# 11. character strings과 string functions

### Functions

#### gets(), puts(), strcat(), strcmp(), strcpy()

### string의 생성과 사용

### string functions, character functions

## 11.1 String 표현과 String I/O

### puts() – stdio.h

//Listing 11.1

// strings1.c

#include <stdio.h>

#define MSG "I am a symbolic string constant."

#define MAXLENGTH 81

int main(void)

{

char words[MAXLENGTH] = "I am a string in an array.";

const char\* pt1 = "Something is pointing at me.";

puts("Here are some strings:");//자동으로 newline을 append

puts(MSG);

puts(words);

puts(pt1);

words[8] = 'p';

puts(words);

getchar();

return 0;

}

## 11.1.1 program내에서 string 정의

### string constant를 사용한 char arrays, char pointer로 선언

## 11.1.1.1 string constants

### string literal, string constant

#### Character string constant는 *static storage* class에 저장된다

##### string constant는 program duration 동안에 오직 한번 생성되고 끝날때까지 지속된다. 해당 함수가 여러 번 호출되어도 오직 한번만 저장된다

##### quoted phrase “”은 string이 저장된 위치에 대한 pointer 역할을 한다

#### "are"표현은 address를 나타내고 %p을 사용하여 string의 첫번째 문자의 address를 출력할 수 있다

#### \*"space farers"는 address가 가리키는 위치의 값을 나타낸다

//Listing 11.2

/\* strptr.c -- strings as pointers \*/

#include <stdio.h>

int main(void)

{

printf("%s, %p, %c\n", "We", "are", \*"space farers");

getchar();

return 0;

}

## 11.1.1.2 character string array와 초기화

### character string array – compiler가 필요한 공간을 알고 메모리를 배정

const char ml[40] = “Limit”;

const char str[] = “new”;

const char \*ptr = “something”;

## 11.1.1.3 array 대비 pointer

### quoted string은 executable file의 data segment(데이터 영역의 초기화된 상수 값 영역)에 저장된다

#### C 프로그램: 코드 영역, 데이터 영역, stack 영역, heap 영역

##### 3가지 종류의 데이터 영역: const 초기화, global 변수, static 변수

###### 1) 읽기 전용으로 초기화되는 영역, .rodata가 대표적 영역,const로 선언된 변수, const string, printf의 format string,

###### 2) 읽기/쓰기 가능한 초기화 되는 영역, .data 영역,

###### 3) 초기화되지 않은 영역: BSS(Block started by symbol),초기화되지 않은 global 변수, static으로 선언된 영역

#### quoted string은 *static memory*에 있다고 말한다

##### 데이터 영역 > 읽기전용 초기화되는 영역-.rodata 영역

#### array을 위한 memory는 program이 실행된 후에 할당된다: stack 영역, local variable이므로

##### 데이터 영역의 quoted string이 stack 영역의 array로 복사된다 – 결국 같은 string이 2개 만들어지는 것이다

### const char ar1[] = "Something is pointing at me.";

#### printf("%s", \*"space"); //error 발생! 이유는?

\*주소인데 s는 string을 가져오라고 하므로

#### address ar1은 &ar1[0]과 같다

##### ar1은 address *constant이다*

##### ++ar1은 허용되지 않는다

#### array에서 다음 element를 가리키기 위해서 ar1+1을 사용할 수 있다

#### string literal은 const 데이터이다

**char \*name = "hong";**

**//\*name = "a";**

**char \*p;**

**p = "hong";//error 이유는?**

**Int a[2][3];**

**Int \*p=a;-int (\*)[]**

**Int \*p=&a[0];-int (\*)[]**

**Int \*p=&a[0][0];-\*\*a dereferencing**

**P는 1차원만 표현할 수 있는데 2,3은 2차원이므로 \*\*p라고 써야함**

#### pt가 constant 데이터를 가리키므로 const char \*pt로 선언하는 것이 맞다

//Listing 11.3

// addresses.c -- addresses of strings

//string을 가리키는 array, pointer, string 자체의 주소를 확인

#define MSG "I'm special."

