제9장

Strings

학습 목표

- 문자 배열 형식의 Strings
  - C-Strings
- 문자 조작 도구
  - 문자 입/출력
  - get, put 멤버 함수

- (10장 끝나고) 표준 string 클래스
  - String 처리

개요

- 두 가지 형태의 string:

- C-strings
  - 문자들의 조합으로 이루어진 배열
  - 배열의 끝 부분에 '\0' 널 문자가 첨가됨
  - C 언어에서 유래된 기존의 방식

- String 클래스
  - 템플릿을 사용함

C-Strings 변수

- 문자들의 조합으로 이루어진 배열:

  예: char s[10];

  – c-strings 형식의 변수로 선언되고 최대 9개의 문자들을 저장할 수 있음
  – 배열의 끝부분에 널 문자(0, '\0', NULL)를 첨가하여, 문자열 마지막을 표시

- 일반적으로 배열의 모든 요소들의 값이 채워지지는 않음
  – 정적 배열에서는 부족하지 않을 만큼의 넉넉한 크기로 배열을 선언해야
  – 널 문자가 첨가되므로, 1byte를 추가적으로 고려해야

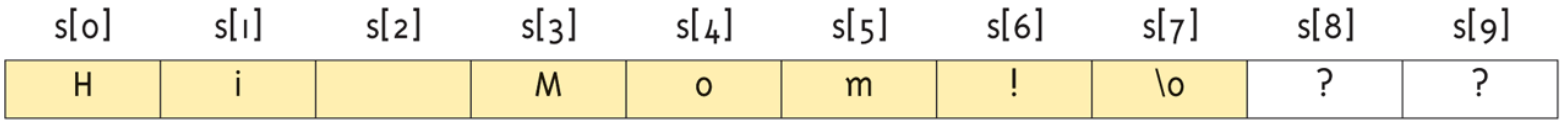
- 문자 배열 v.s. C-Strings 변수
  – C-String은 널 문자가 첨가되며 글자간 연속성에 의미가 있음
  – 문자배열은 끝에 널 문자가 필요하지 않으며, 글자간 연속성이 무의미

C-Strings 변수의 값 저장

- 일반적인 배열 선언:

  예: char s[10];

  – s 변수에 "Hi Mom!", 문자들을 저장하면 다음과 같음:



C-Strings 변수 초기화

- c-strings 변수 초기화가 가능함:

  예: char myMessage[20] = "Hi there.";
  - 배열의 모든 요소들의 값을 지정할 필요 없음
  - 초기화 단계에서 배열의 끝부분에 '\0' 문자가 삽입됨

- 배열의 크기 선언을 생략할 수 있음:

  예: char shortString[] = "abc";
  - 저장된 문자열보다 하나가 더 크게 배열의 크기가 자동적으로 결정됨
  - 따라서 이 스트링은:

    char shortString[] = {"a", "b", "c"}; 와 동일하지 않음

C-Strings 인자

- c-strings 는 배열로 해석 됨
- 배열 인자를 사용하여 배열의 요소 값을 얻을 수 있음:

  예: char ourString[5] = "Hi";
  - ourString[0] 은 "H"
  - ourString[1] 은 "i"
  - ourString[2] 은 "\0“ 문자가 저장됨을 의미하고
  - ourString[3] 은 데이터가 저장되지 않음
  - ourString[4] 은 데이터가 저장되지 않음

C-Strings 인자 다루기

- 배열의 인자를 이용하여 요소의 값을 저장할 수 있음
  - 문자 “배열 " 이므로

```
char happyString[7] = "DoBeDo";
happyString[6] = ‘Z’; // C-Strings이 아니게 됨
```

  - 주의할 점!
    - 배열 bounary에 대해서 항상 유의할 것
    - 중간에 ‘\0’을 넣으면 뒤에 내용이 사라짐

  - C-Strings 문자열 다루기(ex. 모든 문자열을 Z로 채우기)

```
for ( i = 0 ; i < SIZE ; i ++)
      myString[i] = ‘Z’ ;
```

```
for ( i = 0 ; myString[i] != ‘\0’ ; i ++)
      myString[i] = ‘Z’ ;
```

```
for ( i = 0 ; myString[i] != ‘\0’ && i < SIZE-1 ; i ++)
      myString[i] = ‘Z’ ;
myString[i] = 0 ;
```

