

PORT POLIO

C 애플리케이션구현

E 반

학번 : 20170666

이름 : 박승찬

강의계획서

2020 학년도 1 학기	전공	컴퓨터정보공학과(사회맞춤형 지능형 컴퓨팅과정)	학부	컴퓨터공학부
과 목 명	C 애플리케이션구현(2016003-PE)			

강의실 과 강의시간	월:5(3-217),6(3-217),7(3-217),8(3-217)	학점	3
교과분류	이론/실습	시수	4

담당 교수	강환수 + 연구실 : 2 호관-706 + 전 화 : 02-2610-1941 + E-MAIL : hskang@dongyang.ac.kr + 면담가능기간 : 월 11 시~12 시 화 14 시~17 시
-------	---

학과 교육목표	
---------	--

과목 개요	본 과목은 프로그래밍 언어 중 가장 널리 사용되고 있는 C 언어를 학습하는 과목으로 C++, JAVA 등과 같은 언어의 기반이 된다. 본 과목에서는 지난 학기에서 배운 시스템프로그래밍 1 에 이어 C 언어의 기본 구조 및 문법 체계 그리고 응용 프로그래밍 기법 등을 다룬다. C 언어에 대한 학습은 Windows 상에서 이루어지며, 기본적인 이론 설명 후 실습문제를 프로그래밍하며 숙지하는 형태로 수업이 진행된다.
-------	--

학습목표 및 성취수준	대학 교육목표와 학과 교육목표를 달성하기 위하여 이 과목을 수강함으로써 학습자는 C 언어의 문법 전 반과 응용 프로그램 기법을 알 수 있다. 직전 학기의 수강으로 인한 C 언어의 기초부터 함수, 포인터 등의 내용 이해를 바탕으로하여 이번 학기에는 지난 학기 내용의 전체적인 복습과 함께 C 언어 전체를 학습하고, 특히 응용 능력을 배양하여 프로그램으로 문제를 해결하는 능력을 익히게 된다.
-------------	--

	도서명	저자	출판사	비고
--	-----	----	-----	----

주교재	Perfect C	강환수, 강환일, 이동규	인피니티북스	
-----	-----------	---------------	--------	--

수업시 사용도구	Visual C++
----------	------------

평가방법	중간고사 30%, 기말고사 30%, 과제물 및 퀴즈 20%, 출석 20%
------	--

수강안내	C 언어를 활용하여 응용프로그램을 구현할 수 있다.
------	------------------------------

1 주차	[개강일(3/16)]
------	-------------

학습주제	강의 소개 및 전 학기 강의 내용 복습: C 언어 기초 및 조건문과 반복문 복습
목표및 내용	C 언어 기초 통합개발환경 테스트 기초적인 코드 실습

미리읽어오기	교재 1~5 장
과제,시험,기타	수업 중에 제시함
2 주차	[2 주]
학습주제	C 언어 기초 문법
목표및 내용	변수와 상수 연산자 l-value 와 r-value
미리읽어오기	교재 1~5 장
과제,시험,기타	수업 중에 제시함
3 주차	[3 주]
학습주제	조건문
목표및 내용	6 장 조건문 학습
미리읽어오기	교재 6 장
과제,시험,기타	수업 중에 제시함
4 주차	[4 주]
학습주제	반복문
목표및 내용	7 장 반복문 학습
미리읽어오기	교재 7 장
과제,시험,기타	수업 중에 제시함
5 주차	[5 주]
학습주제	포인터
목표및 내용	8 장 포인터 학습 단일포인터 다중포인터 여러가지 포인터
미리읽어오기	교재 8 장

과제,시험,기타	수업 중에 제시함
6 주차	[6 주]
학습주제	배열
목표및 내용	9 장 배열
미리읽어오기	교재 9 장
과제,시험,기타	수업 중에 제시함

7 주차	[7 주]
학습주제	함수
목표및 내용	10 장 함수
미리읽어오기	교재 10 장
과제,시험,기타	수업 중에 제시함
8 주차	[중간고사]
학습주제	중간고사
목표및 내용	중간고사
미리읽어오기	
과제,시험,기타	
9 주차	[9 주]
학습주제	문자열
목표및 내용	11 장 문자열
미리읽어오기	교재 11 장
과제,시험,기타	수업 중에 제시함
10 주차	[10 주]
학습주제	변수 유효범위
목표및 내용	12 장 변수 유효범위
미리읽어오기	교재 12 장
과제,시험,기타	수업 중에 제시함
11 주차	[11 주]
학습주제	구조체
목표및 내용	13 장 구조체

미리읽어오기	교재 13 장
과제,시험,기타	수업 중에 제시함
12 주차	[12 주]
학습주제	함수와 포인터 활용
목표및 내용	14 장 함수와 포인터활용
미리읽어오기	교재 14 장
과제,시험,기타	수업 중에 제시함

13 주차	[13 주]
학습주제	파일처리
목표및 내용	15 장 파일처리
미리읽어오기	교재 15 장
과제,시험,기타	수업 중에 제시함
14 주차	[14 주]
학습주제	항상심화강좌(동적할당)
목표및 내용	16 장 동적할당
미리읽어오기	교재 16 장
과제,시험,기타	수업 중에 제시함
15 주차	[기말고사]
학습주제	기말고사
목표및 내용	기말고사
미리읽어오기	
과제,시험,기타	..
수업지원 안내	장애학생을 위한 별도의 수강 지원을 받을 수 있습니다. 언어가 문제가 되는 학생은 글로 된 과제 안내, 확대문자 시험지 제공 등의 지원을 드립니다.

목차

1. 문자와 문자열

- 문자와 문자열
- 문자열 관련 함수
- 여러 문자열 처리

2. 변수 유효범위

- 전역변수와 지역변수
- 정적 변수와 레지스터 변수
- 메모리 영역과 변수 이용

3. 구조체와 공용체

- 구조체와 공용체
- 자료형 재정의
- 구조체와 공용체의 포인터와 배열

11-1 문자와 문자열

문자와 문자열의 개념

- 문자는 영어의 알파벳이나 한글의 한 글자를 작은 따옴표로 둘러싸서 'A'와 같이 표기
- 작은 따옴표에 의해 표기된 문자는 문자 상수
- 문자의 모임인 일련의 문자는 문자열
- 문자의 나열인 문자열은 'ABC'처럼 작은 따옴표로 둘러싸도 문자가 될 수 없으며 오류가 발생

문자와 문자열의 선언

- Char형 변수에 문자를 저장
- 문자열을 저장하려면 문자의 모임인 '문자 배열'을 사용
- 문자열의 마지막을 의미하는 NULL 문자 '\0'가 마지막에 저장되어야 하므로 문자열이 저장되는 배열크기는 반드시 저장될 문자 수보다 1이 커야 함
- 배열 초기화 시 배열크기는 지정하지 않는 것이 더 편리하며, 만일 지정한다면 마지막 문자인 '\0'을 고려해 실제 문자 수보다 1이 더 크게 배열크기를 지정해야 함

