2020 학년도 1 학기
C 애플리케이션구현
중간대체 포트폴리오

학과	컴퓨터정보공학과
학번	20191768
성명	신윤규

동양미래대학교
DONGYANG MIRAE UNIVERSITY

목차

목차

C 언어란?

C 언어는 1972 년 데니스 리치가 개발한 프로그래밍 언어이며 운영체제인 유닉스를 개발하기 위해 C 언어를 개발한 것이며 켄 톰슨이 1970 년 개발한 B 언어에서 유래된 프로그래밍 언어입니다.

기타 프로그래밍의 발전 과정은 밑에 그림과 같습니다.

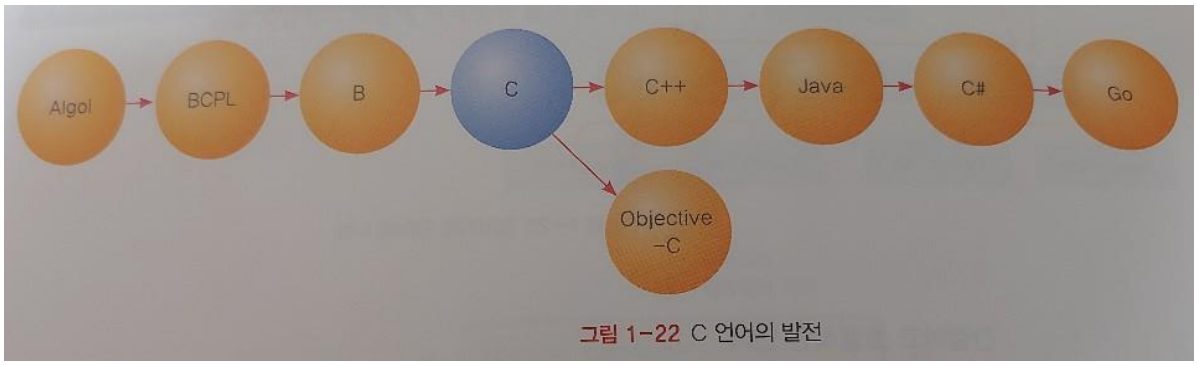


그림 1-22 C 언어의 발전

다음으로 C 언어 특징은

첫번째 절차지향적 언어입니다.

두번째는 간결하고 효율적인 언어입니다.

세번째는 이식성이 좋은 프로그래밍 언어입니다.

반면 단점은 배우기 어렵다는 단점이 있지만 한번 배우고 나면 기타 프로그래밍 언어 습득에 많은 도움을 준다고 합니다.

C 언어의 사용분야는 아두이노와 같은 임베디드 시스템에 C 를 많이 사용합니다.

또한 보안 쪽에서도 많은 사용을 한다고 합니다.

CHAPTER 11(문자와 문자열)

우선 JAVA 같은 경우 문자는 char 를 사용하고 문자열은 String 을 사용합니다. 반면

C 언어의 문자와 문자열들은 기타 언어와 달리 문자는 JAVA 와 같이 char 를 사용하지만 문자열에서는 char 에 배열을 선언해 문자열을 사용한다는 특징이 있습니다.

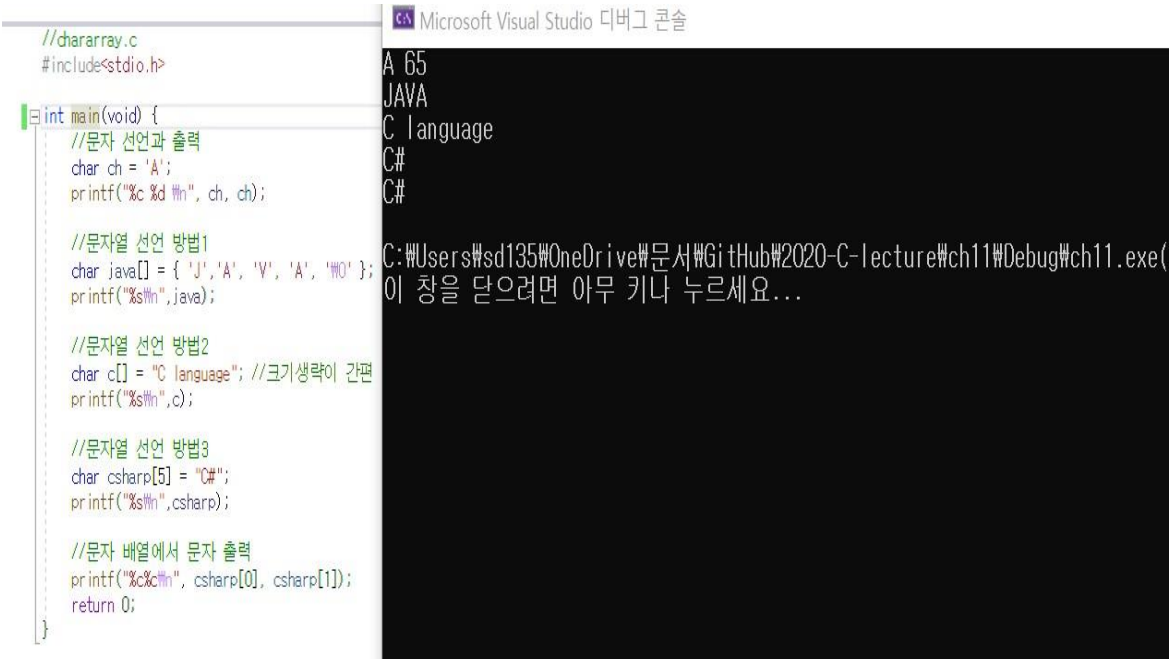
C 에서 문자열은 항상 널 문자(NULL)가 붙습니다. NULL 은 0 으로 표현이 가능하고 문자열의 끝을 나타내기 때문에 printf 는 문자열을 출력할 때 NULL 에서 출력을 끝냅니다.

마지막으로 문자열 포인터에 대해 알아보도록 하겠습니다.

Char 포인터에 “”(큰따옴표)를 사용하면 문자열 포인터라고 할 수 있습니다.

출력을 할 때 서식 지정자 %s 로 사용하면 문자열로 출력이 됩니다.

실습 예제 11-1



```c
//chararray.c
#include<stdio.h>

int main(void) {
    //문자 선언과 출력
    char ch = 'A';
    printf("%c %d \n", ch, ch);

    //문자열 선언 방법1
    char java[] = { 'J','A', 'V', 'A', '\0' };
    printf("%s\n",java);

    //문자열 선언 방법2
    char c[] = "C language"; //크기생략이 간편
    printf("%s\n",c);

    //문자열 선언 방법3
    char csharp[5] = "C#";
    printf("%s\n",csharp);

    //문자 배열에서 문자 출력
    printf("%c%c\n", csharp[0], csharp[1]);
    return 0;
}
```

```
Microsoft Visual Studio 디버그 콘솔
A 65
JAVA
C language
C#
C#

C:\Users\sd135\OneDrive\문서\GitHub\2020-C-lecture\ch11\Debug\ch11.exe(
이 창을 닫으려면 아무 키나 누르세요...
```

문자 변수 ch 에 A 를 선언하고 문자열 변수 java 와 C 에는 각 칸마다 문자를 저장을 하며

[]안에 범위를 설정해 주어 다른 데이터 값을 낭비하지 않게 해주었습니다.

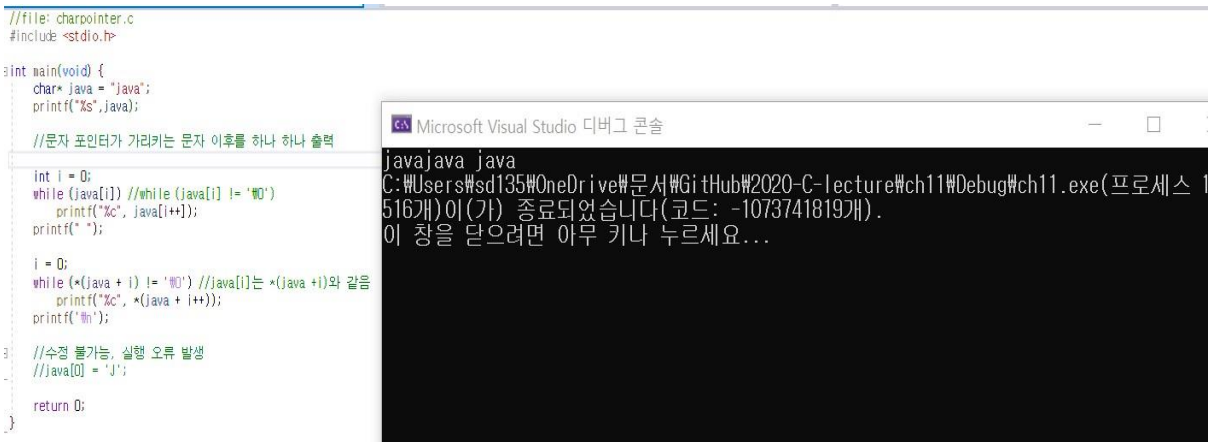
변수 java 와 C 에서는 선언을 해주고 크기가 결정이 되는 형식입니다. 반면 char csharp[5] 변수는 범위를 설정해주고 그 안에 4 개의 문자가 넘지 않게 작성을 해주었습니다.

실습 예제 11-2

밑에 그림을 보시면 문자 포인터를 이용해 java 라는 값을 불러 while 문안에 java 배열에 변수 i 를 초기화 해줘 while(java[i])값을 넣어 코드를 설계한 것을 보실 수가 있습니다.

하지만 제가 주석처리를 하였지만 java[0] = 'J'; 사용하면 오류가 납니다. 오류 나는 이유는

포인터 변수를 통해 간접 참조는 가능하지만 직접 참조는 못하기 때문입니다.

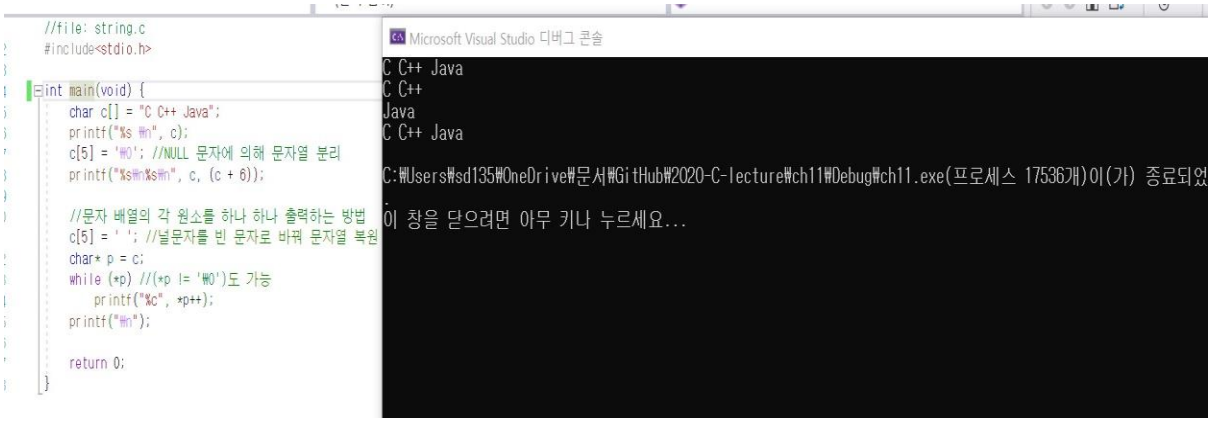


실습 예제 11-3

문자열 c 에 값을 저장하고 문자열 5 번째에 널 값을 주어 문자열을 분리하였습니다.

다시 문자열 5 번째에 빈 문자로 바꾸어 문자열을 복원시켜 연결을 해주었습니다.

포인터 p 를 c 의 주소 값을 가리켜 while 문을 사용해 저장된 칸을 불러주었습니다.



실습 예제 11-4

#include<conio.h>는 getchar() , getche(), getch()를 사용할 수 있게 하는 라이브러리 함수 입니다. 우선 각 함수의 기능을 살펴보도록 하겠습니다.

함수 getchar()

엔터키를 눌러야 이전에 입력한 문자의 입력을 보여주는 함수입니다.

함수 getche()

한 문자를 입력하면 바로 옆에 같은 문자가 출력되는 기능입니다.

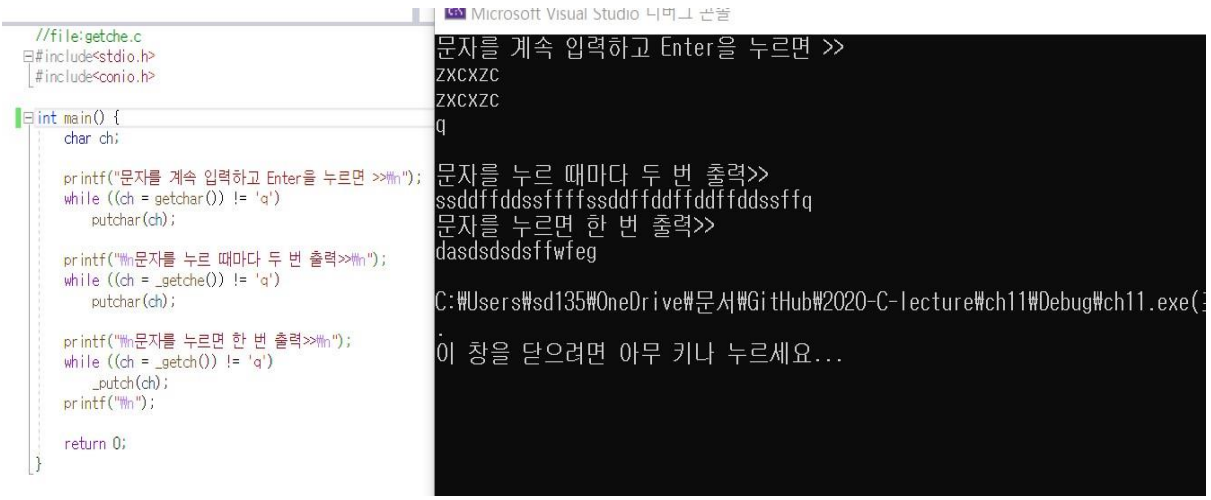
함수 getch()

타 함수와 다르게 putch(ch)를 이용해 출력을 할 수 있게 합니다.

각 함수들 별로 while 문을 써주고 q 를 입력하면 나가게 해줍니다.

Putchar 로 출력을 하게 만들고 특이점이 있으면 getche()와 getch() 같은 함수는 엔터키를 누르고 입력을 하게 됩니다.

다시 입력했던 라인의 글자가 바뀌게 되는 것을 확인할 수가 있습니다.