= 및 == 연산자를 사용한 C-strings 조작

- C-strings 은 일반 변수와 다르게 작동됨
  - 할당문 또는 비교문에 사용하지 못함:

```
char aString[10];
aString = "Hello";  // 컴파일 오류!  왜??
```

  - "=" 연산자는 c-string 선언 때만 사용할 수 있음!

- 대입을 위해서는 별도의 라이브러리 함수를 사용해야 함:

  예: strcpy(aString, "Hello");
  - strcpy 함수는 <cstring> 라이브러리에 정의되어 있음
  - strcpy를 수행하면 aString의 값은 "Hello" 로 저장됨
    - 일반 배열과 다르게 String의 크기를 미리 정의할 필요 없음!

C-strings 비교

- String 비교를 위해서 == 연산자를 사용하면 틀린 결과

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

if (aString == anotherString)   // 항상 거짓!
```

- 반드시 라이브러리 함수를 사용해야 됨:

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

- c-strings 변수 사용에는 별도의 C++ 라이브러리가 필요하지 않음
  그러나 c-strings 조작을 위해서는 <cstring> 라이브러리가 필요함

<cstring> 에 정의된 C-String 함수들 (1 of 2)

- string 조작 함수

디스플레이 9.1 <cstring>에서 사전 정의된 C-스트링 함수들

함수	설명	유의사항
strcpy(*Target_String_Var, Src_String*)	C-스트링 변수인 *Src_String* 값을 *Target_String_Var* C-스트링 변수에 복사	*Target_String_Var* 변수 값이 *Src_String* 변수 값을 저장할 공간이 충분한지 여부를 점검하지 않음
strncpy(*Target_String_Var, Src_String, Limit*)	최대 *Limit* 값으로 설정된 문자 개수까지 복사된다는 것을 제외하고는 strcpy 함수와 동일함	*Limit* 값을 잘 선택하여 사용하면 strcpy 함수보다 안전하게 사용할 수 있음. 모든 C++ 컴파일러들이 지원하지 않음
strcat(*Target_String_Var, Src_String*)	*Src_String* C-스트링 변수 값을 *Target_String_Var* C-스트링 변수 값의 끝 부분에 연결함	두 스트링 변수 값을 연결한 후에, *Target_String_Var* 변수 값을 저장하는 메모리 공간이 충분한지 확인하지 않음

<cstring> 에서 정의 된 C-String 함수들 (2 of 2)

디스플레이 9.1 (계속)

함수	설명	유의사항
strncat(*Target_String_Var, Src_String, Limit*)	*Limit* 개수의 문자열을 연결한다는 사실을 제외하고는 strcat과 동일함	*Limit* 값을 잘 선택하면 strcat 함수보다 안전하게 사용할 수 있음. 모든 C++ 컴파일러들이 지원하지 않음
strlen(*Src_String*)	*Src_String* 변수 값이 차지하는 공간의 길이를 리턴함(널 문자인 경우, 공간의 길이 값을 가지지 않음)	
strcmp(*String_1, String_2*)	String_1과 String_2 값이 동일하면 0을 리턴함. String_1이 String_2 값보다 작으면 0보다 작은 값을 리턴함. String_1이 String_2 값보다 크면 0보다 큰 값을 리턴함(즉, String_1과 String_2 값이 동일하지 않으면 0이 아닌 값을 리턴함. 여기서 말하는 값의 크기는 사전적 순서에 따라 결정함)	*String_1* 값이 *String_2* 값과 동일하면 0의 값을 가지며 false를 리턴함. 이 스트링 값이 동일한 경우, 리턴되는 값과 반대로 작용함
strncmp(*String_1, String_2, Limit*)	*Limit* 개수의 문자열까지 문자들의 값을 비교한다는 사실을 제외하고는 strcat 함수와 동일함	*Limit* 값을 잘 선택하고 사용하면, strcmp 함수보다 안전하게 사용할 수 있음. 모든 C++ 컴파일러들이 지원하지 않음

C-strings 함수: strlen(str)과 strcat(str1, str2)

- strlen(str) : str의 길이(string length)를 연산할 때 사용됨:

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

  - String을 구성하는 문자의 개수를 계산함
    - 널 문자는 연산에 포함되지 않음
  - 함수 실행 결과: 6

- strcat(str1, str2): str1 뒤에 str2를 이어 줌(string concatenate)

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

  - 위의 코드 수행 결과:

    stringVar 변수는 "The rainin Spain " 을 저장함
  - 두 string들을 연결할 때 중간에 공백 문자를 적절히 삽입하는데 유의!

C-strings 인자 및 파라미터

- c-strings는 배열임을 명심할 것
- 따라서 c-strings에서 파라미터는 배열에서 파라미터와 동일함
  - C-strings은 다른 함수에 전달할 수 있고
    함수의 수행을 통하여 값이 변경될 수 있음!

- String은 배열과 같이 함수를 호출할 때 크기도 같이 명시해야 됨
  - 함수도 string의 끝부분을 인식하기 위해서 "\0" 널 문자를 이용함
  - 따라서 함수에서 c-strings 파라미터를 변경하지 않는 한
    크기를 알 필요는 없음
  - c-strings 인자의 변경을 차단하기 위해서 "const" 변경자를 사용함

C-Strings 출력과 입력

- 삽입 연산자, <<를 이용하여 출력함
  - 예를 들면: `cout << news << " Wow.\n";`
  - 여기서 *news* 는 c-string 변수임
  - c-strings 출력을 위해서 << 연산자의 기능이 오버로딩 되었음!

- 입력 연산자, >> 를 이용하여 string 입력함
- 단, 공백을 만나면 " 입력 과정이 중단됨 "
  - 탭, 여택, 라인 공백을 만나면 입력 과정은 " 중단”
- c-string의 크기에 유의할 것
  - 원하는 string을 모두 저장할 수 있도록 크기가 충분해야 됨!
  - C++ 언어에서 c-string의 크기를 초과한 데이터가 저장되는지 여부를 확인하는 기능은 없음!

C-Strings 입력 예제