문자와 문자열 출력 (실습 예제 11-1)

-문자열 저장을 위한 문자열 배열 처리와 문자열 출력 (실습 예제 11-1)

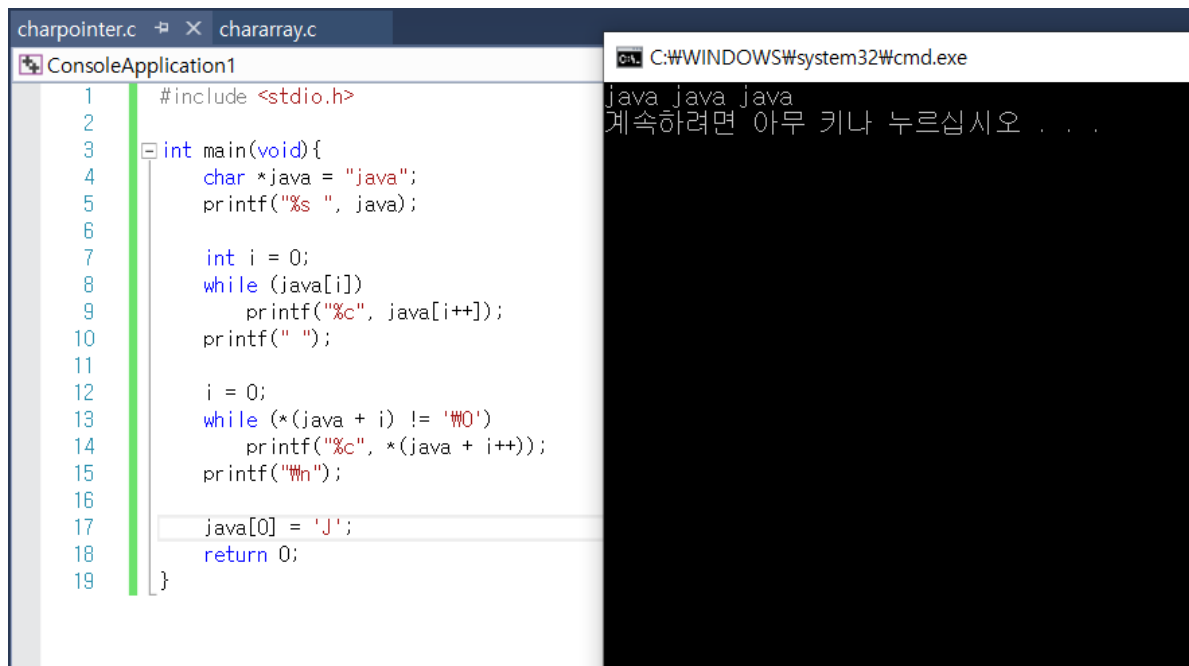
```
chararray.c  X
ch11
1  #include <stdio.h>
2
3  int main(void)
4  {
5      char ch = 'A';
6      printf("%c %d\n", ch, ch);
7
8      char java[] = { 'J', 'A', 'V', 'A', '\0' };
9      printf("%s\n", java);
10
11     char c[] = "C lanuage";
12     printf("%s\n", c);
13
14     char csharp[5] = "C#";
15     printf("%s\n", csharp);
16
17     printf("%c%c\n", csharp[0], csharp[1]);
18
19     return 0;
20 }
```

```
C:\WINDOWS\system32\cmd.exe
A 65
JAVA
C lanuage
C#
C#
계속하려면 아무 키나 누르십시오 . . .
```

문자열 구성하는 문자 참조

- 문자열 상수를 문자 포인터에 저장하는 방식
- 형식제어문자 %s
- 문자 포인터에 의한 선언으로는 문자 하나 하나의 수정은 할 수 없음

-문자 포인터로 문자열 처리 (실습 예제 11-2)



The screenshot shows a C program in a text editor and its execution in a command prompt. The C program, named charpointer.c, defines a main function that prints the string "java" using printf("%s", java);. It then iterates through the string character by character using while (java[i]) and prints each character with printf("%c", java[i++]);. After a space, it iterates through the string again using while (*(java + i) != '\0') and prints each character with printf("%c", *(java + i++));. Finally, it prints a newline with printf("\n");. The console output shows the string "java" printed twice, once as a whole and once character by character, followed by a newline.

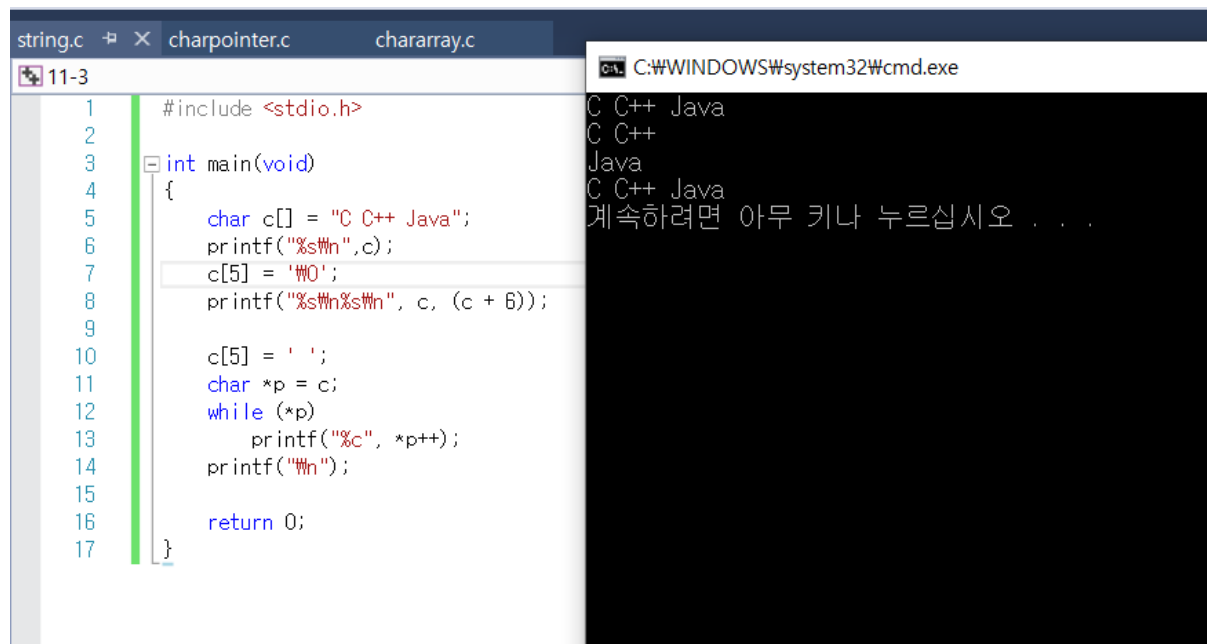
```
charpointer.c  X chararray.c
ConsoleApplication1
1  #include <stdio.h>
2
3  int main(void){
4      char *java = "java";
5      printf("%s ", java);
6
7      int i = 0;
8      while (java[i])
9          printf("%c", java[i++]);
10     printf(" ");
11
12     i = 0;
13     while (*(java + i) != '\0')
14         printf("%c", *(java + i++));
15     printf("\n");
16
17     java[0] = 'J';
18     return 0;
19 }
```

```
C:\WINDOWS\system32\cmd.exe
java java java
계속하려면 아무 키나 누르십시오 . . .
```