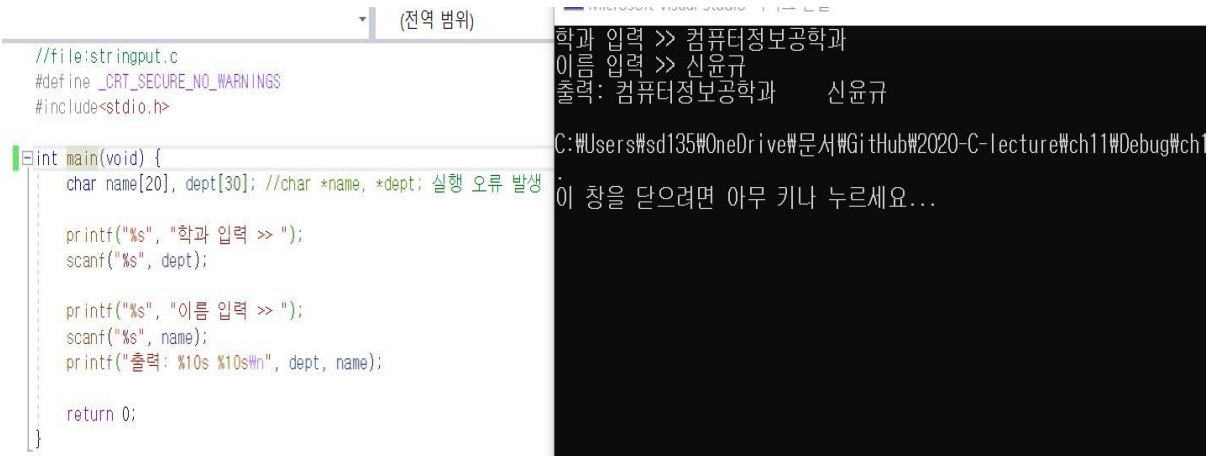
실습 예제 11-5

콘솔창에 문자를 입력하고 출력하게 하기 위해 scanf 를 사용합니다.

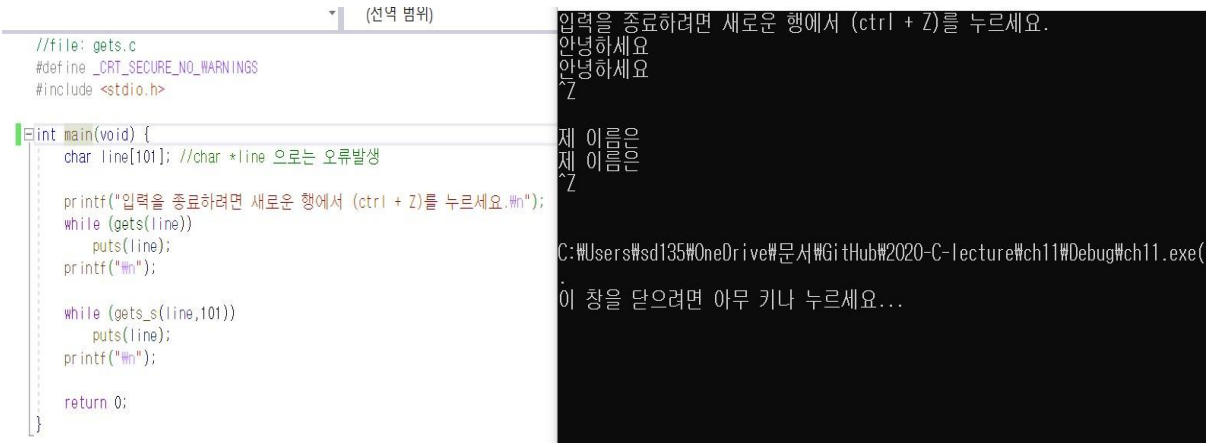
#define _CRT_SECURE_NO_WARNINGS 를 입력해 scanf 의 오류를 막습니다.

문자열로 scanf("%s",str) 공식을 문자열로 사용을 합니다. 문자열과 달리

정수형에서 scanf 를 이용하면 scanf("%d", &i); 이용하여 차이가 있음을 알 수가 있습니다. 또한 안전하게 사용하려면 scanf_s 를 이용하면 안전하게 사용을 할 수가 있습니다.



실습 예제 11-6



```
//file: gets.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void) {
    char line[101]; //char *line 으로는 오류발생

    printf("입력을 종료하려면 새로운 행에서 (ctrl + Z)를 누르세요.₩n");
    while (gets(line))
        puts(line);
    printf("₩n");

    while (gets_s(line,101))
        puts(line);
    printf("₩n");

    return 0;
}
```

```
입력을 종료하려면 새로운 행에서 (ctrl + Z)를 누르세요.
안녕하세요
안녕하세요
^Z

제 이름은
제 이름은
^Z


C:₩Users₩sd135₩OneDrive₩문서₩GitHub₩2020-C-lecture₩ch11₩Debug₩ch11.exe(
.
이 창을 닫으려면 아무 키나 누르세요...
```

위 예제에서 gets()와 puts()에 대해 알아보도록 하겠습니다.

gets()/gets_s()

엔터키를 누를때까지의 정보를 저장하고 입력한다. 그리고 마지막에 입력된 ₩n 은 ₩0 로 교체되어 인자인 배열에 저장이 됩니다. gets()함수에 저장되면 함수자체를 한 행의 문자열로 간주한다는 것을 알 수가 있습니다.

puts()

이 두개의 함수의 장점은 처리 속도가 빠르다는 장점이 있으며 위 함수를 사용하기 위해 헤더파일 stdio.h 로 삽입해야 합니다.

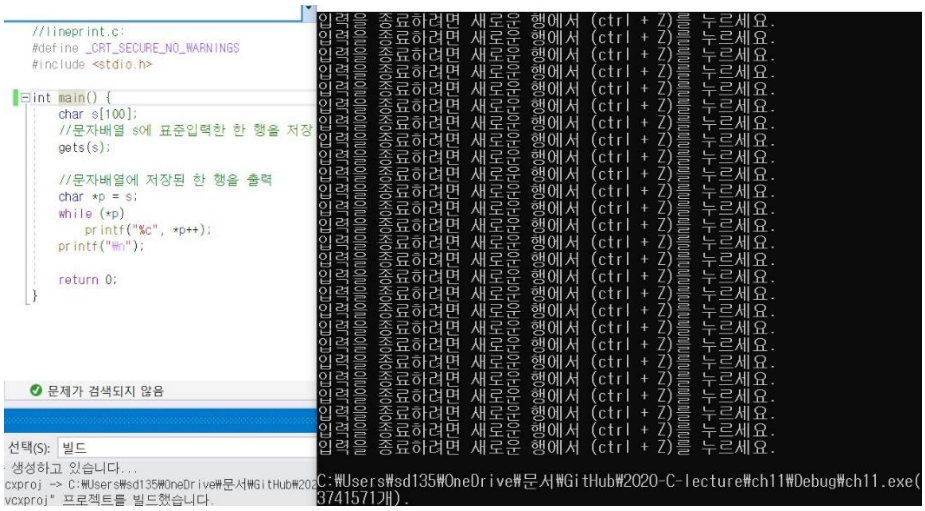
문자열 한 행 출력을 할 때 좋게 사용할 수 있는 것은 puts()가 효율적이라고 볼 수 있습니다.

또한 puts 나 gets 를 쓰면 'ctrl+z'를 이용해야 자동으로 빠져나갈 수 있습니다.

TIP

puts(),gets()는 문자열 입출력에 유리하고 scanf(), printf()는 그 외의 입출력에 유리한 것을 알 수 있습니다.

Lab 11-1



실습 예제 11-7

위 예제에서는 문자배열 함수(memcpy()/memchr/strlen)을 사용하였습니다.

우선 사용한 함수들에 대한 설명을 하겠습니다.

Memcpy()함수

Memcpy(데이터가 복사될 곳, 복사할 곳의 주소, 복사 할 크기); 로 구성을 하며

이 코드에서 src->dst 값을 복사하고 붙여 넣기 해 출력을 해준 것을 알 수있습니다.

TIP

만약 복사할 변수와 복사될 변수의 메모리영역이 겹치면 memmove() 함수를 이용해야 한다.

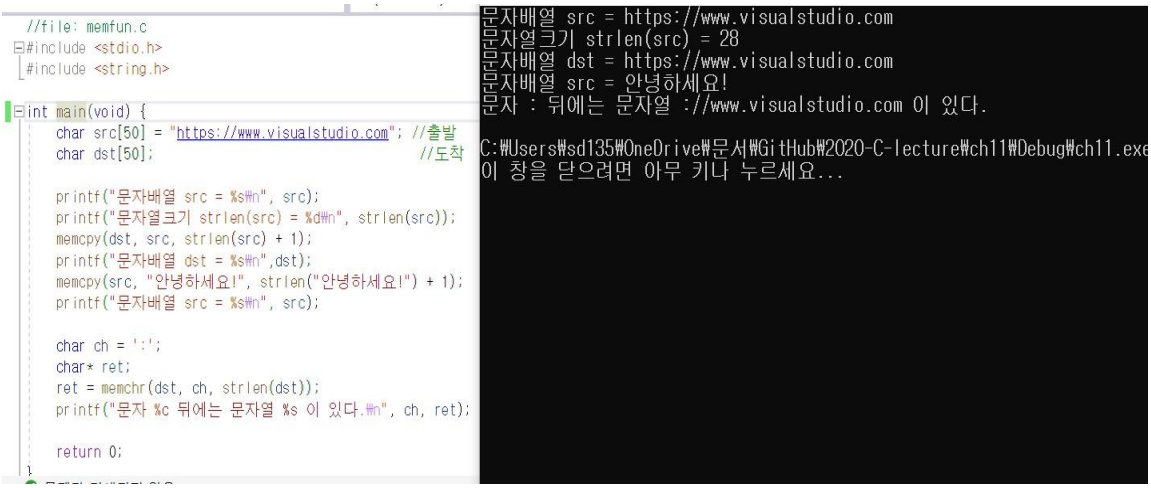
Memchr()함수

Memchr(검사할 메모리의 포인터, 검색할 문자,검사 영역 크기);로 구성을 하였습니다.

이 코드에서 ret 변수에 memchr()함수를 이용해 값을 저장해 준 것을 볼 수가 있습니다.

Strlen()함수

Strlen(변수)를 사용하면 해당 배열 변수의 길이를 알 수가 있습니다.



```c
//file: memfun.c
#include <stdio.h>
#include <string.h>

int main(void) {
    char src[50] = "https://www.visualstudio.com"; //출발
    char dst[50];                                  //도착

    printf("문자배열 src = %s₩n", src);
    printf("문자열크기 strlen(src) = %d₩n", strlen(src));
    memcpy(dst, src, strlen(src) + 1);
    printf("문자배열 dst = %s₩n",dst);
    memcpy(src, "안녕하세요!", strlen("안녕하세요!") + 1);
    printf("문자배열 src = %s₩n", src);

    char ch = ':';
    char* ret;
    ret = memchr(dst, ch, strlen(dst));
    printf("문자 %c 뒤에는 문자열 %s 이 있다.₩n", ch, ret);

    return 0;
}
```

```
문자배열 src = https://www.visualstudio.com
문자열크기 strlen(src) = 28
문자배열 dst = https://www.visualstudio.com
문자배열 src = 안녕하세요!
문자 : 뒤에는 문자열 ://www.visualstudio.com 이 있다.

C:₩Users₩sd135₩OneDrive₩문서₩GitHub₩2020-C-lecture₩ch11₩Debug₩ch11.exe
이 창을 닫으려면 아무 키나 누르세요...
```

실습 예제 11-8

헤더파일을 string.h 을 삽입해야 합니다.

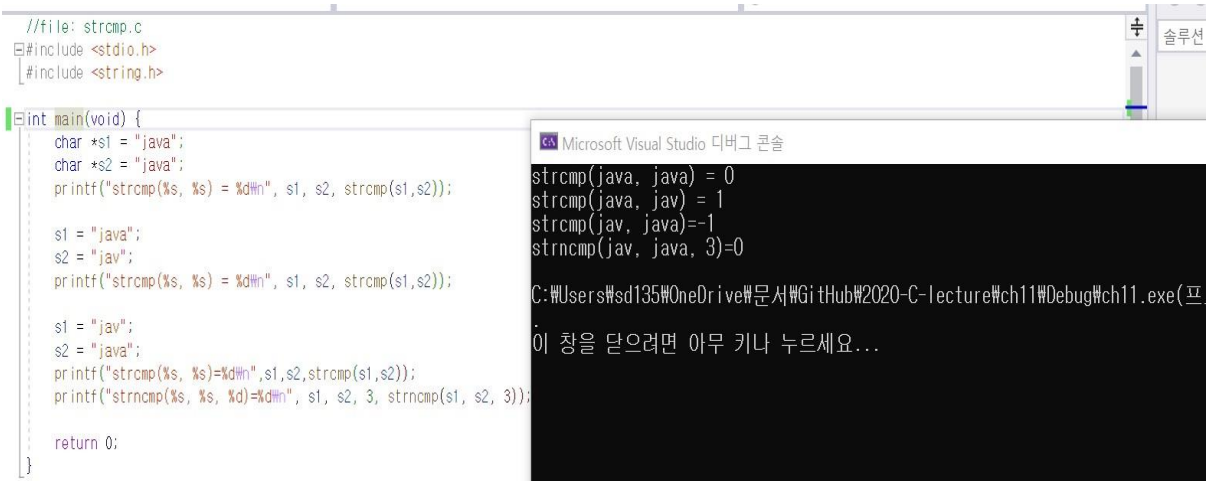
Strcmp()

Ex) strcmp("a","ab") = 음수 / strcmp("ab","a") = 양수 / strcmp("ab","ab") = 0

Strncmp() 함수

Ex) strncmp("java", "javA", 2) = 0 //인자 2 인 문자 둘째자리까지 비교하여 같으므로 0 이 나옵니다.