#include <stdio.h>

int main()

{

char ar[] = MSG;

const char\* pt = MSG;

printf("address of \"I'm special\": %p \n", "I'm special");//MSG와는 .이 없는 차이

printf(" address ar: %p\n", ar);

printf(" address pt: %p\n", pt);

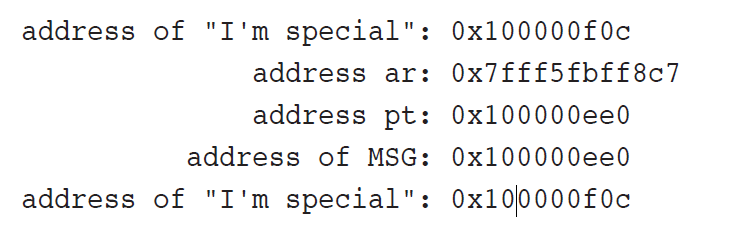
printf(" address of MSG: %p\n", MSG);

printf("address of \"I'm special\": %p \n", "I'm special");

getchar();

return 0;

}



#### pt와 MSG는 같은 address을 갖는다, ar의 주소는 다름

#### string literal "I'm special."이 main()에서 두개의 printf()문장에서 사용되었지마는, compiler는 two storage location에 저장하여 사용하고 있다. 실행하여 확인할 필요가 있음- “I’m special”로서 . 점이 없다는 차이가 있음

##### MSG의 address와 다르다

##### static data에 사용된 메모리 주소와 stack 영역의 dynamic memory, ar를 위해 사용된 메모리 주소와 다르다

#### string의 array 표현과 pointer representations 간이 차이는?

##### array name heart은 constant이나 pointer head는 variable이다

## 11.1.1.4 Array와 Pointer 차이

### string에 대한 char array, char pointer의 초기화

char heart[] = "I love Tillie!";

const char \*head = "I love Millie!";

#### array 표기법 사용

for (i = 0; i < 6; i++)

putchar(heart[i]); putchar('\n');

for (i = 0; i < 6; i++)

putchar(head[i])); putchar('\n');

#### pointer 표기법 사용

for (i = 0; i < 6; i++)

putchar(\*(heart + i)); putchar('\n');

for (i = 0; i < 6; i++)

putchar(\*(head + i)); putchar('\n');

#### pointer 사용, ++ 사용

while (\*(head) != '\0') /\* stop at end of string \*/

putchar(\*(head++)); /\* print character, advance pointer \*/

#### const modifier를 사용하지 않는 pointer 초기화

char \* word = "frame";

word[1] = 'l'; // allowed??

##### 메모리 접근 에러

const char \* pl = "Klingon"; // recommended usage

##### string literal에 pointer를 사용하지 않는 것이 좋다

## 11.2 Arrays of Character Strings

### char string을 2가지 표현법

### 두가지 approaches가 있다

#### array of pointers to strings

#### array of char arrays

//Listing 11.4

// arrchar.c -- array of pointers, array of strings

#include <stdio.h>

#define SLEN 40

#define LIM 5

int main(void)

{

const char\* mytalents[LIM] = {

"Adding numbers swiftly",

"Multiplying accurately", "Stashing data",

"Following instructions to the letter",

"Understanding the C language"

};

char yourtalents[LIM][SLEN] = {

"Walking in a straight line",

"Sleeping", "Watching television",

"Mailing letters", "Reading email"

};

int i;

puts("Let's compare talents.");

printf("%-36s %-25s\n", "My Talents", "Your Talents");

for (i = 0; i < LIM; i++)

printf("%-36s %-25s\n", mytalents[i], yourtalents[i]);