'\0'문자에 의한 문자열 분리

- 함수 printf()에서 %s는 문자 포인터가 가리키는 위치에서 NULL 문자 까지를 하나의 문자열로 인식함

-문자 포인터로 문자열 처리 (실습 예제 11-3)



The screenshot shows a C program in a text editor and its execution in a command prompt. The program, named `charpointer.c`, includes `<stdio.h>` and defines a `main` function. It declares a character array `c` with the value "C C++ Java". It prints the string using `printf("%s\n", c);`. Then, it sets `c[5]` to a null terminator `'\0'` and prints the string again using `printf("%s\n%s\n", c, (c + 6));`. Next, it iterates through the string using a pointer `p` and prints each character using `printf("%c", *p++);`. Finally, it returns 0.

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     char c[] = "C C++ Java";
6     printf("%s\n", c);
7     c[5] = '\0';
8     printf("%s\n%s\n", c, (c + 6));
9
10    c[5] = ' ';
11    char *p = c;
12    while (*p)
13        printf("%c", *p++);
14    printf("\n");
15
16    return 0;
17 }
```

The command prompt output shows the following text:

```
C C++ Java
C C++
Java
C C++ Java
계속하려면 아무 키나 누르십시오 . . .
```

버퍼처리 함수 `getchar()`

- 함수 `getchar()`는 문자의 입력에 사용
- 함수 `putchar()`는 문자의 출력에 사용
- 문자 입력을 위한 함수 `getchar()`는 라인 버퍼링 방식을 사용
- 함수를 이용하려면 헤더파일 `conio.h` 삽입

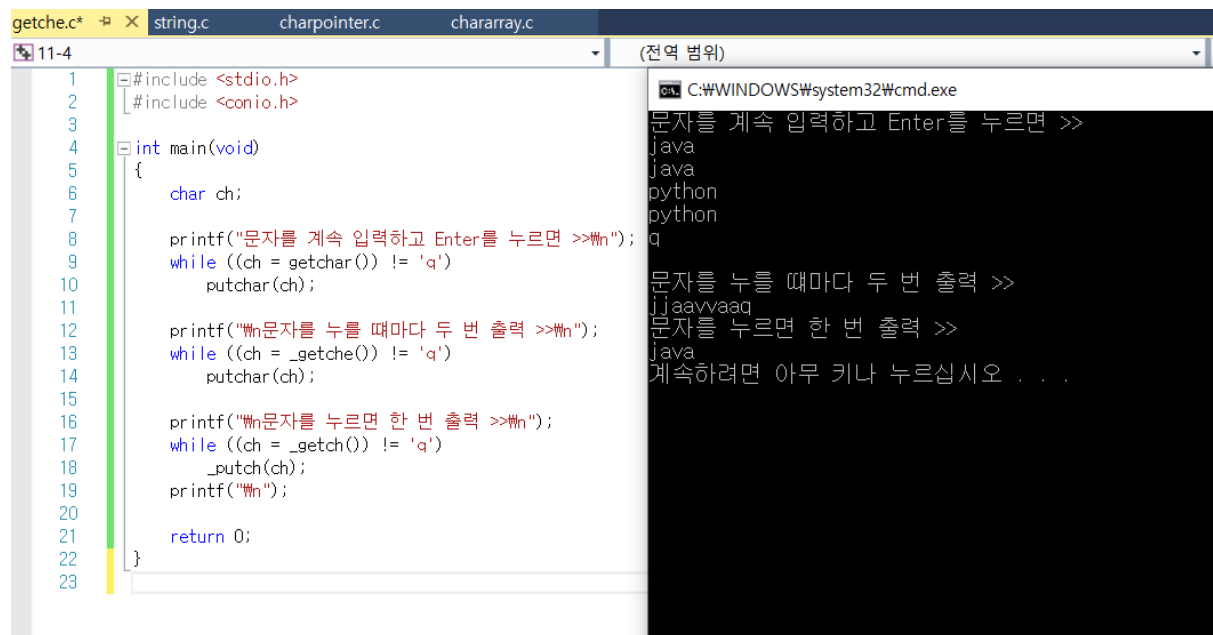
함수 `getche()`

- 버퍼를 사용하지 않고 문자를 입력하는 함수
- 함수를 이용하려면 헤더파일 `conio.h` 삽입

함수 `getch()`

- 입력한 문자가 화면에 보이지 않음
- 함수를 이용하려면 헤더파일 `conio.h` 삽입

-함수 getchar(), _getche(), _getch()의 차이를 알아보는 예제 (실습 예제 11-4)