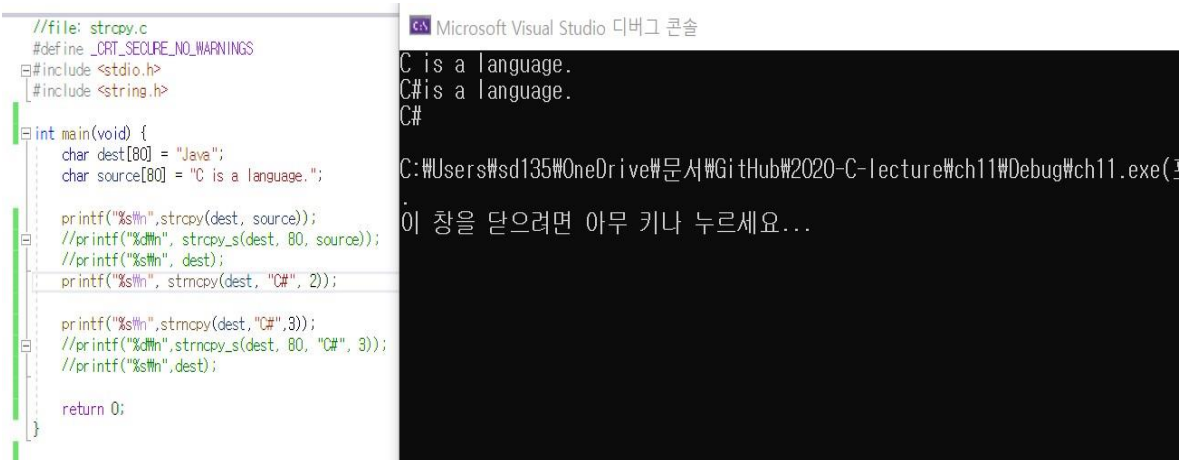
위 함수들은 두개의 변수를 서로 비교해 나타내 보여주는 역할을 합니다.



```c
//file: strcmp.c
#include <stdio.h>
#include <string.h>

int main(void) {
    char *s1 = "java";
    char *s2 = "java";
    printf("strcmp(%s, %s) = %d₩n", s1, s2, strcmp(s1,s2));

    s1 = "java";
    s2 = "jav";
    printf("strcmp(%s, %s) = %d₩n", s1, s2, strcmp(s1,s2));

    s1 = "jav";
    s2 = "java";
    printf("strcmp(%s, %s)=%d₩n",s1,s2,strcmp(s1,s2));
    printf("strncmp(%s, %s, %d)=%d₩n", s1, s2, 3, strncmp(s1, s2, 3));

    return 0;
}
```

```
Microsoft Visual Studio 디버그 콘솔
strcmp(java, java) = 0
strcmp(java, jav) = 1
strcmp(jav, java)=-1
strncmp(jav, java, 3)=0

C:₩Users₩sd135₩OneDrive₩문서₩GitHub₩2020-C-lecture₩ch11₩Debug₩ch11.exe(프
.
이 창을 닫으려면 아무 키나 누르세요...
```

실습 예제 11-9



```c
//file: strcpy.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void) {
    char dest[80] = "Java";
    char source[80] = "C is a language.";

    printf("%s₩n",strcpy(dest, source));
    //printf("%d₩n", strcpy_s(dest, 80, source));
    //printf("%s₩n", dest);
    printf("%s₩n", strncpy(dest, "C#", 2));

    printf("%s₩n",strncpy(dest,"C#",3));
    //printf("%d₩n",strncpy_s(dest, 80, "C#", 3));
    //printf("%s₩n",dest);

    return 0;
}
```

```
Microsoft Visual Studio 디버그 콘솔
C is a language.
C#is a language.
C#

C:₩Users₩sd135₩OneDrive₩문서₩GitHub₩2020-C-lecture₩ch11₩Debug₩ch11.exe(
.
이 창을 닫으려면 아무 키나 누르세요...
```

헤더파일 string.h 를 사용을 해줬습니다.

Strcpy / strcpy_s

Strcpy(목적지, 복사할 변수);

strcpy_s();를 사용하면 버퍼 크기가 데이터를 넘을 때 자동으로 막지 못합니다. 즉 지정한 크기를 넘어 복사하려 하면 프로그램을 멈추고 에러창을 띄워주세요라는 의미를 보는 것이 좋다고 할 수 있습니다.

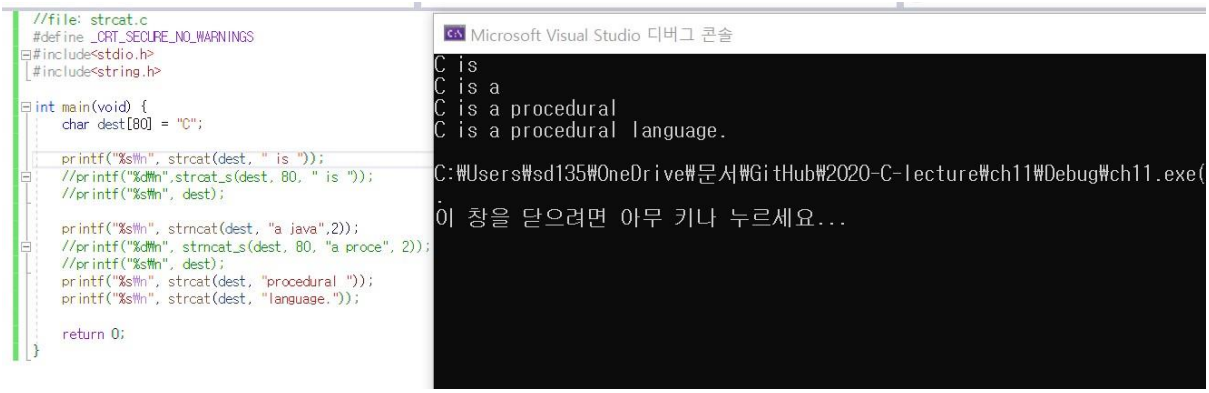
Strncpy / strncpy_s

Strncpy(목적지, 복사할 변수, 복사할 범위);

strncpy_s(); 는 strncpy 와 공식은 같지만 null 값까지 같이 포함시키기 때문에 복사할 범위에서 -1 을 해줘야 null 값까지 포함을 시킬 수 있습니다.

위 코드에서 사용한 것과 같이 복사할 변수(dest)에 변수(source)의 값을 넣어주고 그 후에 C#로 다시 변경해주는 것을 볼 수 있습니다.

실습 예제 11-10



헤더파일 string.h 를 사용을 해줬습니다.

strcat(); / strcat_s();

strcat(최종문자열, 붙일 문자열); / strcat_s(최종문자열, 붙일 문자열, 지정범위);

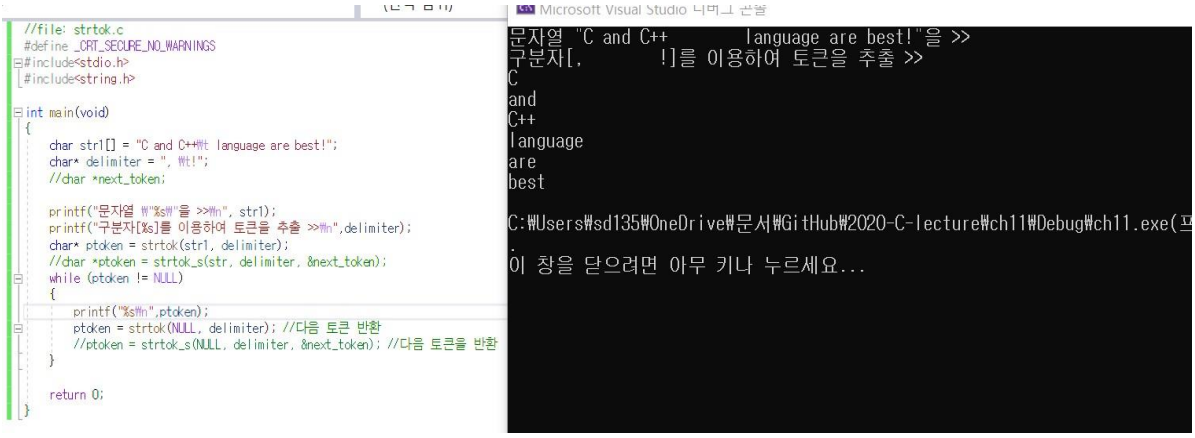
strcpy()와 다른 점이 있다면 strcpy 는 새 값으로 변수가 저장되는 거지만 strcat()은 기존의 값은 유지된 상태로 붙일 문자열까지 같이 덧붙여 쓰기 때문에 strcat()을 이용하려면 배열의 크기가 커야 된다는 것을 알게 되었습니다.

실습 예제 11-11

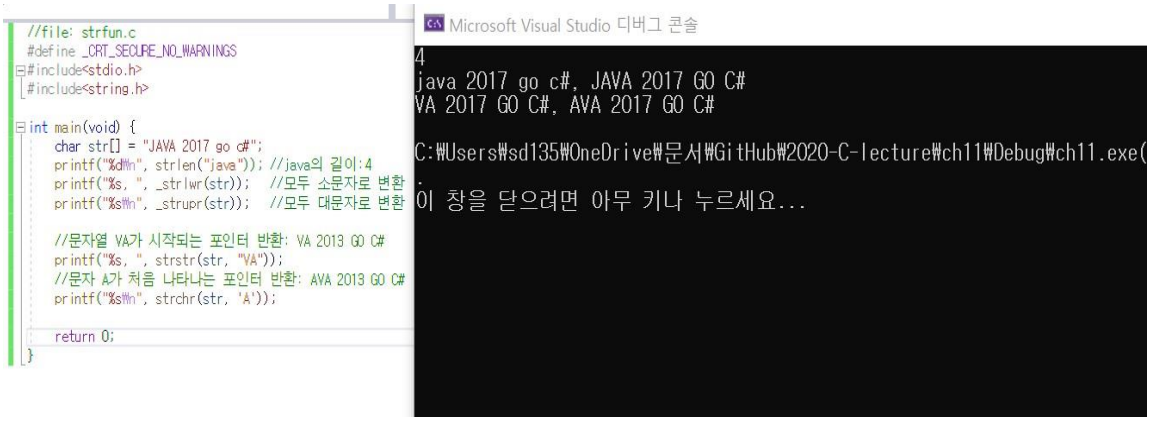
Strtok() / strtok_s()

간단히 말해문자열 자르기로 알면 되며 공식은 strtok("문자열","구분자")나눠 이렇게 사용합니다.

Strtok_s("문자열","구분자",범위);를 지정해 문자열을 잘라 출력을 하게 됩니다.



실습 예제 11-12



Strlen()

NULL 문자를 제외한 문자열 길이를 반환하는 함수입니다.

Strstr()

문자열 찾기 strstr(변수,"찾을 문자열") 출력 결과는 해당 변수의 전체 값이 출력된다.

Strchr()

문자열에서 문자 찾기로 구분되며 strchr(변수,"찾을 문자") 만약 변수에서 찾을 문자가

중복이 된경우 변수+숫자를 입력하면 다음 변수로 넘어가서 찾게 해줍니다.

Strlwr()

Strlwr(변수);를 사용하면 대문자에서 소문자로 변경이 됩니다.

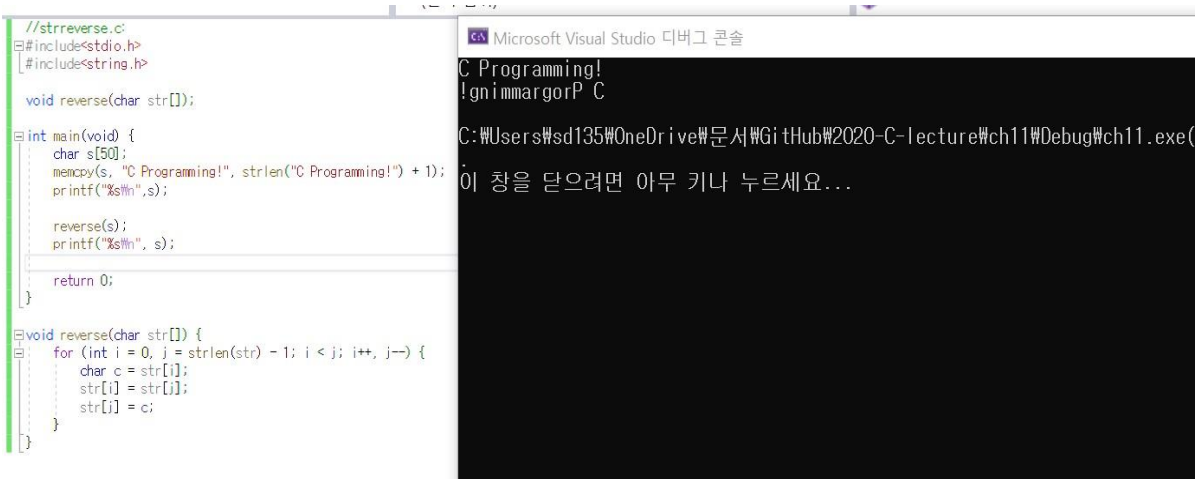
Strupr()

Strupr(변수);를 사용하면 소문자에서 대문자로 변경이 됩니다.

LAB 11-2

Reverse 를 이용해 for 문 역순 코드전용으로 만들고 함수원형을 선언하여 main 에다가 넣어 줍니다.

문자열 s 에 문자열을 memcpy 를 복사 후 reverse(s)넣어주고 출력을 하면 역순으로 출력이 됩니다.



```
//strreverse.c
#include<stdio.h>
#include<string.h>

void reverse(char str[]);

int main(void) {
    char s[50];
    memcpy(s, "C Programming!", strlen("C Programming!") + 1);
    printf("%s\n",s);

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

    return 0;
}

void reverse(char str[]) {
    for (int i = 0, j = strlen(str) - 1; i < j; i++, j--) {
        char c = str[i];
        str[i] = str[j];
        str[j] = c;
    }
}
```

```
Microsoft Visual Studio 디버그 콘솔
C Programming!
!gnimmargorP C

C:₩Users₩sd135₩OneDrive₩문서₩GitHub₩2020-C-lecture₩ch11₩Debug₩ch11.exe(
.
이 창을 닫으려면 아무 키나 누르세요...
```

실습 예제 11-13

포인터 배열과 이차원 배열을 비교하며 이차원 배열 같은 경우 문자열의 길이가 서로 다른 경우에는 '₩0' 문자가 들어가 낭비되는 메모리 공간이 있을 수 있지만 문자열 배열에서 문자열 처리는 문자열을 수정할 수 있는 장점이 있습니다.

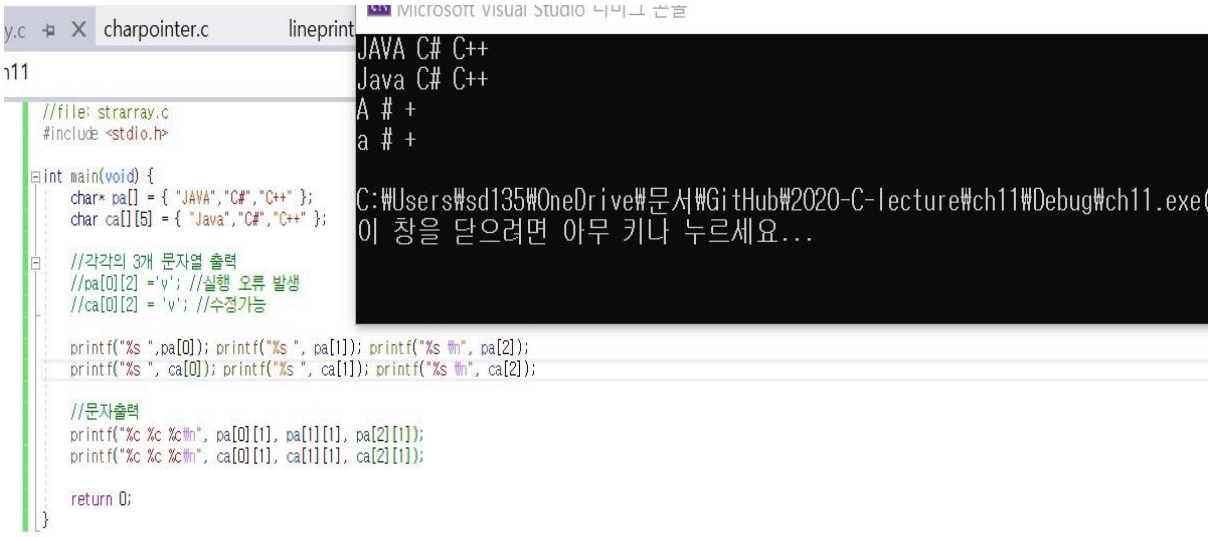
포인터 배열과 배열 포인터가 다르다는 것을 알게 되었습니다.

먼저 포인터 배열은 포인터가 여러 개 모여 배열이 된 것이고

배열 포인터는 배열 타입을 가리키는 하나의 포인터라는 점입니다.

