챕터 21 2019202050 이강현

실습 1.

#include <stdio.h> // 헤더파일 선언

int main(void)

{

int ch1, ch2; // int형 변수 ch1, ch2 선언

ch1 = getchar(); // ch1에 문자 입력

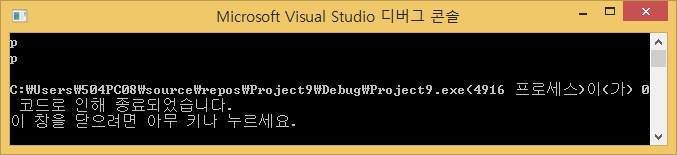
ch2 = fgtc(stdin); // 엔터키 입력

putchar(ch1); // 문자 출력

fputc(ch2, stdout); // 엔터키 출력

return 0;

}



한줄 토의: 입출력 스트림에 대한 개념과 입출력 함수의 종류를 알게 됨.

실습2.

#include <stdio.h> // 헤더파일 선언

int main(void)

{

int ch; // int형 변수 ch 선언

while (1) { // 반복문 무한 반복

ch = getchar(); // ch에 값 문자 저장

if (ch == EOF)

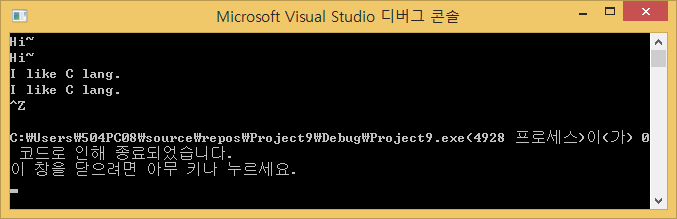
break; // EOF값을 받을 때 무한루프 종료

putchar(ch); // ch에 저장된 문자 출력

}

return 0; // 종료

}



한줄 토의: EOF는 -1로 정의된 상수이라는 것과 콘솔 대상의 fgetc, getchar 함수호출로 EOF를 반환하기 위해 Ctrl+Z를 이용함을 알게 됨.

실습 3.

#include <stdio.h> // 헤더파일 선언

int main(void)

{

char\* str = "Simple String"; // char형 포인터 변수 str선언 후 초기화

printf("1. puts test ----- \n");

puts(str); // str 출력

puts("So Simple String");// "So Simple String" 출력

printf("2. fputs test ----- \n"); // 단순 출력문

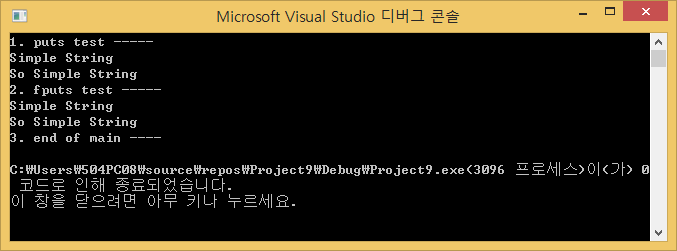
fputs(str, stdout); printf("\n"); // str 출력 후 줄내림

fputs("So Simple String", stdout); printf("\n"); //"So Simple String" 출력 후 줄내림

printf("3. end of main ---- \n"); // 단순 출력문

return 0;

}



한줄 토의: puts 함수는 출력 후 자동으로 개행하고 별도의 모니터 출력 선언이 없어도 출력된다. 그러나 fputs 함수는 출력 후 자동개행이 없으며 모니터 출력을 선언하여 이용가능하다.

실습4.

#include <stdio.h>

int main(void)

{

char str[7]; // 크기가 char형 배열 str 선언

int i; // int형 변수 i 선언

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

{

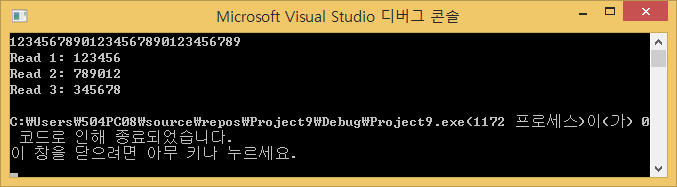
fgets(str, sizeof(str), stdin); // stdin으로부터 문자열 입력 받아 str에 저장

printf("Read %d: %s\n", i + 1, str); // str출력

}

return 0; // 종료

}



한줄 토의: 문자열의 끝에 자동으로 널 문자가 추가된다. fgets은 stdin에서 문자열을 입력받아 저장할 수 있다. 엔터도 문자열로 인식한다.

실습5.

#include <stdio.h>

int main(void)

{

char perID[7]; // 크기가 7인 char형 배열

char name[10];//크기가 10인 char형 배열

fputs("주민번호 앞 6자리 입력: ", stdout);//모니터 출력

fgets(perID, sizeof(perID), stdin);//id입력 받기

fputs(“이름 입력: ", stdout);//모니터 출력

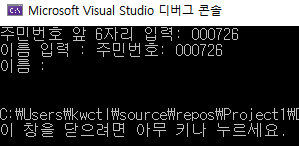
fgets(name, sizeof(name), stdin);//id입력받기

printf("주민번호: %s \n",perID);

printf("이름 : %s \n", name);

return 0; // 종료

}



한줄 토의: 실행창을 보면 입력이 제대로 이루어지지 않았다는 것을 확인할 수 있는데 이는 엔터키도 입력처리하기 때문에 발생하는 문제이다. 따라서 엔터가 저장된 버퍼를 비워야 한다.