```
1 #include <stdio.h>
2 #include <conio.h>
3
4 int main(void)
5 {
6     char ch;
7
8     printf("문자를 계속 입력하고 Enter를 누르면 >>\n");
9     while ((ch = getchar()) != 'q')
10         putchar(ch);
11
12     printf("\n문자를 누를 때마다 두 번 출력 >>\n");
13     while ((ch = _getche()) != 'q')
14         putchar(ch);
15
16     printf("\n문자를 누르면 한 번 출력 >>\n");
17     while ((ch = _getch()) != 'q')
18         _putch(ch);
19     printf("\n");
20
21     return 0;
22 }
```

Output (C:\WINDOWS\system32\cmd.exe):

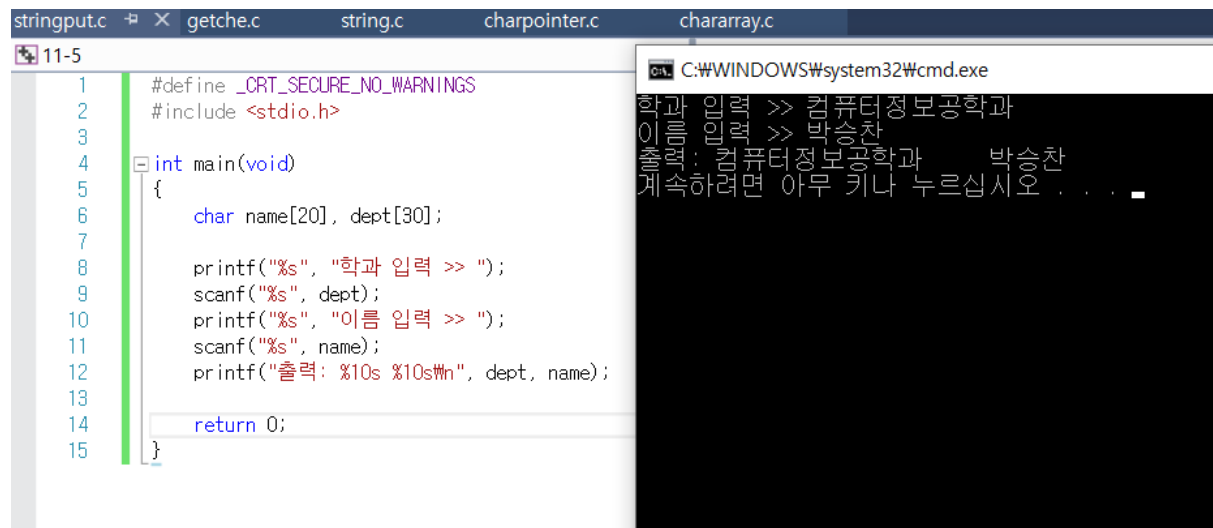
```
문자를 계속 입력하고 Enter를 누르면 >>
java
python
python
q

문자를 누를 때마다 두 번 출력 >>
jjaaavvaag
문자를 누르면 한 번 출력 >>
java
계속하려면 아무 키나 누르십시오 . . .
```

문자배열 변수로 scanf()에서 입력

- 함수 scanf()는 공백으로 구분되는 하나의 문자열을 입력 받을 수 있음

-함수 scanf()와 printf()에서 문자열 입출력 (실습 예제 11-5)



```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3
4 int main(void)
5 {
6     char name[20], dept[30];
7
8     printf("%s", "학과 입력 >> ");
9     scanf("%s", dept);
10    printf("%s", "이름 입력 >> ");
11    scanf("%s", name);
12    printf("출력: %10s %10s\n", dept, name);
13
14    return 0;
15 }
```

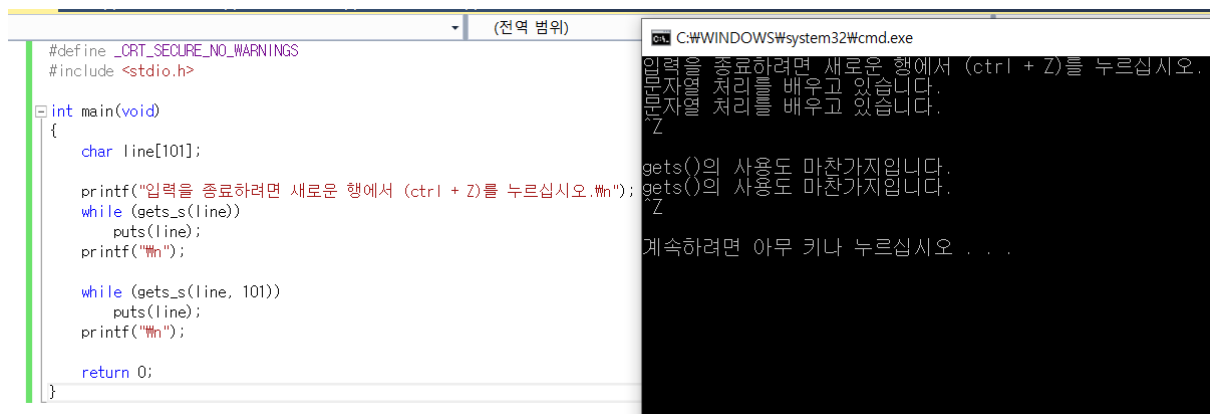
Output (C:\WINDOWS\system32\cmd.exe):

```
학과 입력 >> 컴퓨터정보공학과
이름 입력 >> 박승찬
출력: 컴퓨터정보공학과 박승찬
계속하려면 아무 키나 누르십시오 . . .
```

gets()와 puts()

- gets()는 한 행의 문자열 입력에 유용한 함수
- puts()는 한 행에 문자열을 출력하는 함수

-함수 gets()와 put() 기능을 알아보는 예제 (실습 예제 11-6)



The screenshot shows a C program in a text editor and its execution in a command prompt. The program defines a character array 'line' of size 101 and uses 'gets_s' to read input into it, then 'puts' to print it. It also demonstrates reading a line with 'gets_s' and printing it with 'puts'. The command prompt shows the program being run, with input lines and the corresponding output lines.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void)
{
    char line[101];

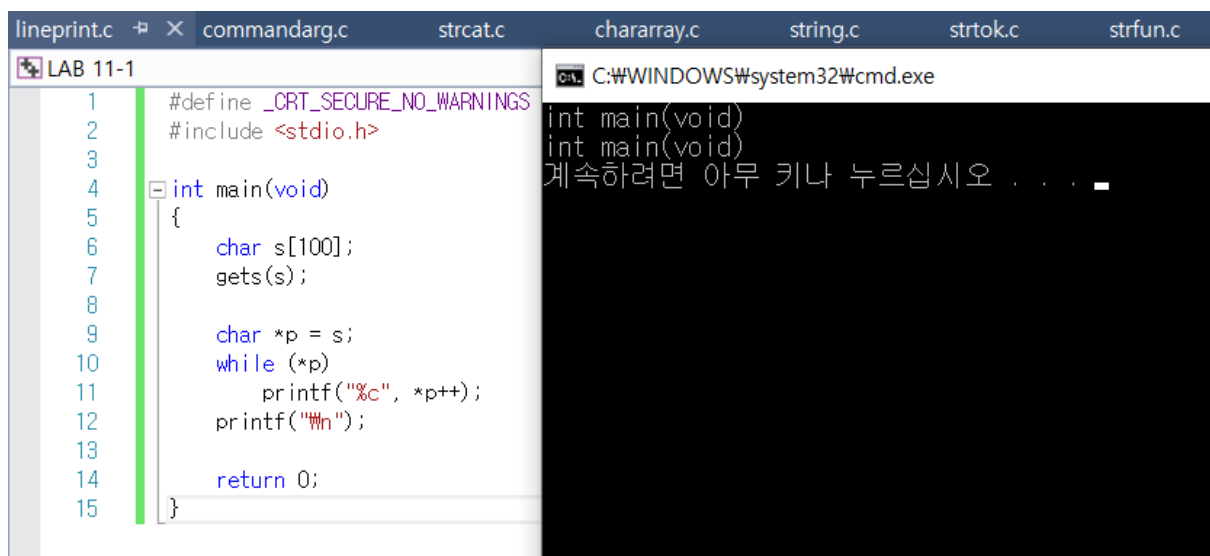
    printf("입력을 종료하려면 새로운 행에서 (ctrl + Z)를 누르십시오.\n");
    while (gets_s(line))
    {
        puts(line);
        printf("\n");
    }

    while (gets_s(line, 101))
    {
        puts(line);
        printf("\n");
    }

    return 0;
}
```

C:\WINDOWS\system32\cmd.exe
입력을 종료하려면 새로운 행에서 (ctrl + Z)를 누르십시오.
문자열 처리를 배우고 있습니다.
문자열 처리를 배우고 있습니다.
Z
gets()의 사용도 마찬가지입니다.
gets()의 사용도 마찬가지입니다.
Z
계속하려면 아무 키나 누르십시오 . . .