여기서 사용한 것은 포인터 배열임을 알게 되어 이렇게 주소를 가리키고 다른 값을 출력해 주는 것을 볼 수 있습니다.

또한 여기서 한가지 특이점이 있었습니다. 바로 포인터 배열에서 일차 배열을 이용하였지만 문자출력을 보시면 본래의 pa 는 이차원 배열인 것을 알 수가 있습니다.

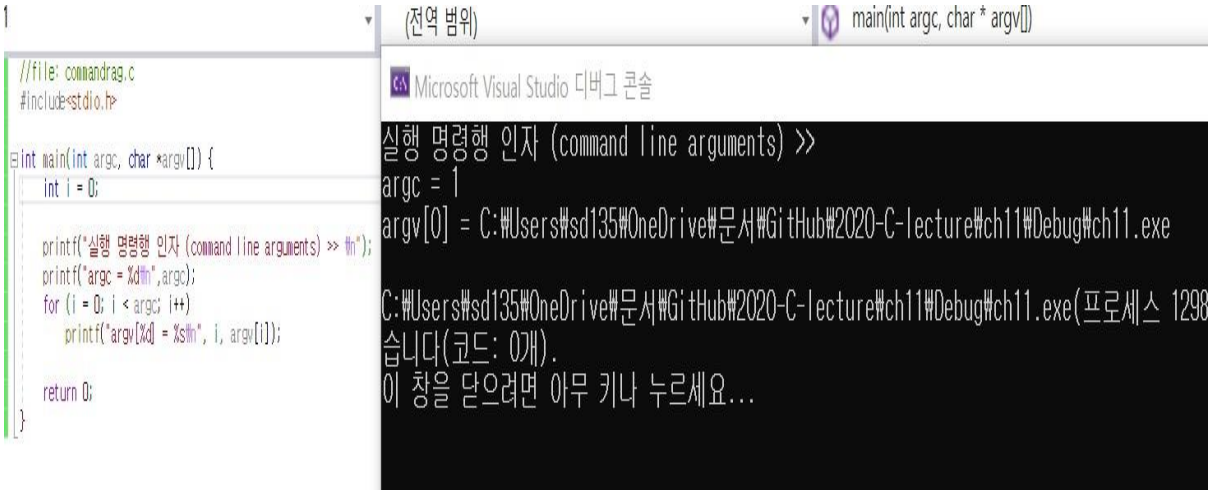


실습 예제 11-14

Main 매개변수 argc 와 *argv[]를 선언하고 명령인수를 이용해 인자를 기술합니다.

만약 처음으로 실행하게 된다면 폴더를 포함한 실행프로그램 이름이 지정되며

for 문을 이용해 출력을 하게 하는 구조입니다.

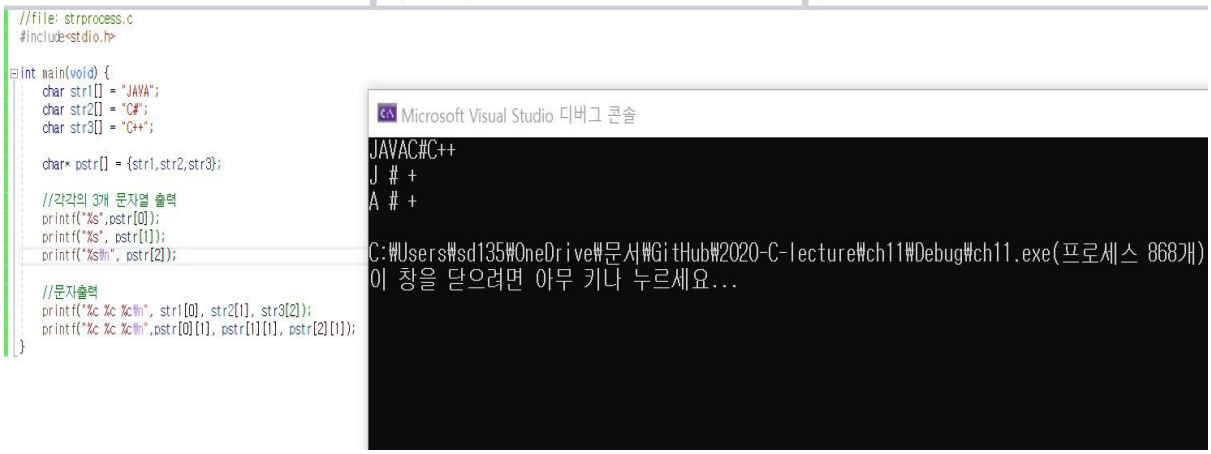


LAB 11-3

*pstr[]에 str1,2,3 주소를 가리키고 각각 출력하여 값을 나타내 주었습니다.

또한 여기서도 포인터배열(pstr)은 일차원 배열이지만 문자출력에서 보면

pstr 은 이차원 배열을 이용해 출력한 것을 볼 수가 있습니다.



CHAPTER 12

기억 부류에 관한 것을 간단히 설명해 보겠습니다.

먼저 기억 부류는 변수가 저장되는 위치에 따라 결정되는 변수의 여러 가지 성질을 의미하며 정적과 자동 이렇게 두가지로 나눌 수 있습니다.

먼저 auto 와 register 은 자동 기억 부류로 나눌 수 있으며 블록내에서만 사용 할 수 있고 블록 밖에 나오게 된다면 자동으로 소멸하게 됩니다.

특히 auto 같은 경우일반 변수 선언할 때 (auto)int a =0; 이런 형식으로 많이 쓰여 auto 는 자동 생략이 되어있는 것으로 보시면 됩니다.

다음으로 register 는 기계어 버전에 데이터 처리 및 계산하기 위해 사용을 합니다.

하드웨어 레지스터에 저장하고 유지관리를 함으로 결과를 메모리에 저장하는 오버헤드를 줄일 수 있습니다.

Static 은 지역과 전역 모든 변수에 이용 가능하며 대표적인 정적 부류라고 보시면 됩니다.

제가 언급한 세가지 기억 부류의 공통점은 새 변수의 선언에 사용되는 키워드입니다.

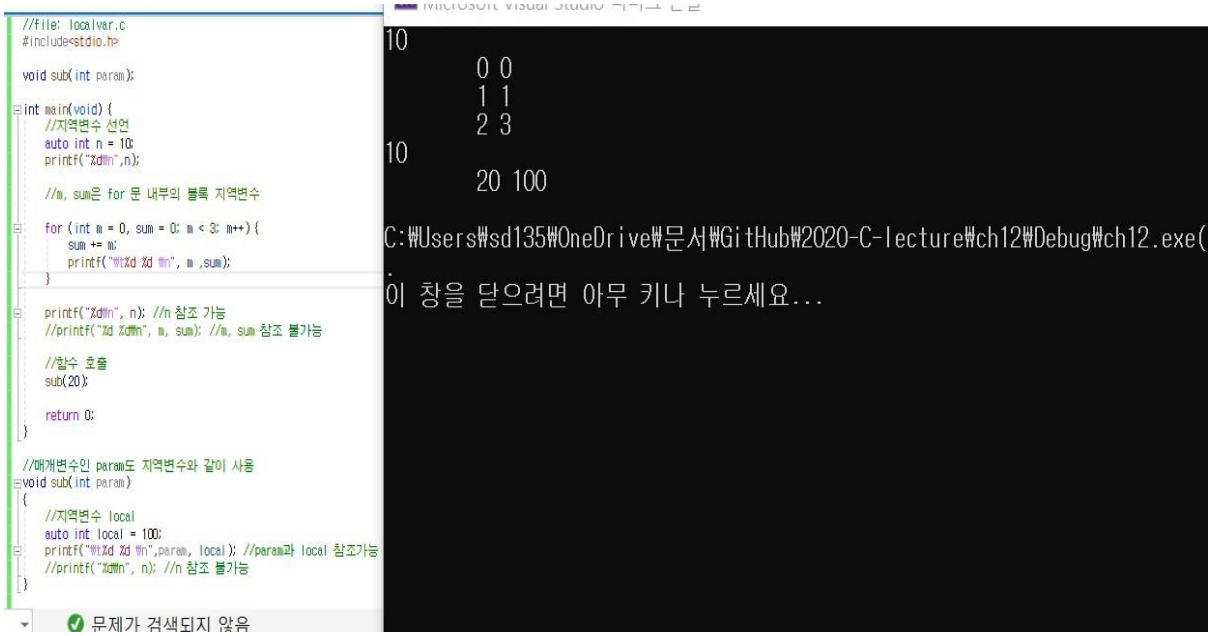
반면 extern 은 타 기억 부류와 달리 전역변수에만 사용이 가능하며 extern 의 기능은 컴파일러에게 변수가 이미 존재하거나 앞으로 사용하게 될 것이라는 것을 알리는 구문으로 이용이 됩니다. 하지만 extern 은 전역, 지역으로 자동으로 지역이 한정될 수가 있습니다.

위 기억 부류 중 register 은 가장 많이 안 쓴다는 것을 알 수가 있습니다.

간략하게 나타내면 아래 표와 같이 나타낼 수 있습니다.

기억부루	전역	지역	정적	레지스터
지정자	extern	auto	static	register
저장 장소	정적 데이터 영역	스택	정적 데이터 영역	CPU의 레지스터
선언 위치	함수의 외부	함수의 내부	함수의 내부	함수의 내부
통용 범위	프로그램 전체	함수의 내부	함수의 내부	함수의 내부
파괴 시기	프로그램 종료 시	함수 종료시	프로그램 종료시	함수 종료시
초기값	0으로 초기화	초기화되지 않음	0으로 초기화	초기화되지 않음

실습 예제 12-1



```
//file: localvar.c
#include<stdio.h>

void sub(int param);

int main(void) {
    //지역변수 선언
    auto int n = 10;
    printf("%d\n",n);

    //m, sum은 for 문 내부의 블록 지역변수

    for (int m = 0, sum = 0; m < 3; m++) {
        sum += m;
        printf("\t%d %d \n", m ,sum);
    }

    printf("%d\n", n); //n 참조 가능
    //printf("%d %d\n", m, sum); //m, sum 참조 불가능

    //함수 호출
    sub(20);

    return 0;
}

//매개변수인 param도 지역변수와 같이 사용
void sub(int param)
{
    //지역변수 local
    auto int local = 100;
    printf("\t%d %d \n",param, local); //param과 local 참조가능
    //printf("%d\n", n); //n 참조 불가능
}
```

```
10
        0 0
        1 1
        2 3
10
        20 100

C:\Users\sd135\OneDrive\문서\GitHub\2020-C-lecture\ch12\Debug\ch12.exe(
.
이 창을 닫으려면 아무 키나 누르세요...
```

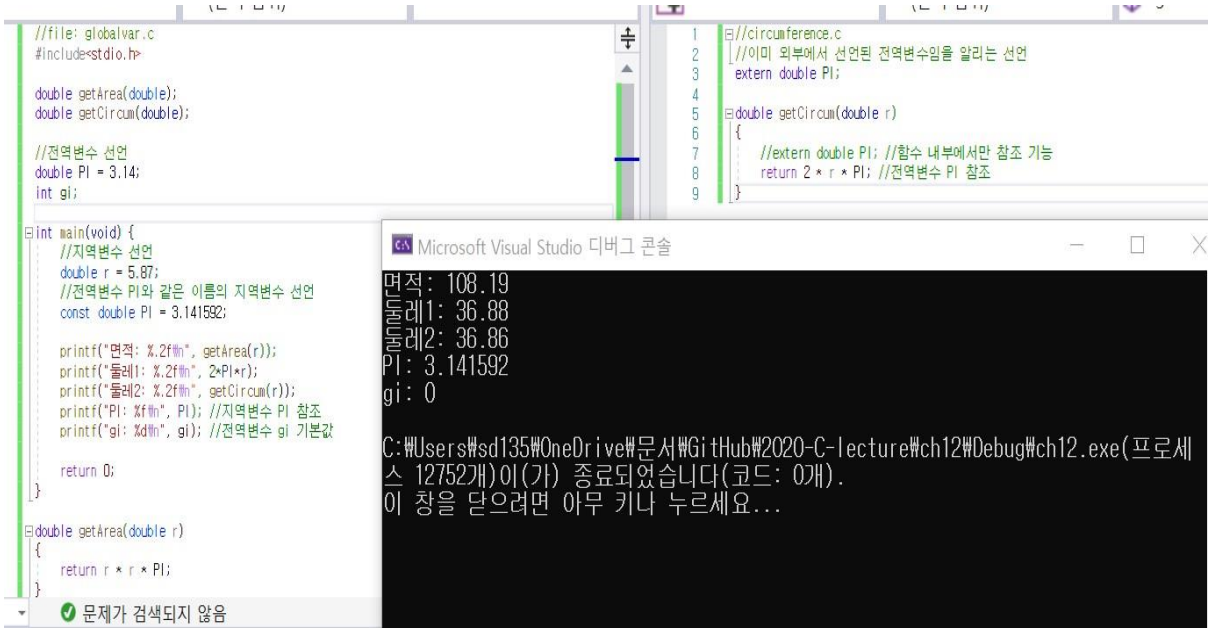
지역변수 Sub 함수를 함수원형을 선언해 main 에 넣고 main 에는 변수 n, m, sum 으로 나타내 출력을 했고 main 에 sub()에 param 를 20 을 넣어 sub()함수의 결과도 출력을 했음을 볼 수가 있습니다.

실습 예제 12-2/3

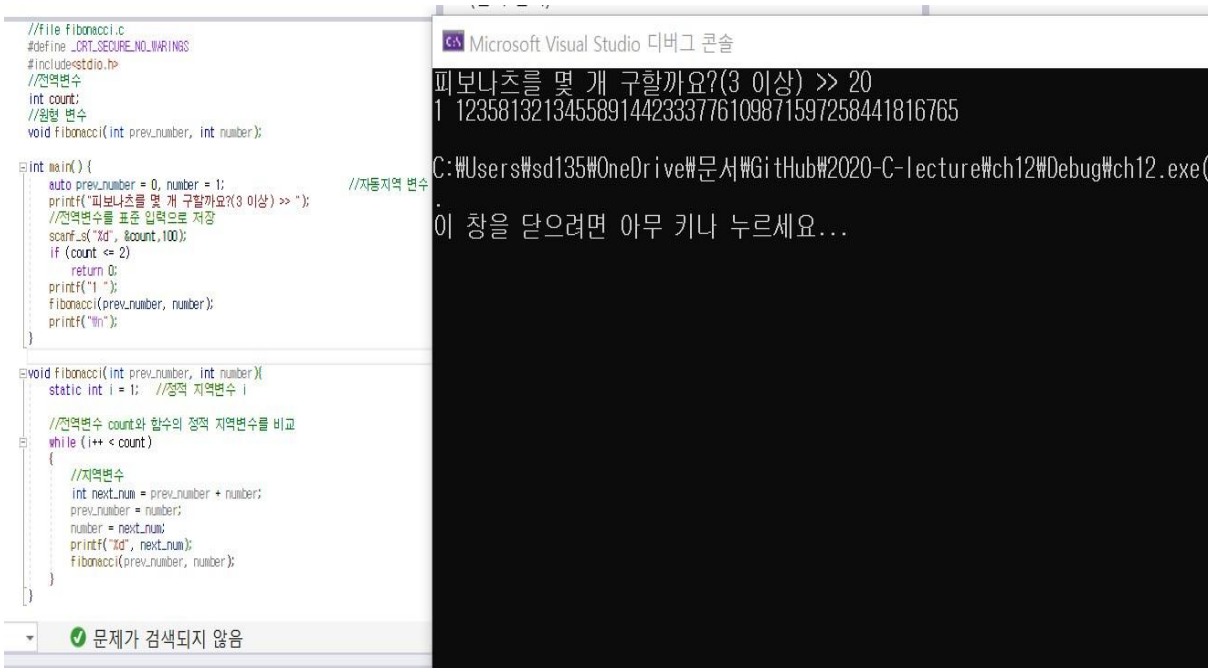
Main 페이지에 지역변수 getArea()와 main()을 놓았고 PI 를 전역변수를 사용했습니다.