```
char a[80], b[80];
cout << "Enter input: ";
cin >> a >> b;
cout << a << b << “[EOF]\n";
```

– 위의 프로그램이 다음과 같이 실행되면 :

Enter input: Do be do to you!

Dobe[EOF]

- 주의: 밑줄부분은 키보드를 통해서 사용자가 입력한 데이터를 의미함

– C-string a 는: "do " 를 저장하고

– C-string b 는 "be " 를 저장함

C-Strings 라인 입력

- 한 줄에 나열된 c-string을 입력할 수 있음
- 서전에 정의된 getline() 멤버함수를 활용함:

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

  - 위의 프로그램이 다음과 같이 수행됨:

    Enter input: Do be do to you!

    Do be do to you![EOF]

예: 인자를 포함하는 명령문 실행

- 명령문에 임자를 포함시켜서 특정 프로그램을 실행할 수 있음 (예: 유닉스 shell, DOS 명령문 실행)
- 예: COPY C:\FOO.TXT D:\FOO2.TXT
  - 여기서 “COPY” 명령문은 두 개의 C-String 파라미터들인, “C:\FOO.TXT” 와 “D:\FOO2.TXT”을 인자로 사용하여 실행됨
  - 즉 foo.txt 파일을 ffo2.txt 파일에 복사하라는 copy 명령문이 실행됨
  - 마찬가지로 C-string 형태의 인자를 포함하는 main 함수를 실행할 수 있음

예: 인자를 포함하는 명령문 실행

- Main 함수에서 사용되는 형태
  - int main(int argc, char *argv[])
  - argc 은 arguments의 개수를 의미함. 프로그램 이름자체도 argument에 속하므로 최소한 argc의 값은 1이 됨.
  - argv 은 C-String 배열 형식임.
    - argv[0]: 호출되는 함수의 이름
    - argv[1]: 호출되는 함수의 첫 번째 파라미터
    - argv[2]: 호출되는 함수의 두 번째 파라미터
    - 등등을 의미함.

예: 인자를 포함하는 명령문 실행