-한 행을 표준입력으로 받아 문자 하나 하나를 그대로 출력 (Lab 11-1)



The screenshot shows a C program in a text editor and its execution in a command prompt. The program reads a line of input into a character array 's' using 'gets', then iterates through each character and prints it individually. The command prompt shows the program being run, with input lines and the corresponding output lines.

```
lineprint.c  X  commandarg.c  strcat.c  chararray.c  string.c  strtok.c  strfun.c
LAB 11-1
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4  int main(void)
5  {
6      char s[100];
7      gets(s);
8
9      char *p = s;
10     while (*p)
11         printf("%c", *p++);
12     printf("\n");
13
14     return 0;
15 }
```

C:\WINDOWS\system32\cmd.exe
int main(void)
int main(void)
계속하려면 아무 키나 누르십시오 . . .

11-2 문자열 관련 함수

다양한 문자열 라이브러리 함수

- 문자열 비교와 복사, 그리고 문자열 연결 등과 같은 다양한 문자열 처리는 헤더파일 string.h 에 함수원형으로 선언된 라이브러리 함수로 제공

-문자배열에 관한 함수 (실습 예제 11-7)

```

1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6      char src[50] = "https://www.visualstudio.com";
7      char dst[50];
8
9      printf("문자배열 src = %s\n", src);
10     printf("문자열 크기 strlen(src) = %d\n", strlen(src));
11     memcpy(dst, src, strlen(src) + 1);
12     printf("문자배열 dst = %s\n", dst);
13     memcpy(src, "안녕하세요!", strlen("안녕하세요!") + 1);
14     printf("문자배열 src = %s\n", src);
15
16     char ch = '안';
17     char *ret;
18     ret = strchr(dst, ch, strlen(dst));
19     printf("문자 %c 뒤에는 문자열 %s 이 있다.\n", ch, ret);
20
21     return 0;
22 }

```

함수 strcmp()

- Strcmp()는 인자인 두 문자열을 사전상의 순서로 비교하는 함수
- Strncmp()는 두 문자를 비교할 문자의 최대 수를 지정하는 함수

-문자열 비교함수 strcmp()와 strncmp() 사용 (실습 예제 11-8)

```

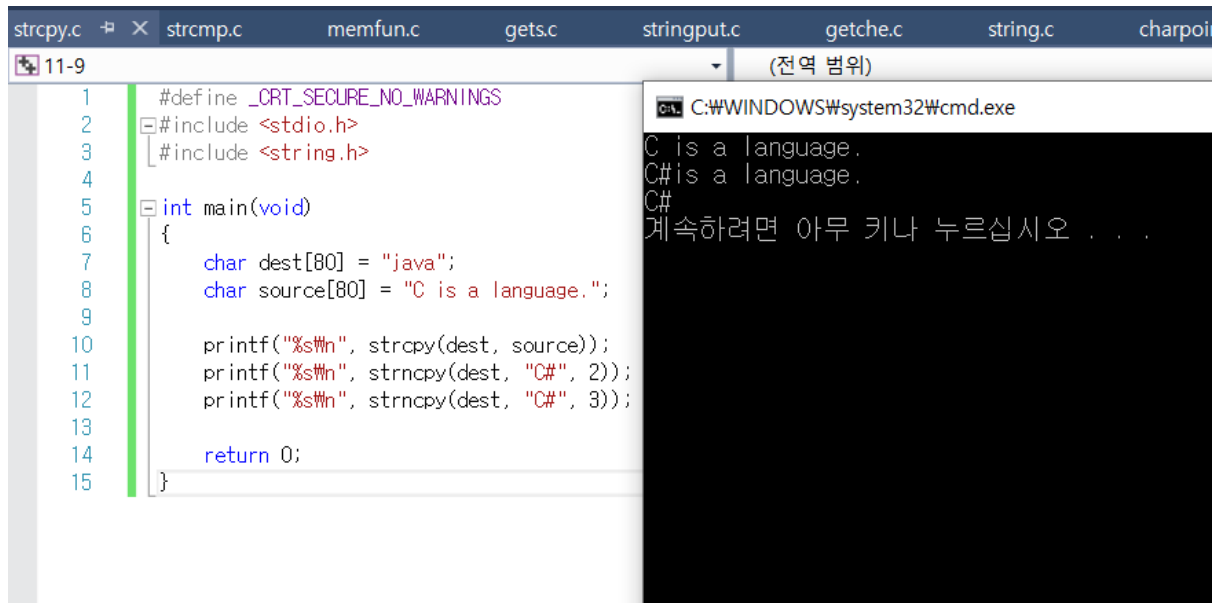
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6      char *s1 = "java";
7      char *s2 = "java";
8      printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
9
10     s1 = "java";
11     s2 = "jav";
12     printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
13     s1 = "jav";
14     s2 = "java";
15     printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
16     printf("strncmp(%s, %s, %d) = %d\n", s1, s2, 3, strncmp(s1, s2, 3));
17
18     return 0;
19 }

```

함수 strcpy()

- 함수 strcpy와 strncpy()는 문자열을 복사하는 함수
- 함수 strcpy()는 앞 인자 문자열 dest에 뒤 인자 문자열 source를 복사

-문자열 복사 함수 strcpy()와 strncpy() 사용 (실습 예제 11-9)



The screenshot shows a C program in a code editor with the following code:

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3 #include <string.h>
4
5 int main(void)
6 {
7     char dest[80] = "java";
8     char source[80] = "C is a language.";
9
10    printf("%s\n", strcpy(dest, source));
11    printf("%s\n", strncpy(dest, "C#", 2));
12    printf("%s\n", strncpy(dest, "C#", 3));
13
14    return 0;
15 }
```

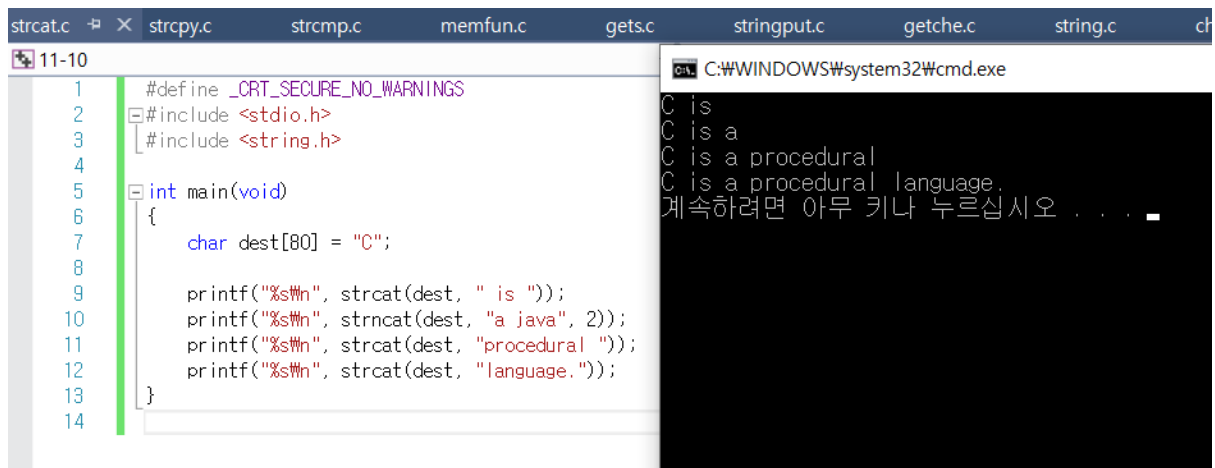
The output window shows the following text:

```
C:\WINDOWS\system32\cmd.exe
C is a language.
C#is a language.
C#
계속하려면 아무 키나 누르십시오 . . .
```

함수 strcat()

- 함수 strcat()는 앞 문자열에 뒤 문자열의 null 문자까지 연결하여, 앞의 문자열 주소를 반환하는 함수

-문자열 연결 함수 사용 (실습 예제 11-10)



The screenshot shows a C program in a code editor with the following code:

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3 #include <string.h>
4
5 int main(void)
6 {
7     char dest[80] = "C";
8
9     printf("%s\n", strcat(dest, " is "));
10    printf("%s\n", strcat(dest, "a java", 2));
11    printf("%s\n", strcat(dest, "procedural "));
12    printf("%s\n", strcat(dest, "language."));
13
14 }
```

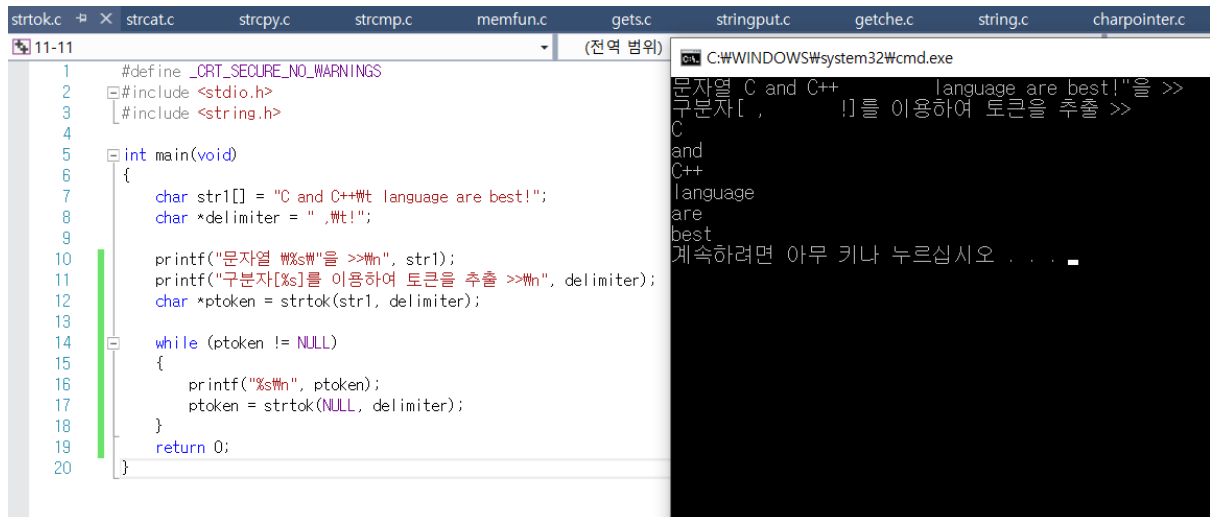
The output window shows the following text:

```
C:\WINDOWS\system32\cmd.exe
C is
C is a
C is a procedural
C is a procedural language.
계속하려면 아무 키나 누르십시오 . . .
```

함수 strtok()

- 함수 strtok()은 문자열에서 구분자인 문자를 여러 개 지정하여 토큰을 추출하는 함수
- 문장 ptoken = strtok(str, delimiter);으로 첫 토큰을 추출
- 결과를 저장한 ptoken이 NULL이면 더 이상 분리할 토큰이 없는 경우이다

-문자열에서 지정한 분리자를 사용하여 토큰을 사용 (실습 예제 11-11)



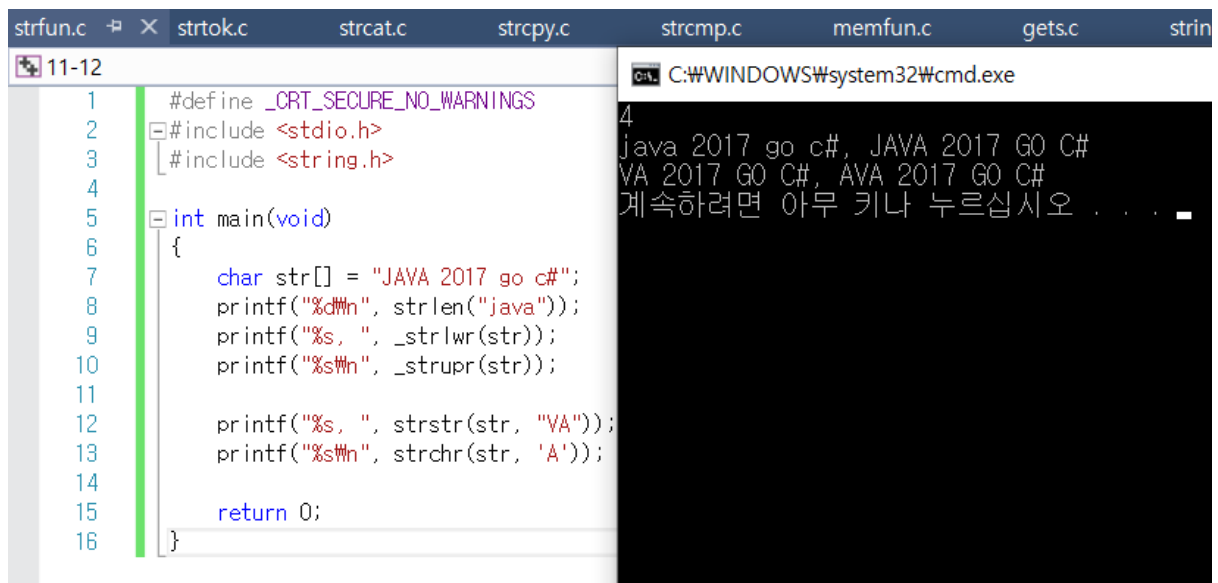
The screenshot shows a C program in a code editor and its execution in a command prompt. The code defines a string "C and C++ language are best!" and a delimiter ".\t!". It uses strtok() to extract tokens and prints them. The command prompt shows the output: "문자열 C and C++ language are best!" and "구분자[, !]를 이용하여 토큰을 추출 >>".