다른파일에 gerCircum()을 이용하여 main 함수에 호출을 해주는 방법을 사용을 하였습니다.

getCircum()같은 경우는 둘레길이의 공식을 넣었고 getArea()에서는 면적을 넣어서 main 에 호출을 해 불러주었습니다.



LAB 12-1



피보나츠는 대표적 공식으로 F(n) = F(n-1)+F(n-2) 형식으로 계산해 수가 커지는 것을 말한다.

이와 같이 코드를 작성해 우선 fibonacci() 함수를 선언해 prev_number, number 의 변수를

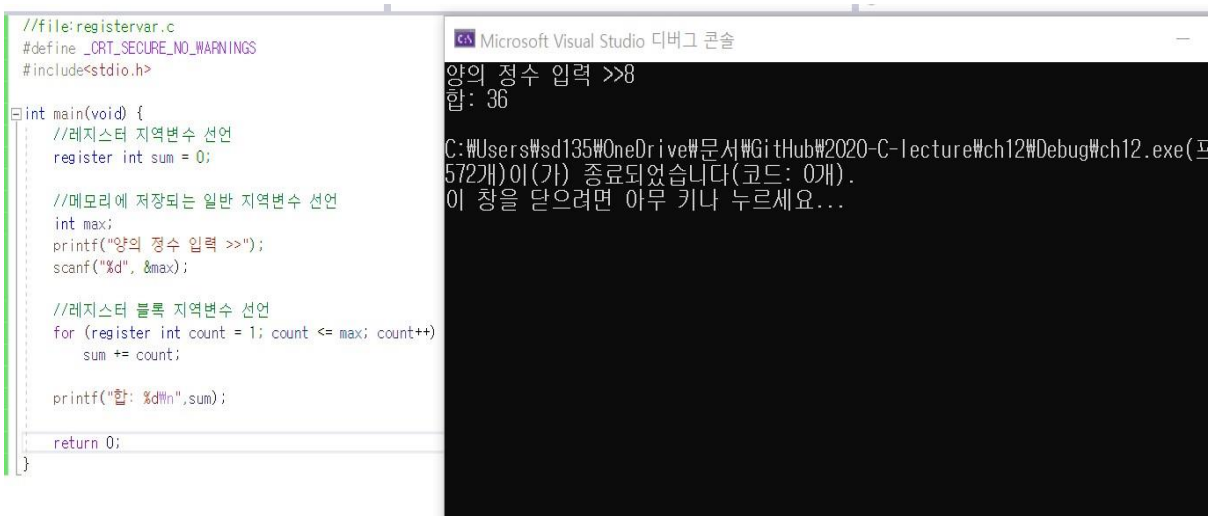
선언하고 while 문을 사용하여 next_num 과 prev_number 와 number 의 값을 교환하고 main 에 scanf 입력 값을 지정하고 Fibonacci(); 함수를 호출하여 출력을 하는 시스템으로 이뤄져 있다.

실습 예제 12-4

Register 지역변수와 일반 지역변수를 선언했고 scanf 의 값을 입력받아 범위를 지정하여 for 문으로 출력해 결과를 출력하는 코드임을 알 수 있습니다.

레지스터는 잘 쓰지 않지만 일반 변수보다 처리 속도가 빠른 장점이 있습니다.

또한 레지스터 변수에 주소연산자(&)를 사용하면 문법오류가 발생합니다.



```c
//file:registervar.c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>

int main(void) {
    //레지스터 지역변수 선언
    register int sum = 0;

    //메모리에 저장되는 일반 지역변수 선언
    int max;
    printf("양의 정수 입력 >>");
    scanf("%d", &max);

    //레지스터 블록 지역변수 선언
    for (register int count = 1; count <= max; count++)
        sum += count;

    printf("합: %d\n",sum);

    return 0;
}
```

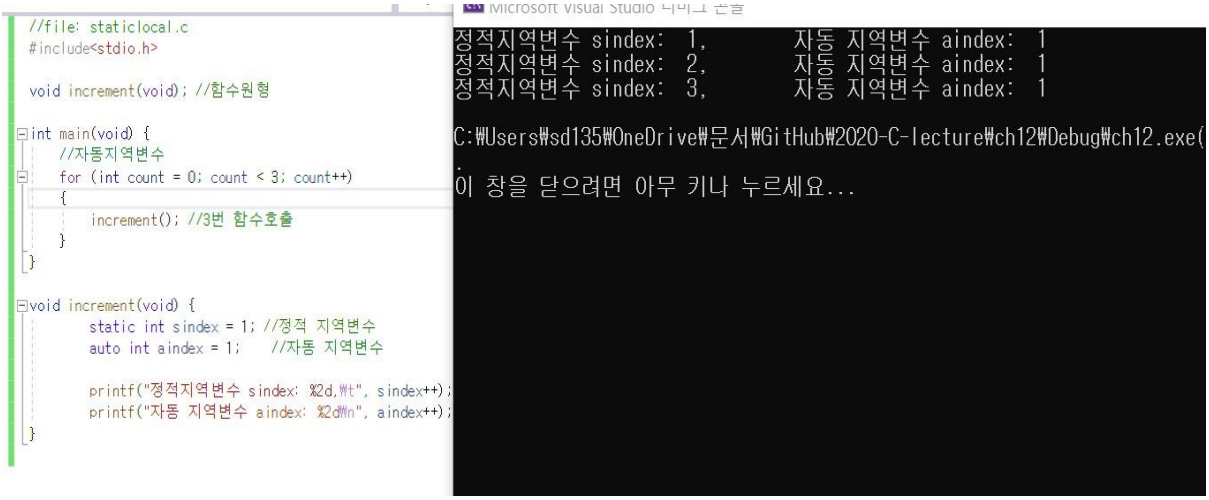
```
Microsoft Visual Studio 디버그 콘솔
양의 정수 입력 >>8
합: 36

C:\Users\sd135\OneDrive\문서\GitHub\2020-C-lecture\ch12\Debug\ch12.exe(프
572개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

실습 예제 12-5

Increment 함수에 변수와 출력을 설정합니다.

main 에서는 for 문을 이용하여 increment 함수를 호출하여 출력을 하는 시스템으로

구성이 되어있습니다.

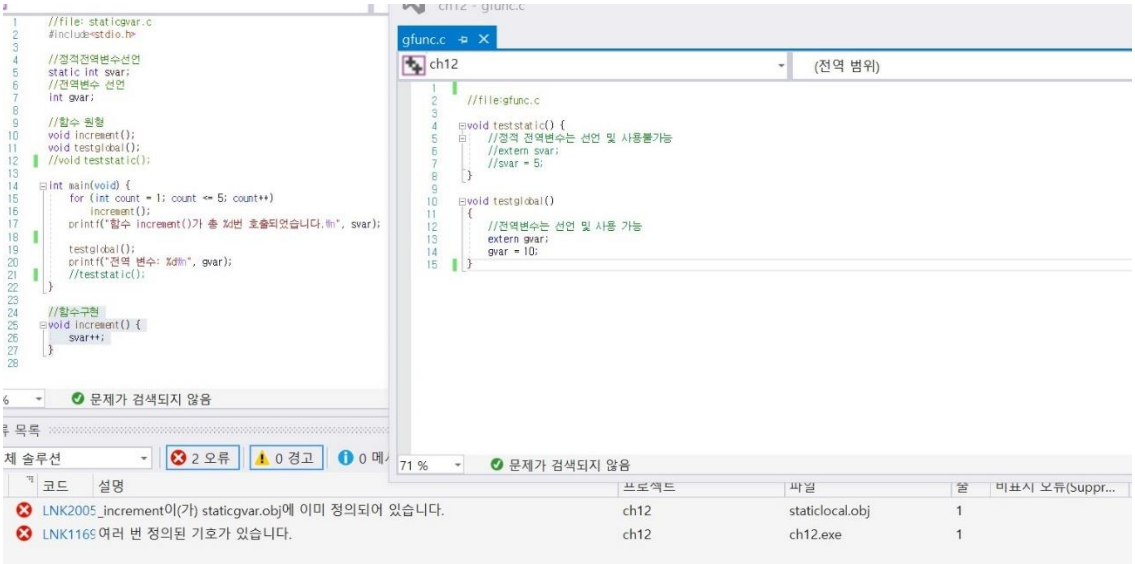


실습 예제 12-6/7

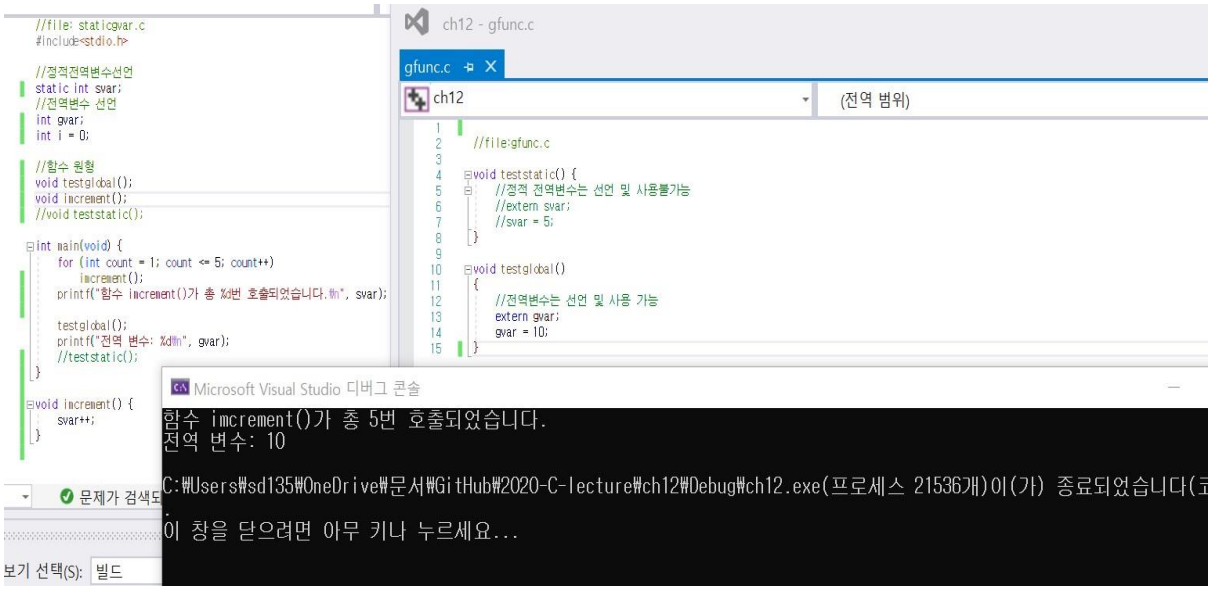
lnk 오류 2005 가 발생이 되어서 오류의 원인을 찾던 중 increment()함수의 오류라 increment()함수를 삭제하거나 원형만 선언하는 등 다양한 방법을 적용해 봤습니다.

챕터 11 을 하나의 파일로 만들어 함수가 중복이 되어 오류가 발생한 것을 알게 되었습니다.

(오류코드 LNK2005)



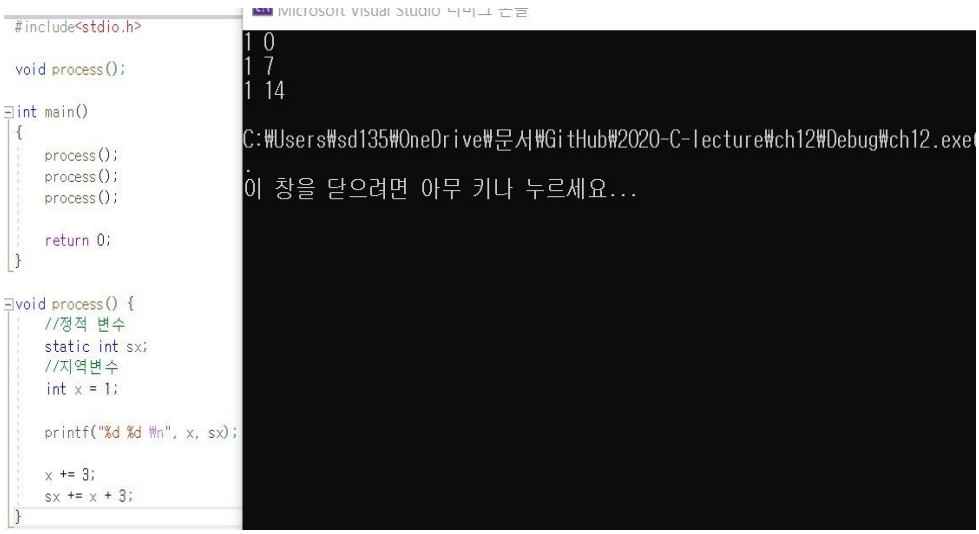
(해결 후 결과)



이 오류를 해결하기 위해 increment 함수를 imcrement 함수로 수정을 하여 나타내어 주었습니다.

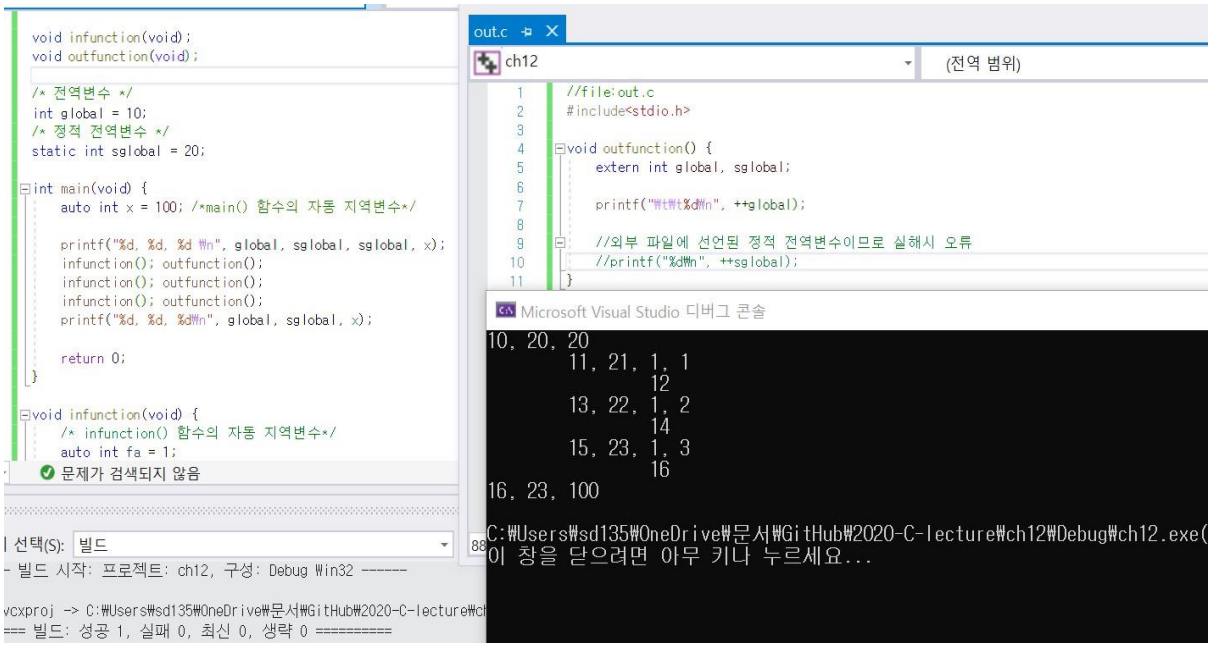
우선 이 코드는 imcrement(교재: increment)함수의 지역변수와 testglobal()의 전역변수를 호출하여 줍니다. Main 에 for 문을 사용하여 increment 함수를 여러 번 호출하고 testglobal()에 있는 값을 main 으로 옮겨 출력해 준 것을 볼 수가 있습니다.

LAB12-2



위 코드에서 보듯이 process()함수에 정적 변수와 지역변수를 만들고 main 으로 호출하여 출력 값을 서로 비교하는 것을 볼 수가 있습니다. 지역변수는 호출할 때 마다 1 로 초기화가 되어 숫자가 1 로 고정이 되지만 정적 변수는 한번 저장하면 안 바뀌기 때문에 호출하면서 숫자가 커지는 것을 볼 수가 있습니다.

실습 예제 12-8/9



외부 outfunction() 함수안에 전역변수 global 와 sglobal 을 가져와 출력을 합니다. 지역변수

fa 와 fs 를 함수 infunction()을 선언하고 함수원형으로 main 에 사용할 준비를 합니다.

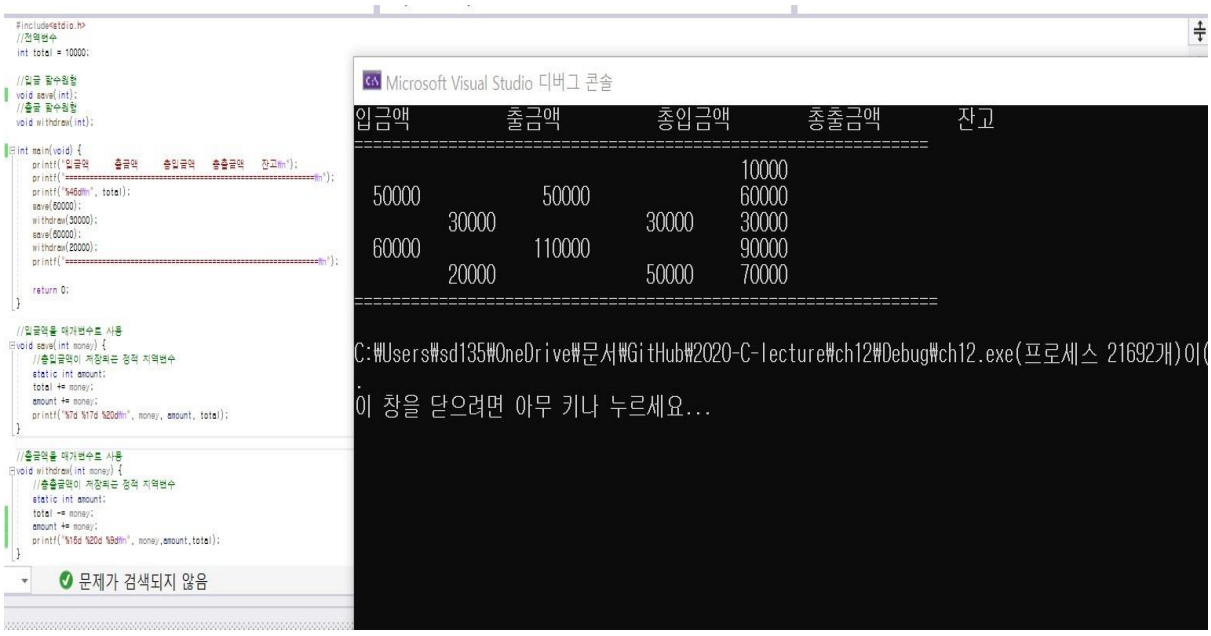
Main 에 함수 호출을 함으로 결과 값이 위와 같이 출력이 됩니다.

LAB 12-3

Save()함수와 withdraw()함수에 money 라는 지역변수를 넣었습니다.

각각의 함수별로 정한 기능을 코드로 구현을 하였습니다.

다음으로 함수원형으로 호출을 하고 main에 2개의 함수를 각각 호출하여 콘솔창을 위와 같이 출력을 하게 되었습니다.



==========

Chapter 13(구조체와 공용체)

우선 구조체는 struct 로 표현을 하며 연관된 변수들을 묶어 관리하여 데이터 관리에 유용하며 각 멤버들의 메모리 시작주소가 다릅니다.

또한 구조체는 멤버 중 가장 큰 변수의 크기 값을 기준으로 잡아 크기 순으로 나머지 변수를 순서대로 배치하여 크기를 결정합니다.

Ex) 가장 큰 멤버변수 long / 사용할 멤버변수 – char / int 각 2 개

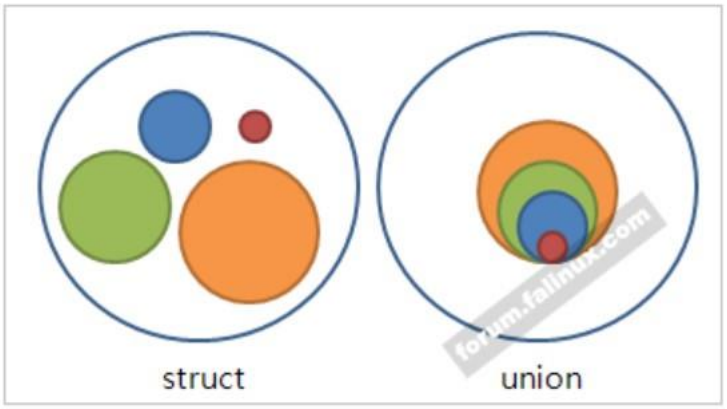
Struct 에 저장되는 크기 = 13byte / 실제: 18byte

다음으로 공용체는 union 으로 사용을 하며 각 멤버들의 시작주소가 동일합니다.

또한 멤버변수 중 가장 큰 크기 값을 공유를 합니다.

Ex) 가장 큰 멤버변수 long(8byte) / 사용할 멤버변수-char / int

메모리를 long 에서 공유를 할 수 있습니다.



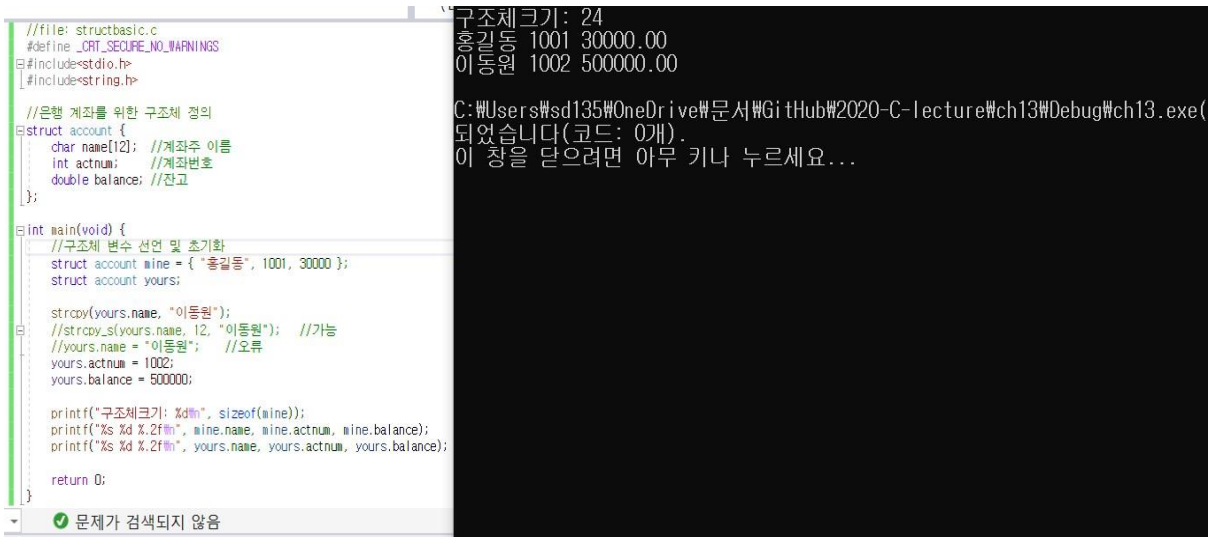
그림으로 나타내면 작성하면 이렇게 나타낼 수 있습니다.

또한 연관되어 있는 자료형을 하나의 자료형으로 이용해 유도자료형이라고 합니다.

TIP

유도자료형: 사용자정의 자료형 / 구조체 / 공용체 / 열거형 / 배열 / 포인터 등이 포함이 됩니다.

실습 예제 13-1



