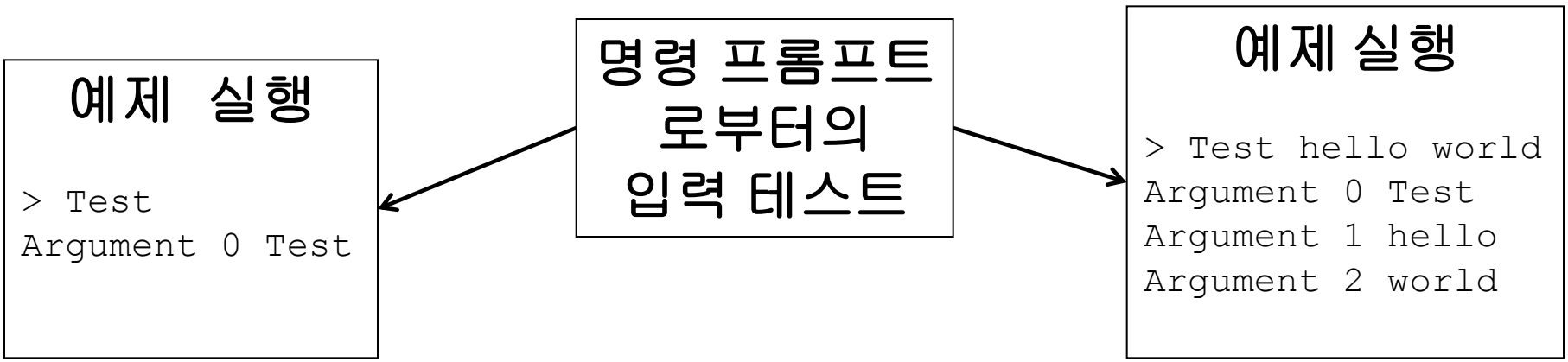
- 명령행 으로부터 실행되는 프로그램들(UNIX shell, DOS 명령프롬프트)은 인자를 보낼 수 있다
  - 예: COPY C:\FOO.TXT D:\FOO2.TXT
    - "COPY"라는 이름의 프로그램을 실행하고 2개의 C-스트링 매개변수를 보낸다 : "C:\FOO.TXT" and "D:\FOO2.TXT"
    - 제공되는 입력에 따라 COPY 프로그램을 수행한다.
- 인자들은 C-스트링의 배열로서 main 함수에 전달된다

예 : 명령행 인자

- main 함수의 헤더
  - int main(int argc, char *argv[])
    - 인자 argc는 주어진 인자의 개수를 명시하는 정수이며, 기본적으로 프로그램의 이름을 하나의 수로 간주하기 때문에 argc 값은 최소한 1이 된다
  - 인자 argv는 C-스트링의 배열
    - argv[0]는 프로그램의 이름을 명시
    - argv[1]은 첫 번째 매개변수를 명시
    - argv[2]는 두 번째 매개변수를 명시

예 : 명령행 인자

```
// Echo back the input arguments
int main(int argc, char *argv[])
{
  for (int i=0; i<argc; i++)
  {
    cout << "Argument " << i << " " << argv[i] << endl;
  }
  return 0;
}
```



getline() 멤버 함수

- 명시적인 입력 크기 지정 가능:

```
char shortString[5];
cout << "Enter input: ";
cin.getline(shortString, 5);
cout << shortString << "END OF OUTPUT\n";
```

  - 결과:

    Enter input: dobedowap
    dobeEND OF OUTPUT

  - 네 개의 문자만을 읽음
    - 하나는 널 문자를 위한 공간!

문자 I/O

- 입/출력 데이터
  - 모두 문자로 취급
  - 예) 숫자10은 ‘1’과 ‘0’ 두 개의 문자가 출력 된 것
  - 변환은 자동으로 수행
    - 저 수준 도구 사용
- C++에서는 하드웨어 수준의 저 수준 도구의 사용이 가능

get() 멤버 함수

- 한번에 하나의 문자를 읽음
- cin 객체의 멤버 함수:
  char nextSymbol;
  cin.get(nextSymbol);
  - 키보드에서 입력된 문자를 nextSymbol 변수에 저장
  - 매개변수는 반드시 문자형
    - 문자열이 아님!