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3 #include <string.h>
4
5 int main(void)
6 {
7     char str1[] = "C and C++ language are best!";
8     char *delimiter = ".\t!";
9
10    printf("문자열 %s\n", str1);
11    printf("구분자[%s]를 이용하여 토큰을 추출 >>\n", delimiter);
12    char *ptoken = strtok(str1, delimiter);
13
14    while (ptoken != NULL)
15    {
16        printf("%s\n", ptoken);
17        ptoken = strtok(NULL, delimiter);
18    }
19    return 0;
20 }
```

문자열의 길이와 위치 검색

- 함수 strlen()은 NULL 문자를 제외한 문자열 길이를 반환하는 함수

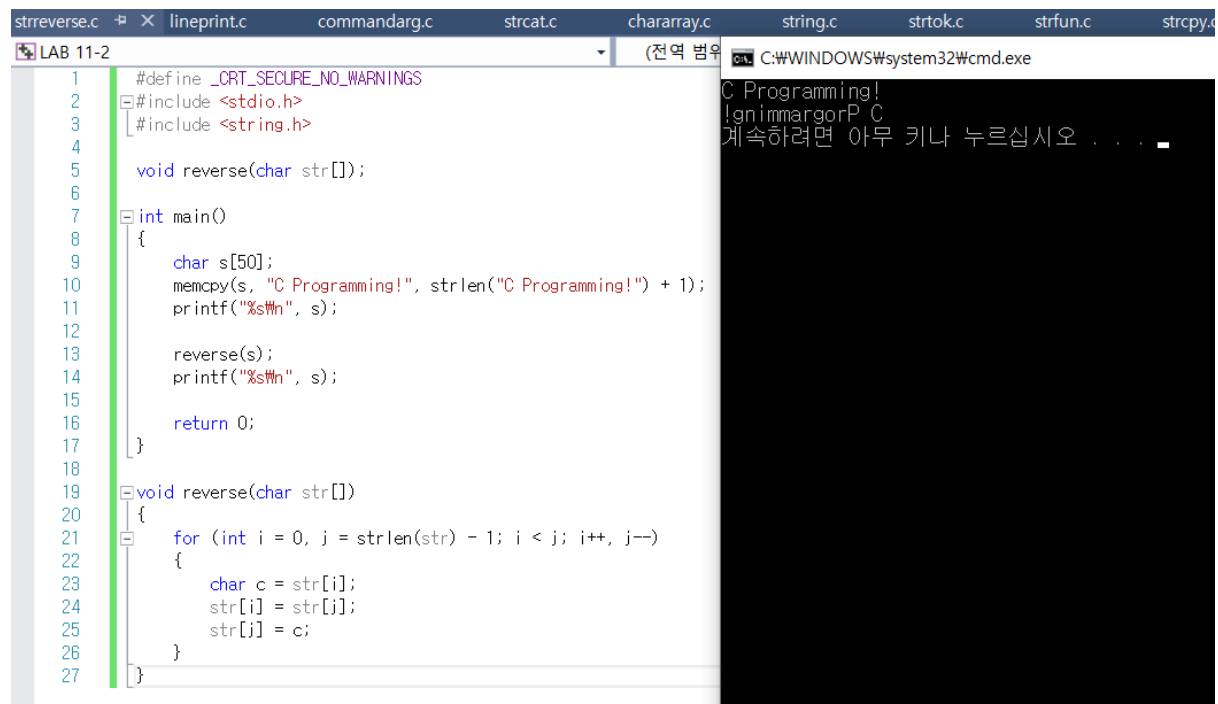
-다양한 문자열 관련 함수의 이해 (실습 예제 11-12)



The screenshot shows a C program in a code editor and its execution in a command prompt. The code uses strlen(), _strlwr(), _strupr(), strstr(), and strchr() to process the string "JAVA 2017 go c#". The command prompt shows the output: "4", "java 2017 go c#", "JAVA 2017 GO C#", "VA 2017 GO C#", "AVA 2017 GO C#", and "계속하려면 아무 키나 누르십시오 . . .".

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3 #include <string.h>
4
5 int main(void)
6 {
7     char str[] = "JAVA 2017 go c#";
8     printf("%d\n", strlen("java"));
9     printf("%s, ", _strlwr(str));
10    printf("%s\n", _strupr(str));
11
12    printf("%s, ", strstr(str, "VA"));
13    printf("%s\n", strchr(str, 'A'));
14
15    return 0;
16 }
```

-문자열을 역순으로 저장하는 함수 reverse() 구현 (Lab 11-2)



The screenshot shows a C program in a text editor with the following code:

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <string.h>
4
5  void reverse(char str[]);
6
7  int main()
8  {
9      char s[50];
10     memcpy(s, "C Programming!", strlen("C Programming!") + 1);
11     printf("%s\n", s);
12
13     reverse(s);
14     printf("%s\n", s);
15
16     return 0;
17 }
18
19 void reverse(char str[])
20 {
21     for (int i = 0, j = strlen(str) - 1; i < j; i++, j--)
22     {
23         char c = str[i];
24         str[i] = str[j];
25         str[j] = c;
26     }
27 }
```

The output window shows the execution results:

```
C:\WINDOWS\system32\cmd.exe
C Programming!
!gnimmargorP C
계속하려면 아무 키나 누르십시오 . . .
```

11-3 여러 문자열 처리

문자 포인터 배열

- 여러 개의 문자열을 처리하는 하나의 방법은 문자 포인터 배열을 이용하는 방법
- 각각의 문자열 저장을 위한 최적의 공간을 사용하는 것이 장점
- 문자열 상수의 수정은 불가능

이차원 문자 배열

- 여러 개의 문자열을 처리하는 문자의 이차원 배열

-여러 개의 문자열을 선언과 동시에 저장하고 처리하는 방법 (실습 예제 11-13)

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     char *pa[] = { "JAVA", "C#", "C++" };
6     char ca[][5] = { "JAVA", "C#", "C++" };
7
8     printf("%s ", pa[0]); printf("%s ", pa[1]); printf("%s\n", pa[2]);
9     printf("%s ", ca[0]); printf("%s ", ca[1]); printf("%s\n", ca[2]);
10
11     printf("%c %c %c\n", pa[0][1], pa[1][1], pa[2][1]);
12     printf("%c %c %c\n", ca[0][1], ca[1][1], ca[2][1]);
13
14     return 0;
15 }

```

Output: JAVA C# C++
 JAVA C# C++
 A # +
 A # +
 계속하려면 아무 키나 누르십시오 . . .