실습6.

#include <stdio.h> //헤더파일 선언

void ClearLineFromReadBuffer(void) // 입력버퍼 비우는 함수

while (getchar() != ’\n’); // getchar() 함수로 버퍼에 남아있는 것들을 문자단위로 읽어드리는 작업을 ‘\n’이 나올 때 까지 반복

}

int main(void) {

char perID[7]; // 크키가 7인 char형 1차원 배열 perID 선언

char name[10]; // 크기가 10인 char형 1차원 배열 name 선언

fputs("주민번호 앞 6자리 입력: ", stdout); "주민번호 앞 6자리 입력: "를 화면에 출력

fgets(perID, sizeof(perID), stdin); // stdin으로부터 7만큼의 perID를 읽어들임

ClearLineFromReadBuffer(); // 입력버퍼 비우기

fputs("이름 입력: ", stdout); // "이름 입력: " 화면 출력

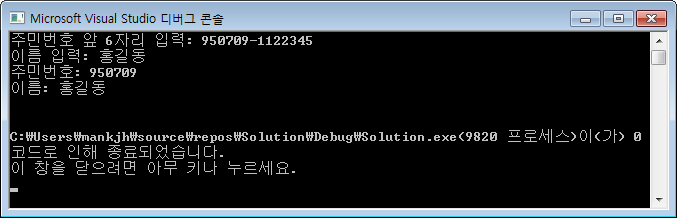
fgets(name, sizeof(name), stdin); // 10만큼의 name을 stdin으로부터 입력받음

printf("주민번호: %s \n", perID); // perID 출력

printf("이름: %s\n", name); // name 출력

return 0; // 종료

}



한줄 토의: 주민번호 6자리만을 입력받을 수 있도록 입력버퍼을 비우는 방법을 알게 됨.

실습7.

#include <stdio.h>

void RemoveBSN(char str[]) // 문자열 마지막의 널문자 제거 함수

{

int len = strlen(str); // int형 변수 lenㅇ르 설정한 후 str의 크기로 초기화

str[len - 1] = 0; // str[len-1]에 있는 널문자를 제거

}

int main(void) // 전달인자가 없는 int형 main 함수

{

char str[100]; // 크기가 100인 char형 일차원 배열 str 선언

printf("문자열 입력: "); // "문자열 입력: " 출력

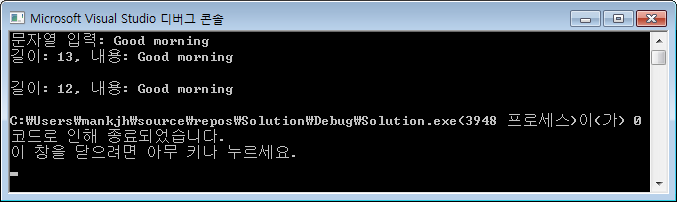
fgets(str, sizeof(str), stdin); // str에 stdin으로 부터 크기가 100 까지인 문자를 저장

printf("길이: %d, 내용: %s \n", strlen(str), str); // str의 길이와 값 출력

RemoveBSN(str); // str에 저장된 문자열의 마지막에 있는 널 문자 제거

printf("길이: %d, 내용: %s \n", strlen(str), str); // str길이와 str 값 출력

return 0; // 종료

}

한줄 토의:널 문자의 유무가 문자열의 끝을 파악할 수 있는 중요한 도구이다. 실습8

#include <stdio.h>

#include <string.h> // 헤더파일 선언

int main(void) // 전달인자가 없는 int형 main 함수

{

char str1[20] = "1234567890"; // 크기가 20인 char형 일차원배열 str1선언 후 초기화

char str2[20]; // 크기가 20인 char형 일차원배열 str2선언

char str3[5]; // 크기가 5인 char형 일차원배열 str3선언

/\*\*\*\* case 1 \*\*\*\*/

strcpy(str2, str1); // str2에 str1에 저장된 문자열 복사

puts(str2); // str2 출력

/\*\*\*\* case 2 \*\*\*\*/

strncpy(str3, str1, sizeof(str3)); //str1에 저장된 문자열을 최대 sizeof(str3)값 만큼 str3에 복사

puts(str3);