put() 멤버 함수

- 한번에 하나의 문자를 출력
- cout 객체의 멤버 함수:
- 예:

```
cout.put('a');
```

  – 문자 'a'를 출력

```
char myString[10] = "Hello";
cout.put(myString[1]);
```

  – 문자 'e'를 출력

기타 멤버 함수

- putback()
  - 읽어온 문자를 되돌림
  - cin.putback(lastChar);
- peek()
  - 입력될 다음 문자를 리턴
  - peekChar = cin.peek();
- ignore()
  - 지정된 문자의 입력 생략
  - cin.ignore(1000, '\n');
    - 최대 1000개의 문자까지 '\n' 생략

문자 조작 함수: 디스플레이 9.3 <cctype> 일부 함수 (1/3)

Display 9.3 Some Functions in <cctype>

FUNCTION	DESCRIPTION	EXAMPLE
`toupper(Char_Exp)`	Returns the uppercase version of *Char_Exp* (as a value of type `int`).	`char c = toupper('a');` `cout << c;` **Outputs:** A
`tolower(Char_Exp)`	Returns the lowercase version of *Char_Exp* (as a value of type `int`).	`char c = tolower('A');` `cout << c;` **Outputs:** *a*
`isupper(Char_Exp)`	Returns `true` provided *Char_Exp* is an uppercase letter; otherwise, returns `false`.	`if (isupper(c))` `  cout << "Is uppercase.";` `else` `  cout << "Is not uppercase.";`

문자 조작 함수: 디스플레이 9.3 <cctype> 일부 함수 (2/3)

Display 9.3 Some Functions in <cctype>

FUNCTION	DESCRIPTION	EXAMPLE
`islower(Char_Exp)`	Returns `true` provided *Char_Exp* is a lowercase letter; otherwise, returns `false`.	`char c = 'a';` `if (islower(c))` `  cout << c << " is lowercase.";` **Outputs:** a is lowercase.
`isalpha(Char_Exp)`	Returns `true` provided *Char_Exp* is a letter of the alphabet; otherwise, returns `false`.	`char c = '$';` `if (isalpha(c))` `  cout << "Is a letter.";` `else` `  cout << "Is not a letter.";` **Outputs:** Is not a letter.
`isdigit(Char_Exp)`	Returns `true` provided *Char_Exp* is one of the digits `'0'` through `'9'`; otherwise, returns `false`.	`if (isdigit('3'))` `  cout << "It's a digit.";` `else` `  cout << "It's not a digit.";` **Outputs:** It's a digit.
`isalnum(Char_Exp)`	Returns `true` provided *Char_Exp* is either a letter or a digit; otherwise, returns `false`.	`if (isalnum('3') && isalnum('a'))` `  cout << "Both alphanumeric.";` `else` `  cout << "One or more are not.";` **Outputs:** Both alphanumeric.

문자 조작 함수: 디스플레이 9.3 <cctype> 일부 함수 (3/3)

`isspace(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a whitespace character, such as the blank or newline character; otherwise, returns `false`.	`//Skips over one "word" and sets c` `//equal to the first whitespace` `//character after the "word":` `do` `{` `  cin.get(c);` `} while (! isspace(c));`
`ispunct(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a printing character other than whitespace, a digit, or a letter; otherwise, returns `false`.	`if (ispunct('?'))` `  cout << "Is punctuation.";` `else` `  cout << "Not punctuation.";`
`isprint(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a printing character; otherwise, returns `false`.	
`isgraph(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a printing character other than whitespace; otherwise, returns `false`.	
`isctrl(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a control character; otherwise, returns `false`.	

표준 스트링 클래스

- 라이브러리에 정의되어 있음:
  ```
  #include <string>
  using namespace std;
  ```
- 문자열 변수와 표현
  - 기본 형과 유사하게 취급 됨
- 할당, 비교, 결합 이 가능:
  ```
  string s1, s2, s3;
  s3 = s1 + s2;            //Concatenation
  s3 = "Hello Mom!"  //Assignment
  ```
  - C-스트링 "Hello Mom!" 은 자동으로 스트링 형으로 변환됨!