```
//file: structbasic.c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<string.h>

//은행 계좌를 위한 구조체 정의
struct account {
    char name[12];  //계좌주 이름
    int actnum;     //계좌번호
    double balance; //잔고
};

int main(void) {
    //구조체 변수 선언 및 초기화
    struct account mine = { "홍길동", 1001, 30000 };
    struct account yours;

    strcpy(yours.name, "이동원");
    //strcpy_s(yours.name, 12, "이동원");   //가능
    //yours.name = "이동원";    //오류
    yours.actnum = 1002;
    yours.balance = 500000;

    printf("구조체크기: %d₩n", sizeof(mine));
    printf("%s %d %.2f₩n", mine.name, mine.actnum, mine.balance);
    printf("%s %d %.2f₩n", yours.name, yours.actnum, yours.balance);

    return 0;
}
```

문제가 검색되지 않음

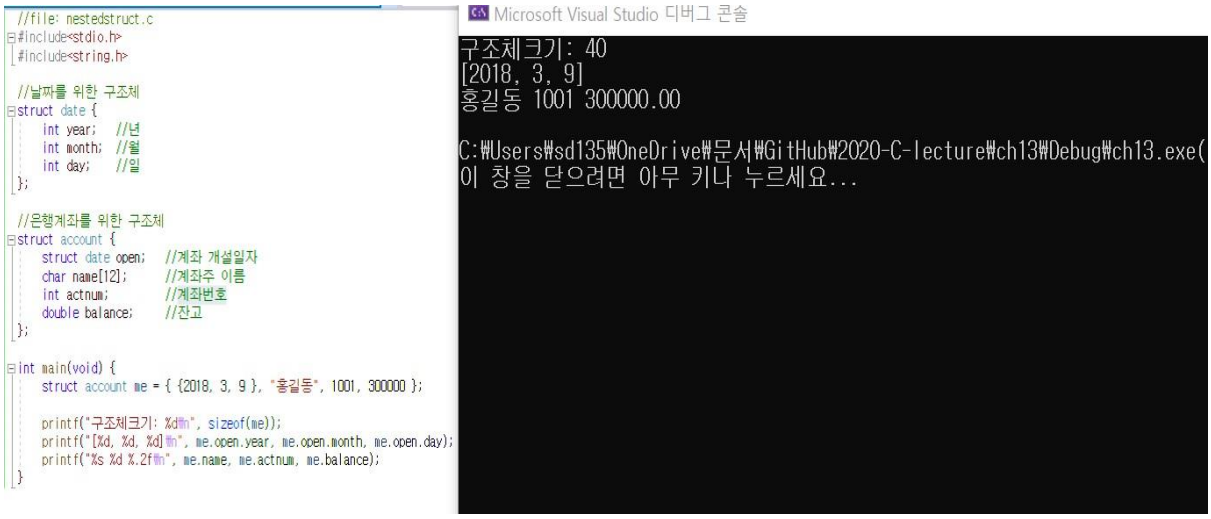
```
구조체크기: 24
홍길동 1001 30000.00
이동원 1002 500000.00

C:₩Users₩sd135₩OneDrive₩문서₩GitHub₩2020-C-lecture₩ch13₩Debug₩ch13.exe(
되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

Struct 틀 account 를 만들어 필요한 변수를 선언하고 main 에서 mine 과 yours 를 구조체를 자료형으로 선언을 하였습니다. 그 중 mine 은 선언과 동시에 한번에 초기화를 해주었고 yours 는 각각 넣어 주고 출력을 하였습니다.

실습 예제 13-2

Struct 틀 date 와 account 를 선언하였습니다. date 구조체는 account 에 자료형 open 으로선언을 해주었습니다.