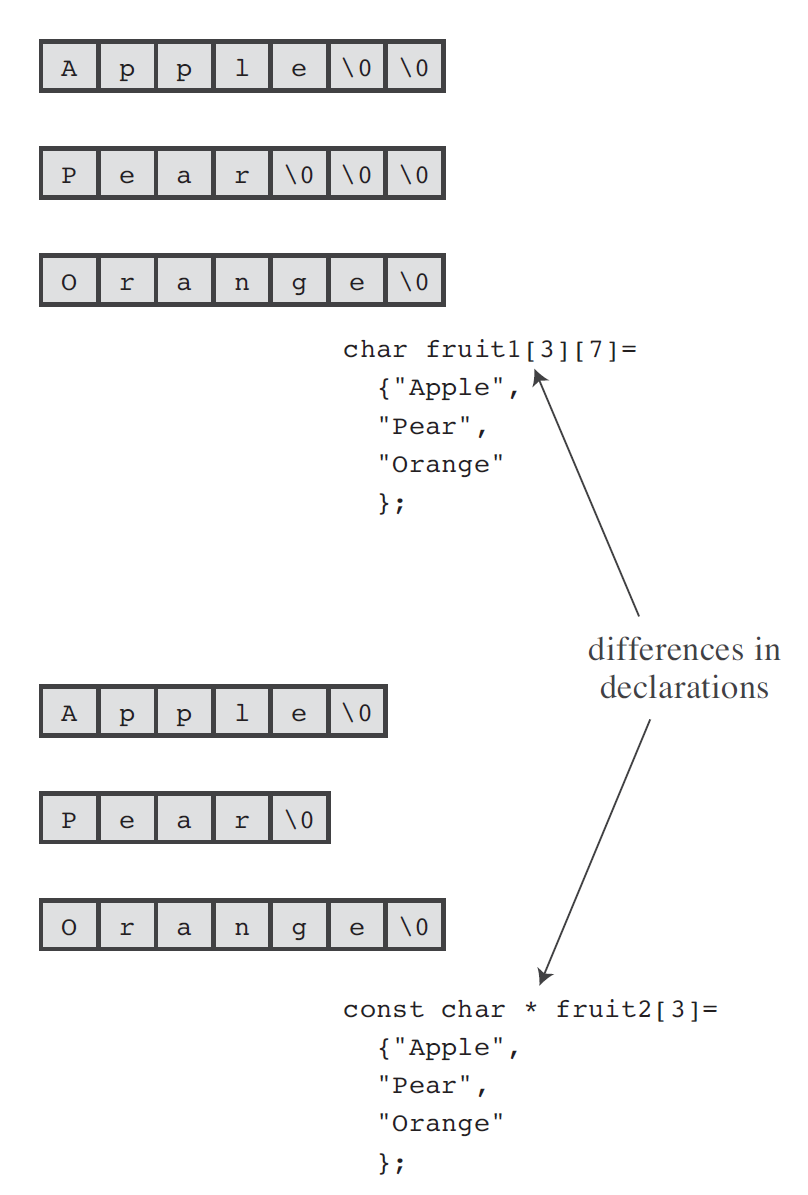
printf("\nsizeof mytalents: %zd, sizeof yourtalents: %zd\n",

sizeof(mytalents), sizeof(yourtalents));//5개 \* 8(pointer 길이) = 40, 200

getchar();

return 0;

}



#### char \* p[2] = {"hong", "bong"};//string은 static memory에, p[2] 변수는 stack 영역에, p[2]는 string 주소를 갖고 있다

##### char \*p[2];

##### p[0] = "Hong"; //error! 이유는? Error 안 남 개꿀잼몰카

## 11.2.1 pointers와 strings

### string에 대한 operation은 pointer로 처리한다

//Listing 11.5

/\* p\_and\_s.c -- pointers and strings \*/

#include <stdio.h>

int main(void)