```
// 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;
}
```

라인 명령문

```
> Test
Argument 0 Test
```

명령문을 통하여 Test 함수가 호출됨

실행 결과

```
> Test hello world
Argument 0 Test
Argument 1 hello
Argument 2 world
```

getline() 함수 추가 설명

- String의 크기를 정확히 알 수 있음:

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

  - 위의 프로그램이 다음과 같이 실행됨:

    Enter input: dobedowap

    dobe[EOF]
  - 단지 4개의 문자들만 입력됨
    - 널 문자가 필요하다는 사실을 다시 한 번 확인하자!

문자 입/출력

- 입력 및 출력에 사용되는 모든 데이터는 문자로 해석됨
  - 예: 10 이라는 숫자는 '1' 과 '0' 의 문자들을 출력한 경우임
  - 문자와 해당되는 숫자로의 변환은 자동적으로 이루어짐
  - C++에서는 하드웨어적 차원에서 입/출력 문자 데이터를 다룰 수 있는 기능을 제공함

- get() : cin 객체의 멤버 함수로 한 글자를 입력받아 저장
  - 문자 하나를 읽은 다음에 nextSymbol 변수에 저장함
  - 함수의 인자는 반드시 문자형임

```
char nextSymbol;
cin.get(nextSymbol);
```

- put() : cout 객체의 멤버 함수로 한 글자를 출력

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

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

<cctype> 라이브러리에 정의된 일부 함수들 (1 of 3)

디스플레이 9.3 <cctype> 일부 함수들

함수	설명	예제

toupper(*Char_Exp*)	*Char_Exp* 값을 대문자로 표현함 (정수형 값으로 처리)	`char c = toupper('a');`
		`cout << c;`
		Outputs : A

tolower(*Char_Exp*)	*Char_Exp* 값을 소문자로 표현함 (정수형 값으로 처리)	`char c = tolower('A');`
		`cout << c;`
		Outputs : a

isupper(*Char_Exp*)	*Char_Exp* 값이 대문자로 표현된 경우, true 값을 리턴하고 아니면 false 값을 리턴함	`if (isupper(c))`
		`  cout << "Is uppercase.";`
		`else`
		`  cout << "Is not uppercase.";`

<cctype> 라이브러리에 정의된 일부 함수들(2 of 3)

islower(*Char_Exp*)	*Char_Exp* 값이 소문자로 표현된 경우, true 값을 리턴하고 아니면 false 값을 리턴함	`char c = 'a';`
		`if (islower(c))`
		`  cout << c << " is lowercase.";`
		Outputs : a is lowercase.

isalpha(*Char_Exp*)	*Char_Exp* 값이 알파벳 문자이면 true 값을 리턴하고 아니면 false 값을 리턴함	`char c = '$';`
		`if (isalpha(c))`
		`  cout << "Is a letter.";`
		`else`
		`  cout << "Is not a letter.";`
		Outputs : Is not a letter.

isdigit(*Char_Exp*)	*Char_Exp* 값이 '0'에서 '9' 사이의 숫자 값인 경우 true 값을, 아니면 false 값을 리턴함	`if (isdigit('3'))`
		`  cout << "It's a digit.";`
		`else`
		`  cout << "It's not a digit.";`
		Outputs : It's a digit.

isalnum(*Char_Exp*)	*Char_Exp* 값이 문자 또는 숫자이면 true 값을, 아니면 false 값을 리턴함	`if (isalnum('3') && isalnum('a'))`
		`  cout << "Both alphanumeric.";`
		`else`
		`  cout << "One or more are not.";`
		Outputs : Both alphanumeric.

<cctype> 라이브러리에 정의된 일부 함수들 (3 of 3)

`isspace(Char_Exp)`	*Char_Exp* 값이 공백 계통의 문자, 예를 들어서 공백 문자, 또는 개행 문자인 경우 `true` 값을, 아니면 `false` 값을 리턴함	`//단어 하나를 건너뛴 다음에`
		`//단어 c의 값을 건너뛴 다음에 위치하는`
		`//첫 공백 문자의 값으로 설정함:`
		`do`
		`{`
		`  cin.get(c);`
		`}`
		`while (! isspace(c));`

`ispunct(Char_Exp)`	*Char_Exp* 값이 공백 문자, 숫자 또는 알파벳 문자가 아닌 프린트 가능한 문자인 경우 `true` 값을, 아니면 `false` 값을 리턴함	`if (ispunct('?'))`
		`  cout << "Is punctuation.";`
		`else`
		`  cout << "Not punctuation.";`

`isprint(Char_Exp)`	*Char_Exp* 값이 프린트 가능한 문자이면 `true` 값을, 아니면 `false` 값을 리턴함	
`isgraph(Char_Exp)`	*Char_Exp* 값이 공백 문자가 아닌 프린트 가능한 문자이면 `true` 값을, 아니면 `false` 값을 리턴함	
`isctrl(Char_Exp)`	*Char_Exp* 값이 컨트롤 문자이면 `true` 값을, 아니면 `false` 값을 리턴함	

표준 string 클래스

- <string> 라이브러리에서 정의됨:
  - Java의 string과 동일