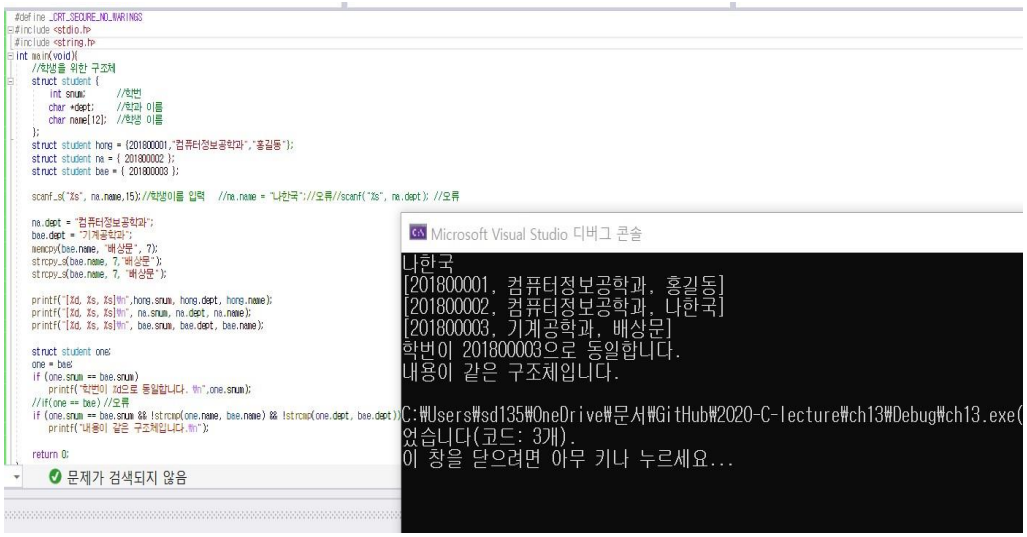
이를 통해 구조체는 다른 구조체에서 선언을 해줄 수 있음을 알 수 있습니다. main 에서 account 구조체를 자료형으로 me 로 선언과 초기화를 해주고 출력을 해주었습니다.



```c
//file: nestedstruct.c
#include<stdio.h>
#include<string.h>

//날짜를 위한 구조체
struct date {
    int year;   //년
    int month;  //월
    int day;    //일
};

//은행계좌를 위한 구조체
struct account {
    struct date open;   //계좌 개설일자
    char name[12];      //계좌주 이름
    int actnum;         //계좌번호
    double balance;     //잔고
};

int main(void) {
    struct account me = { {2018, 3, 9 }, "홍길동", 1001, 300000 };

    printf("구조체크기: %d₩n", sizeof(me));
    printf("[%d, %d, %d]₩n", me.open.year, me.open.month, me.open.day);
    printf("%s %d %.2f₩n", me.name, me.actnum, me.balance);
}
```

```
Microsoft Visual Studio 디버그 콘솔
구조체크기: 40
[2018, 3, 9]
홍길동 1001 300000.00

C:₩Users₩sd135₩OneDrive₩문서₩GitHub₩2020-C-lecture₩ch13₩Debug₩ch13.exe(
이 창을 닫으려면 아무 키나 누르세요...
```

실습 예제 13-3



```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
int main(void){
    //학생을 위한 구조체
    struct student {
        int snum;       //학번
        char *dept;     //학과 이름
        char name[12];  //학생 이름
    };
    struct student hong = {201800001,"컴퓨터정보공학과","홍길동"};
    struct student na = { 201800002 };
    struct student bae = { 201800003 };

    scanf_s("%s", na.name,15);//학생이름 입력   //na.name = "나한국";//오류//scanf("%s", na.dept); //오류

    na.dept = "컴퓨터정보공학과";
    bae.dept = "기계공학과";
    memcpy(bae.name, "배상문", 7);
    strcpy_s(bae.name, 7,"배상문");
    strcpy_s(bae.name, 7, "배상문");

    printf("[%d, %s, %s]₩n",hong.snum, hong.dept, hong.name);
    printf("[%d, %s, %s]₩n", na.snum, na.dept, na.name);
    printf("[%d, %s, %s]₩n", bae.snum, bae.dept, bae.name);

    struct student one;
    one = bae;
    if (one.snum == bae.snum)
        printf("학번이 %d으로 동일합니다. ₩n",one.snum);
    //if(one == bae) //오류
    if (one.snum == bae.snum && !strcmp(one.name, bae.name) && !strcmp(one.dept, bae.dept))
        printf("내용이 같은 구조체입니다.₩n");

    return 0;
```

문제가 검색되지 않음

```
Microsoft Visual Studio 디버그 콘솔
나한국
[201800001, 컴퓨터정보공학과, 홍길동]
[201800002, 컴퓨터정보공학과, 나한국]
[201800003, 기계공학과, 배상문]
학번이 201800003으로 동일합니다.
내용이 같은 구조체입니다.

C:₩Users₩sd135₩OneDrive₩문서₩GitHub₩2020-C-lecture₩ch13₩Debug₩ch13.exe(
었습니다(코드: 3개).
이 창을 닫으려면 아무 키나 누르세요...
```

Main 안에 student 구조체를 생성하고 struct 밖에 구조체 변수를 선언하였습니다. Hong 은 선언과 동시에 초기화를 해주고 na 와 bae 는 문자열 함수를 이용해 값을 입력 받게 하고 one 이라는 구조체 변수를 새로 선언하여 bae 와 비교하여 했습니다. 하지만

여기서 bae == one 으로 이렇게 직접 비교는 할 수가 없기 때문에 코드에서 보시는 바와 같이 구조체 멤버를 각각 비교를 했습니다. 만약 여러 개를 비교하고 싶으면 문자열 함수 strcmp()를 이용해 모두 비교할 수가 있습니다.

실습 예제 13-4

```c
//file : union.c
#include <stdio.h>

//유니온 구조체를 정의하면서 변수 data1도 선언한 문장
union data {
    char ch; //문자형
    int cnt; //정수형
    double real;    //실수형
}data1;//data1은 전역변수

int main(void) {
    union data data2 = { 'A' }; //첫 멤버인 char형으로만 초기화 가능    union data data2 ={10.3};//컴파일시 경고 발생
    union data data3 = data2;   //다른 변수로 초기화 가능
    printf("%d %d \n", sizeof(union data), sizeof(data3));

    //멤버 ch에 저장
    data1.ch = 'a';
    printf("%c %d %f\n",data1.ch,data1.cnt, data1.real);
    //멤버 cnt에 저장
    data1.cnt = 100;
    printf("%c %d %f\n",data1.ch, data1.cnt, data1.real);
    //멤버 real에 저장
    data1.real = 3.156759;
    printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);

    return 0;
}
```

문제가 검색되지 않음

```
Microsoft Visual Studio 디버그 콘솔
8 8
a 97 0.000000
d 100 0.000000
N -590162866 3.156759

C:\Users\sd135\OneDrive\문서\GitHub\2020-C-lecture\ch13\Debug\ch13.exe(
.
이 창을 닫으려면 아무 키나 누르세요...
```

union 이라는 공용체 변수 하나로 각 값들을 수정하여 결과를 출력하였습니다.

LAB 13-1

```c
//file: structcity.c
#include <stdio.h>
#include <string.h>

//지구 위치 구조체
struct position {
    double latitude;    //위도
    double longtitude;  //경도
};

int main(void) {
    //도시 정보 구조체
    struct city {
        char* name; //이름
        struct position place;  //위치
    };
    struct city seoul, newyork; seoul, newyork;

    seoul.name = "서울";
    seoul.place.latitude = 37.33;
    seoul.place.longtitude = 126.58;

    newyork.name = "뉴욕";
    newyork.place.latitude = 40.8;
    newyork.place.longtitude = 73.9;

    printf("[%s] 위도 = %.1f 경도 = %.1f\n",seoul.name, seoul.place.latitude);
    printf("[%s] 위도 = %.1f 경도 = %.1f\n",newyork.name,newyork.place.longtitude,newyork.place.latitude);
}
```

```
Microsoft Visual Studio 디버그 콘솔
[서울] 위도 = 37.3 경도 = 0.0
[뉴욕] 위도 = 73.9 경도 = 40.8

C:\Users\sd135\OneDrive\문서\GitHub\2020-C-lecture
.
이 창을 닫으려면 아무 키나 누르세요...
```

구조체의 변수를 seoul 과 newyork 를 주어 위도와 경도를 입력하고 출력을 하였습니다.

실습 예제 13-5

```c
//file: typedef.c
#include <stdio.h>

//함수 외부에서 정의된 자료형은 이후 파일에서 사용 가능
typedef unsigned int budget;

int main(void) {
    //새로운 자료형 budget 사용
    budget year = 24500000;

    //함수 내부에서 정의된 자료형은 이후 함수재부에서만 사용 가능
    typedef int profit;
    //새로운 자료형 profit 사용
    profit month = 4600000;

    printf("올 예산은 %d, 이달의 이익은 %d 입니다. \n", year, month);

    return 0;
}

void test(void)
{
    //새로운 자료형 budget 사용
    budget year = 24500000;

    //profit은 이함수에서는 사용 불가, 오류 발생
    //profit year;
}
```

문제가 검색되지 않음

```
Microsoft Visual Studio 디버그 콘솔
올 예산은 24500000, 이달의 이익은 4600000 입니다.

C:\Users\sd135\OneDrive\문서\GitHub\2020-C-lecture\ch13\Debug\ch13.exe(프
이 창을 닫으려면 아무 키나 누르세요...
```

Typedef 은 새 자료형의 이름을 재정의 할 수 있도록 합니다.

실습 예제 13-6

```c
//file: typedefstruct.c
#include<stdio.h>

struct date {
    int year;   //년
    int month;  //월
    int day;    //일
};

//struct date 유형을 간단히 date 형으로 사용하기 위한 구문
typedef struct date date;

int main(void) {
    //구조체를 정의함녀서 바로 자료형 software로 정의하기 윟나 구문
    typedef struct {
        char title[30]; //제목
        char company[30];   //제작회사
        char kinds[30];     //종류
        date release;       //출시일
    } software;

    software vs = { "비주얼스튜디오 커뮤니티", "MS", "통합개발환경", {2018,8,29} };

    printf("제품명: %s\n", vs.title);
    printf("회사: %s\n", vs.company);
    printf("종류: %s\n", vs.kinds);
    printf("출시일: %d. %d. %d\n", vs.release.year, vs.release.month, vs.release.day);

    return 0;
}
```

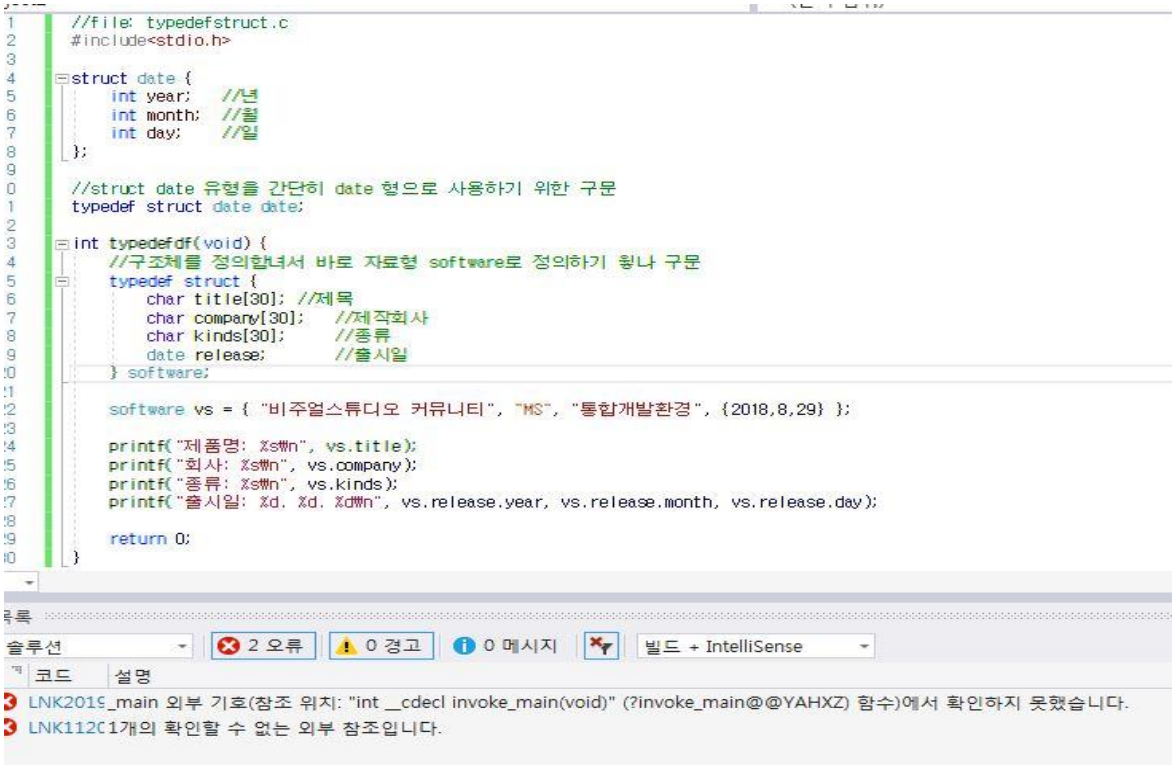
```
Microsoft Visual Studio 디버그 콘솔
제품명: 비주얼스튜디오 커뮤니티
회사: MS
종류: 통합개발환경
출시일: 2018. 8. 29

C:\Users\Administrator\source\repos\Project2\Debug\Project2.exe(12208 프로세스)
이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

Typedef 를 이용하면 위 코드와 같이 struct date 를 선언하고 date 로 재선언을 할 수가 있습니다.

이쪽에서 software 은 새로운 자료형이기 때문에 위와 같이 만들고 출력을 해줄 수가 있다.

또한 위코드를 출력할 때 밑에 그림과 같이 Lnk2019 오류가 나서 찾아보던 중

프로젝트 – 속성 – 링커 – 입력에 콘솔창을 창으로 바꾸거나 창에서 콘솔창을 바꾸는 경우가 있었고 .c 를 .cpp 로 바꿔 되는 경우가 있다고 해서 둘 다 해보았지만 오류가 변하지 않게 되어서 다시 코드를 보던 중 원래 main 으로 사용하던 것을 이름을 바꿔 에러가 난 것이었습니다. 이번 오류로 main 은 꼭 필요하다는 것을 알았습니다.



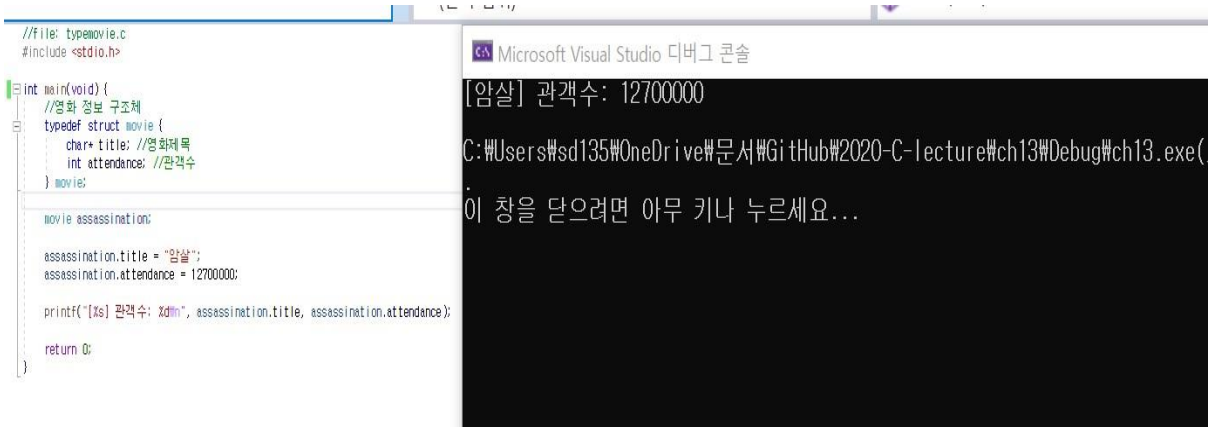
```c
//file: typedefstruct.c
#include<stdio.h>

struct date {
    int year;   //년
    int month;  //월
    int day;    //일
};