{

const char\* mesg = "Don't be a fool!";

const char\* copy;

copy = mesg;//copy of address

printf("%s\n", copy);

printf("mesg = %s; &mesg = %p; value = %p\n",

mesg, &mesg, mesg);

printf("copy = %s; &copy = %p; value = %p\n",

copy, &copy, copy);//copy == mesg, &mesg != &copy

getchar();

return 0;

}

#### copy = mesg; //copy of pointer 값만

#### strcpy() 사용: string 자체의 copy

## 11.3 Character Strings 입력

### program으로 string의 read

#### string을 저장한 space를 set aside 해야 한다

## 11.3.1 string을 저장할 공간 만들기

char \*name;

scanf("%s", name);

### scanf()는 입력된 string을 argument로 주어진 address에 복사하는 기능을 수행한다

#### argument는 uninitialized pointer이기 때문에 오류이다

##### name might point anywhere

#### char name[];으로 사용해야 함

##### scanf(), gets(), fgets()

## 11.3.2 gets()-visual studio에서 틀어막음

### read an entire line of input at a time

#### newline까지 전체 line을 읽다, newline 버리고, c string을 저장

//Listing 11.6

/\* getsputs.c -- using gets() and puts() \*/

#include <stdio.h>

#define STLEN 81

int main(void)

{

char words[STLEN];

puts("Enter a string, please.");

//gets(words); // read an entire line of input instead of a single word until newline

//gets() not check if the input line fits into the array

gets\_s(words);

printf("Your string twice:\n");

printf("%s\n", words);

puts(words);

puts("Done.");

getchar();

return 0;

}

##### gets()는 input line이 array 에 잘 fit되는지를 check하지 않는다

###### gets()는 array가 시작하는 주소는 알아도 얼마나 많은 문자가 있는지를 알지 못한다

##### input이 길면 buffer overfow

###### 초과된 문자들은 unused memory를 사용하게 되고 다른 영역에 overwrite하게 된다

###### segmentation fault

#### C99, C11: gets() 비추천 또는 제외

### gets()의 대체

#### fgets(): read할 문자 숫자를 2nd argument로 지정

#### 3rd argument: indicate which file to read

//Listing 11.7

/\* fgets1.c -- using fgets() and fputs() \*/

#include <stdio.h>

#define STLEN 14

int main(void)

{

char words[STLEN];

puts("Enter a string, please.");

fgets(words, STLEN, stdin);

printf("Your string twice (puts(), then fputs()):\n");

puts(words);

fputs(words, stdout);

puts("Enter another string, please.");

fgets(words, STLEN, stdin);

printf("Your string twice (puts(), then fputs()):\n");

puts(words);

fputs(words, stdout);

puts("Done.");

getchar();

return 0;

}

##### return a pointer to char

//Listing 11.8

/\* fgets2.c -- using fgets() and fputs() \*/

#include <stdio.h>

#define STLEN 10

int main(void)

{

char words[STLEN];

puts("Enter strings (empty line to quit):");

while (fgets(words, STLEN, stdin) != NULL && words[0] != '\n')

fputs(words, stdout);

puts("Done.");

return 0;

}

#### fgets()은 newline은 string에 포함하고 있음

##### newline의 제거

while (words[i] != '\n') // assuming \n in words

i++;

words[i] = '\0';

//Listing 11.9

/\* fgets3.c -- using fgets() \*/

#include <stdio.h>

#define STLEN 10

int main(void)

{

char words[STLEN];

int i;

puts("Enter strings (empty line to quit):");

while (fgets(words, STLEN, stdin) != NULL

&& words[0] != '\n')

{

i = 0;

while (words[i] != '\n' && words[i] != '\0')

i++;

if (words[i] == '\n')

words[i] = '\0';

else // must have words[i] == '\0'

while (getchar() != '\n')

continue;

puts(words);

}

puts("done");

return 0;