```
#include <string>
using namespace std;
```

- String 변수 및 표현
  - String 클래스를 이용하면 간단히 해결됨
- String 할당, 비교, 연결:

```
string s1, s2, s3;
s3 = s1 + s2;                    //string 연결
s3 = "Hello Mom!"                // string 할당
```

  - 여기서 "Hello Mom!" c-string은 자동적으로 string 형으로 변환됨!

디스플레이 9.4. string 클래스를 이용한 프로그램

디스플레이 9.4 string 클래스를 이용한 프로그램

```
1   //표준 string 클래스를 사용한 예제 프로그램.
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  }
```

빈 스트링의 값을 초기화시킴

스트링 변수 값을 초기화시키는 두 가지 방법

Sample Dialogue

```
I love fried ants!
Bon appetite!
```

String 클래스를 이용한 입/출력

- 일반 데이터 형과 같이 다루면 됨!

```
string s1, s2;
cout << “User types in: “ ;
cin >> s1;
cin >> s2;
cout << s1 << endl ;
cout << s2 << endl ;
```

  - 위의 프로그램 실행 결과:

```
User types in:
May the hair on your toes grow long and curly!
Mary
the
```

String 클래스를 이용한 getline() 함수 실행

- 다음과 같이 getline 함수를 사용하면:

```
string line;
cout << "Enter a line of input: ";
getline(cin, line);
cout << line << "[EOF]";
```

- 위의 프로그램을 다음과 같이 실행하면:

```
Enter a line of input: Do be do to you!
Do be do to you![EOF]
```

  - c-string에서 사용되는 getline() 함수와 유사함

getline() 함수의 다른 해석

- 입력 기능의 중지에 활용할 수 있음:

```
string line;
cout << "Enter input: ";
getline(cin, line, '?');
```

- 즉 "?" 문자를 만나면 입력 과정이 중단됨
  - getline() 함수는 reference를 리턴 할 수 있음
  - string s1, s2;

    getline(cin, s1) >> s2;
  - 실행 결과: (cin) >> s2;

함정: 입력 혼합

- cin >> 변수와 getline 함수의 혼용 사용에 유의함