디스플레이 9.4
스트링 클래스 프로그램

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!
```

스트링 클래스 I/O

- 다른 형과 동일함!
- string s1, s2;
  cin >> s1;
  cin >> s2;
- 입력:
  May the hair on your toes grow long and curly!
- 공백문자 이후 무시:
  s1 ← "May"
  s2 ← "the"

클래스 스트링과 getline()

- 라인 단위 입력을 위해:

```
string line;
cout << "Enter a line of input: ";
getline(cin, line);
cout << line << "END OF OUTPUT";
```

- 실행결과:

```
Enter a line of input: Do be do to you!
Do be do to you!END OF INPUT
```

  - C-스트링의 getline() 사용과 유사

getline() 함수의 다른 버전

- Delimiter 문자 설정:
  string line;
  cout << "Enter input: ";
  getline(cin, line, '?');
  - '?' 문자까지 입력
- getline() 은 참조를 리턴
  - string s1, s2;
    getline(cin, s1) >> s2;
  - 결과: cin >> s2;

함정: 혼합 입력

- cin >> var과 getline의 혼용에 유의
  - int n;
    string line;
    cin >> n;
    getline(cin, line);
  - 입력: 42
    Hello hitchhiker.
    - 변수 n은 42로 세팅됨
    - line은 빈 문자열로 세팅!
  - cin >> n 은 '\n'에서 읽기를 멈춤, '\n'이 getline()에 입력됨!

스트링 클래스를 이용한 스트링 처리

- C-스트링과 같은 오퍼레이션이 가능
- 향상된 버전!
  - 표준 스트링 클래스의 멤버 함수는 100개 이상이 존재
- 일부 멤버 함수:
  - .length()
    - 스트링 변수의 길이를 리턴
  - .at(i)
    - i번째 문자의 참조를 리턴

디스플레이 9.7 표준 스트링 클래스의 멤버 함수 (1/2)

Display 9.7 Member Functions of the Standard Class `string`

EXAMPLE	REMARKS
Constructors	
`string str;`	Default constructor; creates empty `string` object `str`.
`string str("string");`	Creates a `string` object with data `"string"`.
`string str(aString);`	Creates a `string` object `str` that is a copy of `aString`. `aString` is an object of the class `string`.
Element access	
`str[i]`	Returns read/write reference to character in `str` at index `i`.
`str.at(i)`	Returns read/write reference to character in `str` at index `i`.
`str.substr(position, length)`	Returns the substring of the calling object starting at `position` and having `length` characters.
Assignment/Modifiers	
`str1 = str2;`	Allocates space and initializes it to `str2`'s data, releases memory allocated for `str1`, and sets `str1`'s size to that of `str2`.
`str1 += str2;`	Character data of `str2` is concatenated to the end of `str1`; the size is set appropriately.
`str.empty( )`	Returns `true` if `str` is an empty `string`; returns `false` otherwise.

(continued)

디스플레이 9.7 표준 스트링 클래스의 멤버 함수 (2/2)

Display 9.7 Member Functions of the Standard Class `string`

EXAMPLE	REMARKS
`str1 + str2`	Returns a `string` that has `str2`'s data concatenated to the end of `str1`'s data. The size is set appropriately.
`str.insert(pos, str2)`	Inserts `str2` into `str` beginning at position `pos`.
`str.remove(pos, length)`	Removes substring of size `length`, starting at position `pos`.
Comparisons	
`str1 == str2  str1 != str2`	Compare for equality or inequality; returns a Boolean value.
`str1 < str2  str1 > str2`	Four comparisons. All are lexicographical comparisons.
`str1 <= str2  str1 >= str2`	
`str.find(str1)`	Returns index of the first occurrence of `str1` in `str`.
`str.find(str1, pos)`	Returns index of the first occurrence of string `str1` in `str`; the search starts at position `pos`.
`str.find_first_of(str1, pos)`	Returns the index of the first instance in `str` of any character in `str1`, starting the search at position `pos`.
`str.find_first_not_of (str1, pos)`	Returns the index of the first instance in `str` of any character *not* in `str1`, starting search at position `pos`.

C-스트링과 스트링 객체의 변환

- 자동 형 변환
  - C-스트링 → 스트링 객체:
    char aCString[] = "My C-string";
    string stringVar;
    stringVar = aCstring;
    - 완벽하게 허용되고 적절한 방법이다!
  - 스트링 객체 → C-스트링
    - aCString = stringVar;
    - 변환이 지원되지 않음!
    - 명시적인 변환 방법을 사용:
      strcpy(aCString, stringVar.c_str());

요약

- C-스트링 변수는 문자 배열
  - 널 문자('\0') 추가 됨
- C-스트링은 배열과 동일하게 동작
  - 할당, 비교 연산 수행 불가
- <cctype> 과 <string> 라이브러리에는 유용한 문자열 조작 함수가 존재
- cin.get() 함수는 하나의 문자를 읽음
- getline() 함수는 라인단위의 입력이 가능
- 스트링 클래스는 C-스트링 보다 향상되고 편리함

Q&A