/\*\*\*\* case 3 \*\*\*\*/

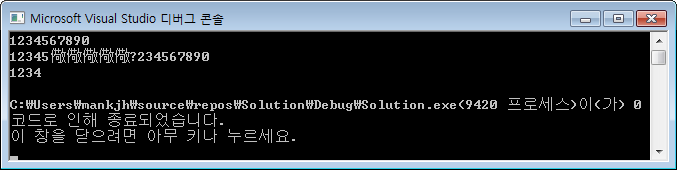
strncpy(str3, str1, sizeof(str3) - 1); //str1에 저장된 문자열을 최대 sizeof(str3)-1값 만큼만 str3에 복사

str3[sizeof(str3) - 1] = 0; // 널문자 삽입

puts(str3); // str3 cnffur

return 0; // 종료

}



한줄 토의: 문자열을 단순히 저장할 때 뿐만 아니라 값을 넘길 때도 마지막의 널문자를 고려해야한다는 점을 알게됨.실습9

#include <stdio.h>

#include <string.h> int main(void) // 전달인자가 없는 int형 main 함수

{

char str1[20] = "First~"; // 크기가 20인 char형 일차원배열 str1선언 후 초기화

char str2[20] = "Second"; // 크기가 20인 char형 일차원배열 str2선언 후 초기화

char str3[20] = "Simple num: "; // 크기가 20인 char형 일차원배열 str3선언 후 초기화

char str4[20] = "1234567890"; // 크기가 20인 char형 일차원배열 str4선언 후 초기화

/\*\*\*\* case 1 \*\*\*\*/

strcat(str1, str2); // str1에 str2를 덧붙임

puts(str1); // str1 출력

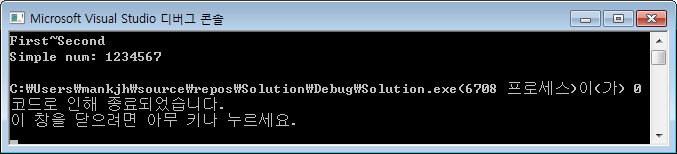
/\*\*\*\* case 2 \*\*\*\*/

strncat(str3, str4, 7); // str3에 str4를 최대 8만큼 덧붙임

puts(str3); // str2 출력

return 0; // 종료

}



한줄 토의: string헤더파일의 함수를 알아둬야 할 것 같다.

실습10

#include <stdio.h>

#include <string.h> // 헤더파일 선언

int main(void) // 전달인자가 없는 int형 main 함수

{

char str1[20]; // 크기가 20인 char형 일차원배열 str1선언 후 초기화

char str2[20]; // 크기가 20인 char형 일차원배열 str2선언 후 초기화

printf("문자열 입력 1: "); // "문자열 입력 1: " 출력

scanf("%s", str1); // str1 입력

printf("문자열 입력 2: "); // "문자열 입력 2: " 출력

scanf("%s", str2); // str2 입력

//fputs(strcmp(str1, str2), stdin);

if (!strcmp(str1, str2)) { // str1과 str2가 동일할 경우 if문은 참이 되고 다를경우 거짓이 됨

puts("두 문자열은 완벽히 동일합니다."); // 참일 경우 "두 문자열은 완벽히 동일합니다." 출력

}

else { // 그렇지않을 경우 경우

puts("두 문자열은 동일하지 않습니다."); // "두 문자열은 동일하지 않습니다." 출력

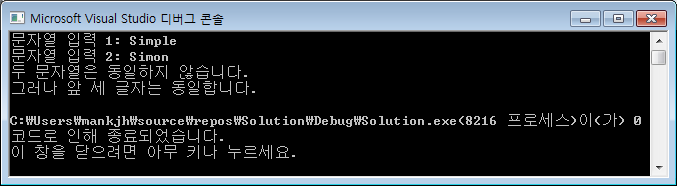
if (!strncmp(str1, str2, 3)) // str1과 str2의 앞에서 3칸만 비교

puts("그러나 앞 세 글자는 동일합니다."); // "그러나 앞 세 글자는 동일합니다." 출력

}

return 0; // 종료

}



한줄 토의: Strcmp와 strncmp 함수를 통해 문자열 간의 비교가 가능함을 알게됨.

실습 11

#include <stdio.h>

#include <stdlib.h> // 헤더파일 선언

int main(void) // 전달인자가 없는 int형 main 함수

{

char str[20]; // 크기가 20인 char형 일차원배열 str선언 후 초기화

printf("정수 입력: "); // "정수 입력: " 출력

scanf("%s", str); // str 입력

printf("%d \n", atoi(str)); // atoi(str) 출력 (int형으로 변환)

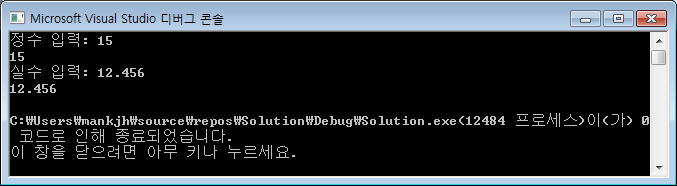
printf("실수 입력: "); // "실수 입력: " 출력

scanf("%s", str); // str 입력

printf("%g \n", atof(str)); // atof(str) 출력(double형으로 변환)

return 0; // 종료

}



한줄 토의:문자열로 입력을 받았을 때 값의 자료형을 변환할 수 있게 됨.