//struct date 유형을 간단히 date 형으로 사용하기 위한 구문
typedef struct date date;

int typedefdf(void) {
    //구조체를 정의함녀서 바로 자료형 software로 정의하기 윟나 구문
    typedef struct {
        char title[30]; //제목
        char company[30];   //제작회사
        char kinds[30];     //종류
        date release;       //출시일
    } software;

    software vs = { "비주얼스튜디오 커뮤니티", "MS", "통합개발환경", {2018,8,29} };

    printf("제품명: %s₩n", vs.title);
    printf("회사: %s₩n", vs.company);
    printf("종류: %s₩n", vs.kinds);
    printf("출시일: %d. %d. %d₩n", vs.release.year, vs.release.month, vs.release.day);

    return 0;
}
```

```
목록
솔루션 | 2 오류 | 0 경고 | 0 메시지 | 빌드 + IntelliSense
코드 | 설명
LNK2019 _main 외부 기호(참조 위치: "int __cdecl invoke_main(void)" (?invoke_main@@YAHXZ) 함수)에서 확인하지 못했습니다.
LNK1120 1개의 확인할 수 없는 외부 참조입니다.
```

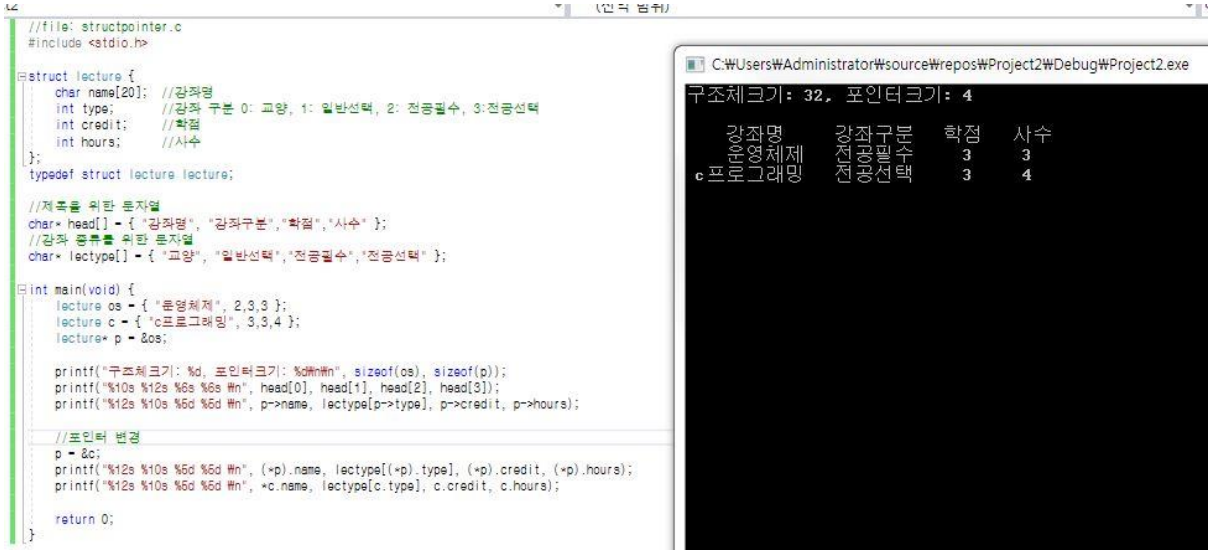
LAB 13-2

Struct 를 movie 라는 자료형으로 사용을 하였습니다.

movie 를 assassination 의 변수로 바꾸어 assassination.변수로 입력을 받아 출력을 하였습니다.



실습 예제 13-7



lecture 라는 struct 에 변수를 선언하였고 typedef 를 이용해 lecture 을 이용할 수 있게 하였습니다.

main 에서 입력을 받아 저장하고 크기까지 출력을 해주었습니다.

포인터 배열은 위와 표와 같이 나타내 사용을 할 수 있습니다.

표 13-2 구조체 변수와 구조체 포인터 변수를 이용한 멤버의 참조

접근 연산식	구조체 변수 os와 구조체 포인터변수 p인 경우의 의미
p->name	포인터 p가 가리키는 구조체의 멤버 name
(*p).name	포인터 p가 가리키는 구조체의 멤버 name
*p.name	*(p.name)이고 p가 포인터이므로 p.name은 문법오류가 발생
*os.name	*(os.name)를 의미하며, 구조체 변수os의 멤버 포인터 name이 가리키는 변수로, 이 경우는 구조체 변수 os 멤버 강좌명의 첫 문자임, 다만 한글인 경우에는 실행 오류
*p->name	*(p->name)을 의미하며, 포인터 p이 가리키는 구조체의 멤버 name이 가리키는 변수로 이 경우는 구조체 포인터 p이 가리키는 구조체의 멤버 강좌명의 첫 문자임, 마찬가지로 한글인 경우에는 실행 오류

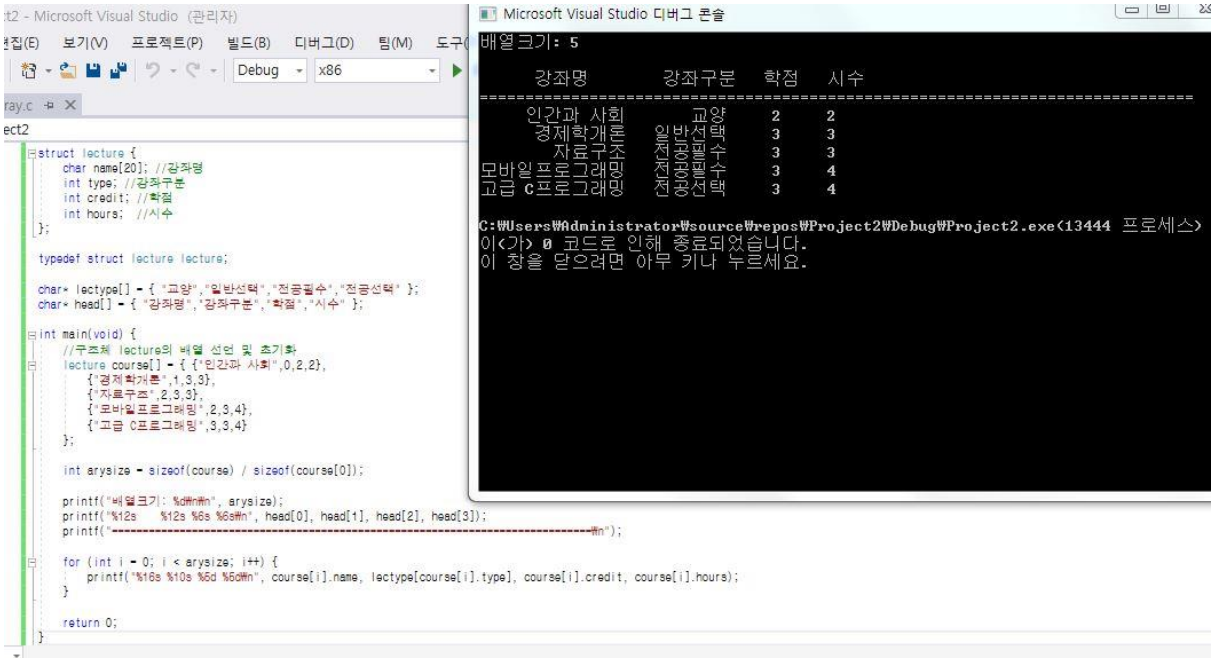
실습 예제 13-8

```
ct2
//file: unionpointer.c
#include <stdio.h>

int main(void) {
    //공용체 union data 정의
    union data
    {
        char ch;
        int cnt;
        double real;
    };

    //유니온 union data를 다시 자료형 udata로 정의
    typedef union data udata;

    //udata형으로 value 와 포인터 p 선언
    udata value, *p;

    p = &value;
    p->ch = 'a';
    printf("%c %c \n", p->ch, (*p).ch);
    p->cnt = 100;
    printf("%d", p->cnt);
    p->real = 3.14;
    printf("%.2f \n", p->real);

    return 0;
}
```

```
a a
1003.14

C:\Users\Administrator\source\repos\Project2\Debug\Project2.exe(11616 프로세스)
이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

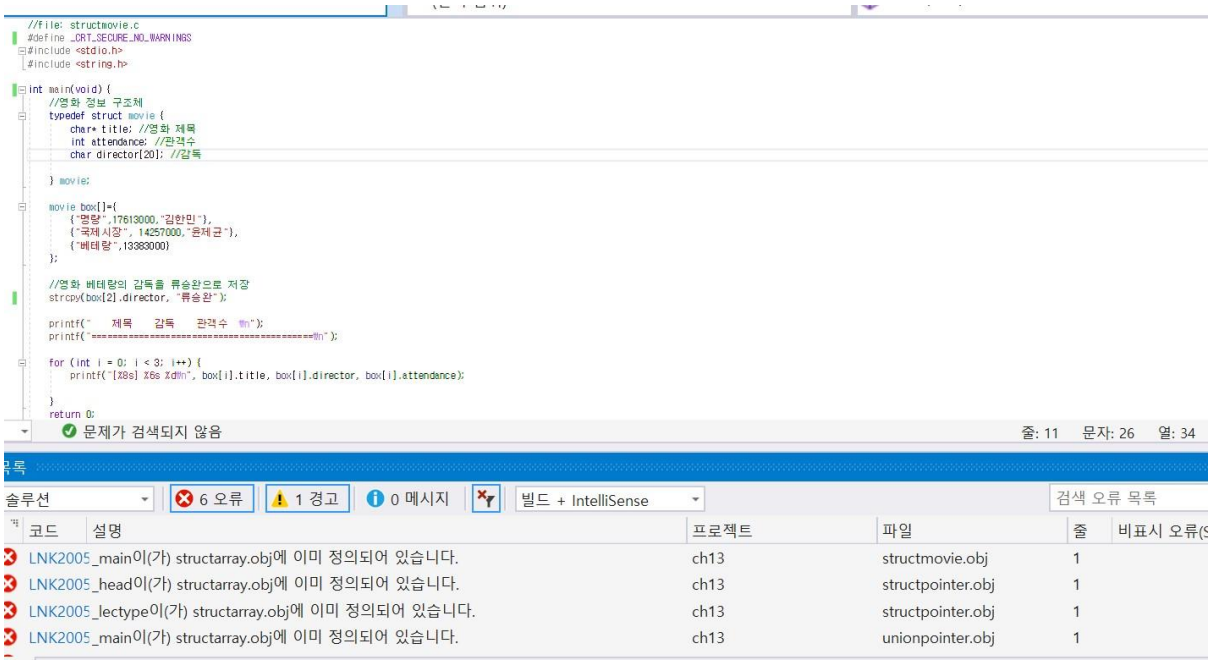
공용체 변수또한 포인터 변수 사용이 가능하며 접근 연산자 '->'를 이용하여 위에 보시는 코드와 같이 나타내어 출력하신 것을 확인하실 수 있습니다.

실습 예제 13-9



```
배열크기: 5

      강좌명        강좌구분   학점   시수
================================================================================
      인간과 사회        교양     2     2
      경제학개론      일반선택     3     3
        자료구조      전공필수     3     3
모바일프로그래밍      전공필수     3     4
고급 C프로그래밍      전공선택     3     4

C:\Users\Administrator\source\repos\Project2\Debug\Project2.exe(13444 프로세스)
이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

LAB 13-3



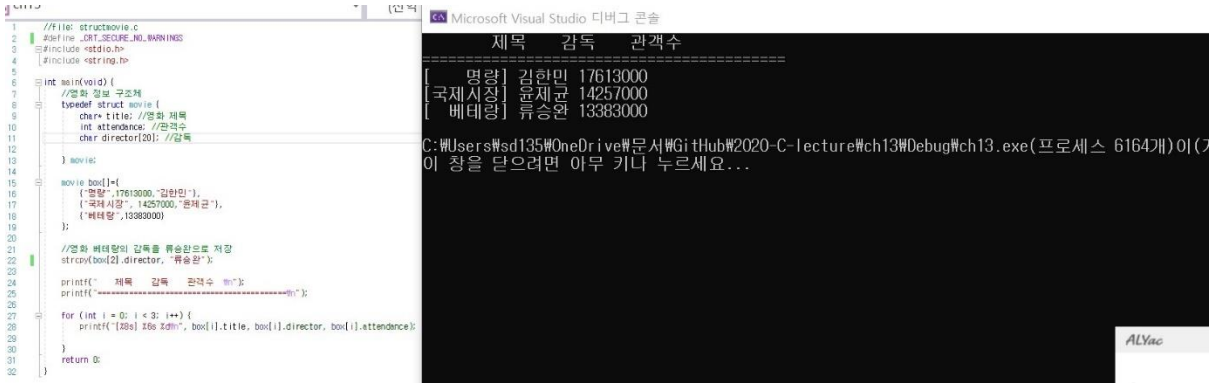
```
//File: structmovie.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void) {
    //영화 정보 구조체
    typedef struct movie {
        char* title; //영화 제목
        int attendance; //관객수
        char director[20]; //감독

    } movie;

    movie box[]={
        {"명량",17613000,"김한민"},
        {"국제시장", 14257000,"윤제균"},
        {"베테랑",13383000}
    };

    //영화 베테랑의 감독을 류승완으로 저장
    strcpy(box[2].director, "류승완");

    printf("   제목    감독    관객수 \n");
    printf("=========================================\n");

    for (int i = 0; i < 3; i++) {
        printf("[%8s] %6s %d\n", box[i].title, box[i].director, box[i].attendance);

    }
    return 0;
```

문제가 검색되지 않음 줄: 11 문자: 26 열: 34

6 오류 1 경고 0 메시지 빌드 + IntelliSense 검색 오류 목록

코드	설명	프로젝트	파일	줄	비표시 오류(S
LNK2005	_main이(가) structarray.obj에 이미 정의되어 있습니다.	ch13	structmovie.obj	1	
LNK2005	_head이(가) structarray.obj에 이미 정의되어 있습니다.	ch13	structpointer.obj	1	
LNK2005	_lectype이(가) structarray.obj에 이미 정의되어 있습니다.	ch13	structpointer.obj	1	
LNK2005	_main이(가) structarray.obj에 이미 정의되어 있습니다.	ch13	unionpointer.obj	1	

이거의 오류 또한 chap 11 에서 났던 오류 LNK2005 가 발생해 다른 파일에서 main 을 사용하지 않게 하니 다시 출력이 잘되는 것을 확인하였습니다.



Typedef struct 를 이용해 movie 로 선언하여 안에 변수를 넣어주었습니다.

구조체 movie 의 배열을 box 로 선언하면서 값을 초기화를 해주고 for 문으로 출력을 하였습니다.

포트폴리오를 만들며

위 포트폴리오를 작업을 하면서 처음으로 작성을 하였지만 온라인 강의를 들으면서 제가 잘 몰랐던 부분에 대해 개념정리를 하는 시간이었던 것 같습니다.

아직 지금까지 정리한 부분이외에도 시간이 남게 된다면 제가 배웠던 언어를 정리해 보는 시간을 가지는 것도 앞으로 이 분야에 취업을 한다는 것을 가정하면 꼭 필요하다는 것을 많이 느끼는 의미 있는 시간이 였던 것 같습니다.

비록 개인적으로 정리를 하였기에 분명 잘못 적은 것과 내용 설명이 미약하다는 단점이 있지만 보시면서 제가 오류가 있으면 지적해 주시면 감사하겠습니다.

프로그래밍을 하며 오류가 나는 것이 두려웠었지만 몇일 인터넷에 검색하고 고민하는 과정이 그 때 당시에는 시간 낭비라고 생각했지만 지금 와서 생각하면 의미 있는 시간이었던 것 같습니다.

우선 제가 대표적으로 오류 났던 것은 LNK 2005 와 LNK 2019 였는데 이것은 의외로 함수의 중복과 main 을 안사용해서 났던 오류임으로 앞으로 코드에 오류가 나면 무조건 제 자신을 의심하고 코드를 다시 살펴보는 작업이 중요하다는 것을 느꼈습니다.

하루 빨리 코로나가 종식이 되어 다음학기에는 반드시 대면 수업을 하며 교수님들께 질문이 있으면 바로 해결할 수 있게 되면 좋겠다는 생각 듭니다.