}

Puts=printf, fget=get

#### gets\_s(words, STLEN) C11

## 11.3.3 scanf()

### gets(): newline을 입력될때까지 모든 문자를 입력받음

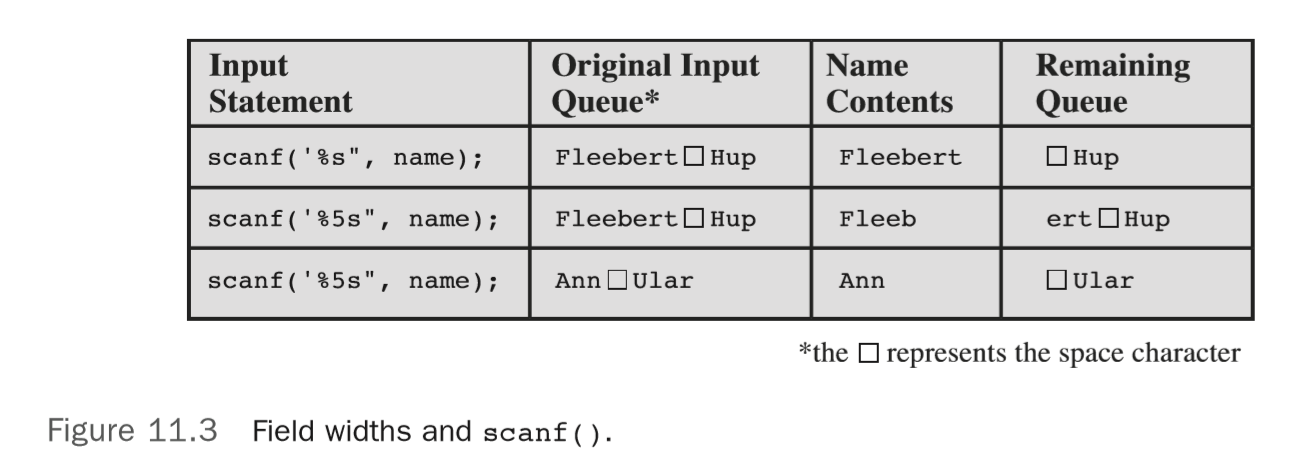
### scanf(): 입력되는 string을 whitespace(blank, tab, newline)로 구분

#### read a single word

#### %s: string의 시작은 non-whitespace, 끝은 whitespace

#### %10s: 10개 문자 또는 whitespace 이전까지 입력받음

### gets(): read the entire line



//Listing 11.11

/\* scan\_str.c -- using scanf() \*/

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

char name1[11], name2[11];

int count;

printf("Please enter 2 names.\n");

count = scanf("%5s %10s", name1, name2);

printf("I read the %d names %s and %s.\n",

count, name1, name2);

getchar();

/\*

int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};

printf("a 주소 = %p\n", a);

//method1

int\*\* p;

//p = &a[0][0];

//\*p = a[0];

//method2

int\* pt = &a[0][0];

p = &pt;

\*p = a[0];

for (int i = 0; i < 12; i++)

{

printf("p = %p\n",\*p);

(\*p)++;

}

char b[3];

char\* ptr;

ptr = b;

scanf("%s", ptr);

\*/

char c[4][4] = {};

char\*\* ptr2;

char\* pt2= &c[0][0];

ptr2 = &pt2;

\*ptr2 = c[0];

for(int i=0;i<4;i++)

{

scanf("%s", \*ptr2);

printf("%s\n", \*ptr2);

(\*ptr2)++;

}

\*ptr2 = c[0];

for (int i = 0; i < 3; i++)

{

printf("%s\n", \*(ptr2)+i);

}

system("pause");

return 0;

}

## 11.3 String Output

### puts(), fputs(), printf()

## 11.3.1 puts()

### argument가 string의 address

//Listing 11.12

/\* put\_out.c -- using puts() \*/

#include <stdio.h>

#define DEF "I am a #defined string."

int main(void)