```
int n;
string line;
cin >> n;
getline(cin, line);
```

  - 만약에 42

    Hello hitchhiker 가 입력되면
    - 변수 n은 42가 저장되고
    - line 값은 빈 string으로 설정됨!
  - cin >> n: 문자 앞에 오는 공백의 입력을 생략하지만, getline() 함수 수행을 위해서 "\n“ 문자는 남겨둠!

string 클래스 함수

- 함수들의 일부는 c-strings 기능과 유사함
- 또한!
  - 100개 이상의 표준 string class 함수들이 존재함
- 이 중에서:
  - .length()
    - string 변수의 길이
  - .at(i)
    - String 변수의 i번째 위치

string 표준 클래스의 멤버 함수들 (1 of 2)

디스플레이 9.7 string 표준 클래스의 멤버 함수

예제	주요 기능
생성자	
`string str;`	디폴트 생성자, str 객체 이름을 갖는 string 생성
`string str("string");`	"string" 값을 가지는 string 객체 생성
`string str(aString);`	aString 값을 복사한 str 객체 이름을 갖는 string 생성 aString은 string 클래스의 객체
원소 값 접근	
`str[i]`	i번째 인덱스에 위치한 문자 값을 입/출력
`str.at(i)`	i번째 인덱스에 위치한 문자 값을 입/출력
`str.substr(position, (length`	position 값에서 시작하여 length 값만큼의 길이를 가지는 문자열인 서브스트링 값을 리턴함
할당/변경자	
`str1 = str2;`	문자 저장공간을 설정하고 str2의 값으로 초기화시킴. str1을 위하여 공간을 설정하고 str1의 저장공간을 str2의 저장공간 크기로 설정함
`str1 += str2;`	str2 문자 값을 str1의 뒤에 연결시킴. size 값은 적절히 선택됨
`str.empty( )`	str이 빈 스트링인 경우 true 값을, 아니면 false 값을 리턴함

string 표준 클래스의 멤버 함수들 (2 of 2)

`str1 + str2`	str2 값이 str1의 끝 부분에 연결된 스트링을 리턴함. size 값은 적절히 선택됨
`str.insert(pos, str2)`	str2 값을 pos 위치부터 시작하는 str 값에 삽입함
`str.remove(pos, length)`	pos 위치부터 시작하여 length 크기만큼 서브스트링의 값을 삭제함
비교	
`str1 == str2 str1 != str2`	값이 서로 동일한지 여부를 확인하여 Boolean 값을 리턴함
`str1 < str2 str1 > str2` `str1 <= str2 str1 >= str2`	스트링 값의 크기를 비교하는 네 가지 방법. 값의 크기는 사전적 순서에 따라 결정함
`str.find(str1)`	str에서 str1 값이 처음으로 존재하는 인덱스 값을 리턴함
`str.find(str1, pos)`	str에서 str1 값이 처음으로 존재하는 인덱스 값을 리턴함. pos 위치부터 str1 값을 찾음
`str.find_first_of (str1, pos)`	str에서 str1에 존재하는 문자가 있는지 여부를 확인함. pos 위치부터 str1 문자의 존재 여부를 확인함
`str.find_first_not_of (str1, pos)`	str에서 str1에 존재하지 않는 문자가 존재하는 여부를 확인하며, pos 위치부터 시작함

C-strings와 string 객체의 상호 변환

- 자동적으로 형 변환이 이루어짐
  - c-string에서 string 객체로 변환 예:
    - 허용됨!

```
char aCString[] = "My C-string";
string stringVar;
stringVar = aCstring;
```

  - `aCString = stringVar;`
    - 허용되지 않음!
    - 자동적으로 c-string 변수로 변환되지 않음!
  - 다음과 같은 함수를 사용하여 변환시켜야 됨:

```
strcpy(aCString, stringVar.c_str());
```

요약

- C-strings 변수는 " 문자들의 조합으로 이루어진 배열"
  - 그리고 배열의 마지막 부분에 "\0“ 널 문자가 첨가됨
- C-strings 변수는 배열처럼 수행됨
  - 할당, 비교 연산자와 같이 사용될 수 없음
  - <cctype> 및 <string> 라이브러리에는 문자들을 조작하는데 유용한 함수들이 포함되어 있음
  - cin.get() 다음에 위치한 문자 1개를 입력함
  - getline() 다음 줄에 위치한 문자들을 입력함
  - string 객체는 c-strings 변수보다 사용자에게 편리성을 더 많이 제공함