{

char str1[80] = "An array was initialized to me.";

const char\* str2 = "A pointer was initialized to me.";

puts("I'm an argument to puts().");

puts(DEF);

puts(str1);

puts(str2);

puts(&str1[5]);

puts(str2 + 4);

return 0;

}

##### puts()는 null character가 있어야 stop

##### null이 없는 경우

//Listing 11.13

/\* nono.c -- no! \*/

#include <stdio.h>

int main(void)

{

char side\_a[] = "Side A";

char dont[] = { 'W', 'O', 'W', '!' };

char side\_b[] = "Side B";

puts(dont); /\* dont is not a string \*/

return 0;

}

## 11.3.2 fputs()

### 2nd argument: write할 file을 표시

##### newline을 output에 자동으로 append하지 않음

## 11.3.3 printf()

### character strings and formatted input/output

#### printf(“%s\n”, string) 은 puts(string)와 같은 효과

## 11.4 String functions

### string.h

#### strlen() , strcat() , strncat(), strcmp(), strncmp(), strcpy(), strncpy()

## 11.4.1 strcmp()

/\* listing 11.21 compare.c -- this will work \*/

#include <stdio.h>

#include <string.h> // declares strcmp()

#define ANSWER "Grant"

#define SIZE 40

char\* s\_gets(char\* st, int n);

int main(void)

{

char try[SIZE];

puts("Who is buried in Grant's tomb?");

s\_gets(try, SIZE);

while (strcmp(try, ANSWER) != 0)

{

puts("No, that's wrong. Try again.");

s\_gets(try, SIZE);

}

puts("That's right!");

return 0;

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

int i = 0;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

while (st[i] != '\n' && st[i] != '\0')

i++;

if (st[i] == '\n')

st[i] = '\0';

else // must have words[i] == '\0'

while (getchar() != '\n')

continue;

}

return ret\_val;

}

/\* listing 11.23 quit\_chk.c -- beginning of some program \*/

#include <stdio.h>

#include <string.h>

#define SIZE 80

#define LIM 10

#define STOP "quit"

char\* s\_gets(char\* st, int n);

int main(void)

{

char input[LIM][SIZE];

int ct = 0;

printf("Enter up to %d lines (type quit to quit):\n", LIM);

while (ct < LIM && s\_gets(input[ct], SIZE) != NULL &&

strcmp(input[ct], STOP) != 0)

{

ct++;

}

printf("%d strings entered\n", ct);

return 0;

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

int i = 0;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

while (st[i] != '\n' && st[i] != '\0')

i++;

if (st[i] == '\n')

st[i] = '\0';

else // must have words[i] == '\0'

while (getchar() != '\n')

continue;

}

return ret\_val;

}

## 11.4.2 strncmp()

/\* listing 11.24 starsrch.c -- use strncmp() \*/

#include <stdio.h>

#include <string.h>

#define LISTSIZE 6

int main()

{

const char\* list[LISTSIZE] =

{

"astronomy", "astounding",

"astrophysics", "ostracize",

"asterism", "astrophobia"

};

int count = 0;

int i;

for (i = 0; i < LISTSIZE; i++)

if (strncmp(list[i], "astro", 5) == 0)

{

printf("Found: %s\n", list[i]);

count++;

}

printf("The list contained %d words beginning"

" with astro.\n", count);

return 0;

}

## 11.4.3 strcpy() strncpy()

/\* listing11.25 copy1.c -- strcpy() demo \*/

#include <stdio.h>

#include <string.h> // declares strcpy()

#define SIZE 40

#define LIM 5

char\* s\_gets(char\* st, int n);

int main(void)

{

char qwords[LIM][SIZE];

char temp[SIZE];

int i = 0;

printf("Enter %d words beginning with q:\n", LIM);

while (i < LIM && s\_gets(temp, SIZE))

{

if (temp[0] != 'q')

printf("%s doesn't begin with q!\n", temp);

else

{

strcpy(qwords[i], temp);

i++;

}

}

puts("Here are the words accepted:");

for (i = 0; i < LIM; i++)

puts(qwords[i]);

return 0;

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

int i = 0;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

while (st[i] != '\n' && st[i] != '\0')

i++;

if (st[i] == '\n')

st[i] = '\0';

else // must have words[i] == '\0'

while (getchar() != '\n')

continue;

}

return ret\_val;

}

/\* listing11.26 copy2.c -- strcpy() demo \*/

#include <stdio.h>

#include <string.h> // declares strcpy()

#define WORDS "beast"

#define SIZE 40

int main(void)

{

const char\* orig = WORDS;

char copy[SIZE] = "Be the best that you can be.";

char\* ps;

puts(orig);

puts(copy);

ps = strcpy(copy + 7, orig);

puts(copy);

puts(ps);

return 0;

}

/\* listing 11.27 copy3.c -- strncpy() demo \*/

#include <stdio.h>

#include <string.h> /\* declares strncpy() \*/

#define SIZE 40

#define TARGSIZE 7

#define LIM 5

char\* s\_gets(char\* st, int n);

int main(void)

{

char qwords[LIM][TARGSIZE];

char temp[SIZE];

int i = 0;

printf("Enter %d words beginning with q:\n", LIM);

while (i < LIM && s\_gets(temp, SIZE))

{

if (temp[0] != 'q')

printf("%s doesn't begin with q!\n", temp);

else

{

strncpy(qwords[i], temp, TARGSIZE - 1);

qwords[i][TARGSIZE - 1] = '\0';

i++;

}

}

puts("Here are the words accepted:");

for (i = 0; i < LIM; i++)

puts(qwords[i]);

return 0;

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

int i = 0;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

while (st[i] != '\n' && st[i] != '\0')

i++;

if (st[i] == '\n')

st[i] = '\0';

else // must have words[i] == '\0'

while (getchar() != '\n')

continue;

}

return ret\_val;

}

## 11.5 sorting strings 연습해라 이샛ㄱ야

/\* listing11.29 sort\_str.c -- reads in strings and sorts them \*/

#include <stdio.h>

#include <string.h>

#define SIZE 81 /\* string length limit, including \0 \*/

#define LIM 20 /\* maximum number of lines to be read \*/

#define HALT "" /\* null string to stop input \*/

void stsrt(char\* strings[], int num);/\* string-sort function \*/

char\* s\_gets(char\* st, int n);

int main(void)

{

char input[LIM][SIZE]; /\* array to store input \*/

char\* ptstr[LIM]; /\* array of pointer variables \*/

int ct = 0; /\* input count \*/

int k; /\* output count \*/

printf("Input up to %d lines, and I will sort them.\n", LIM);

printf("To stop, press the Enter key at a line's start.\n");

while (ct < LIM && s\_gets(input[ct], SIZE) != NULL

&& input[ct][0] != '\0')

{

ptstr[ct] = input[ct]; /\* set ptrs to strings \*/

ct++;

}

stsrt(ptstr, ct); /\* string sorter \*/

puts("\nHere's the sorted list:\n");

for (k = 0; k < ct; k++)

puts(ptstr[k]); /\* sorted pointers \*/

return 0;

}

/\* string-pointer-sorting function \*/

void stsrt(char\* strings[], int num)

{

char\* temp;

int top, seek;

for (top = 0; top < num - 1; top++)

for (seek = top + 1; seek < num; seek++)

if (strcmp(strings[top], strings[seek]) > 0)

{

temp = strings[top];

strings[top] = strings[seek];

strings[seek] = temp;

}

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

int i = 0;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

while (st[i] != '\n' && st[i] != '\0')

i++;

if (st[i] == '\n')

st[i] = '\0';

else // must have words[i] == '\0'

while (getchar() != '\n')

continue;

}

return ret\_val;

}