Main(int argc, char *argv[])

- 명령행에서 입력하는 문자열을 프로그램으로 전달하는 방법
- 실행 프로그램 이름도 하나의 명령행 인자에 포함

-명령행 인자 출력 (실습 예제 11-14)

```

1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     int i = 0;
6
7     printf("실행 명령행 인자(command line arguments) >>\n");
8     printf("argc = %d\n", argc);
9     for (i = 0; i < argc; i++)
10         printf("argv[%d] = %s\n", i, argv[i]);
11
12     return 0;
13 }

```

Output: 실행 명령행 인자(command line arguments) >>
 argc = 4
 argv[0] = D:\visual studio\ch11\Debug\11-14.exe
 argv[1] = C#
 argv[2] = C++
 argv[3] = Java
 계속하려면 아무 키나 누르십시오 . . .

-여러 문자열 처리 (Lab 11-3)

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     char str1[] = "JAVA";
6     char str2[] = "C#";
7     char str3[] = "C++";
8
9     char *pstr[] = { str1, str2, str3 };
10
11     printf("%s ", pstr[0]);
12     printf("%s ", pstr[1]);
13     printf("%s\n", pstr[2]);
14
15     printf("%c %c %c\n", str1[0], str2[1], str3[2]);
16     printf("%c %c %c\n", pstr[0][1], pstr[1][1], pstr[2][1]);
17 }

```

Output: JAVA C# C++
 J # +
 A # +
 계속하려면 아무 키나 누르십시오 . . .

12-1 전역변수와 지역변수

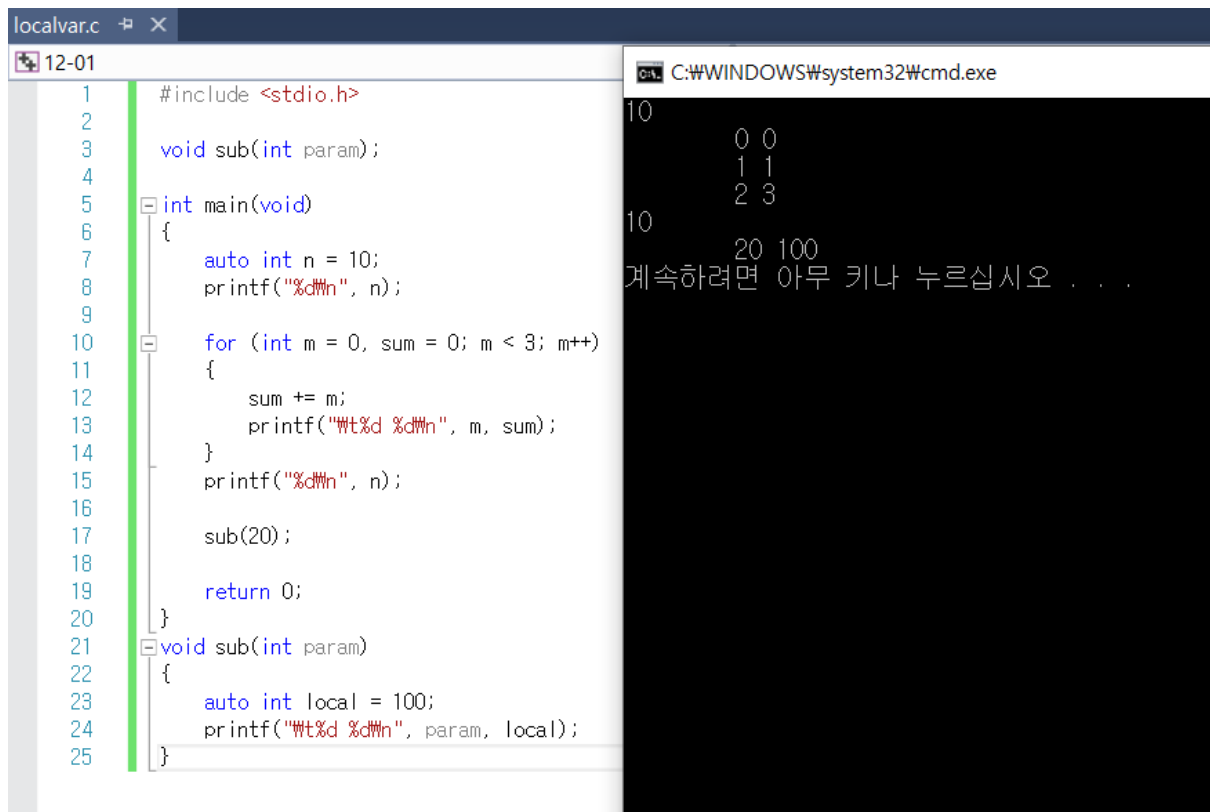
변수 scope

- 변수의 참조가 유효한 범위 : 변수의 유효 범위(scope)
- 변수의 유효 범위는 지역 유효 범위와 전역 유효 범위로 나뉘어 있음

지역변수

- 함수나 블록 내부에서 선언되어 사용하는 변수
- 함수의 매개변수도 함수 전체에서 사용 가능한 지역변수와 같다
- 지역변수는 선언 후 초기화하지 않으면 쓰레기 값이 저장됨
- 지역변수가 할당하는 메모리 영역은 스택
- 지역변수는 선언된 부분에서 자동으로 생성되고 함수나 블록이 종료되는 순간 메모리에서 자동으로 제거된다

-지역변수 선언과 사용 (실습 예제 12-1)



```
localvar.c 12-01
1  #include <stdio.h>
2
3  void sub(int param);
4
5  int main(void)
6  {
7      auto int n = 10;
8      printf("%d\n", n);
9
10     for (int m = 0, sum = 0; m < 3; m++)
11     {
12         sum += m;
13         printf("%t%d %d\n", m, sum);
14     }
15     printf("%d\n", n);
16
17     sub(20);
18
19     return 0;
20 }
21 void sub(int param)
22 {
23     auto int local = 100;
24     printf("%t%d %d\n", param, local);
25 }
```

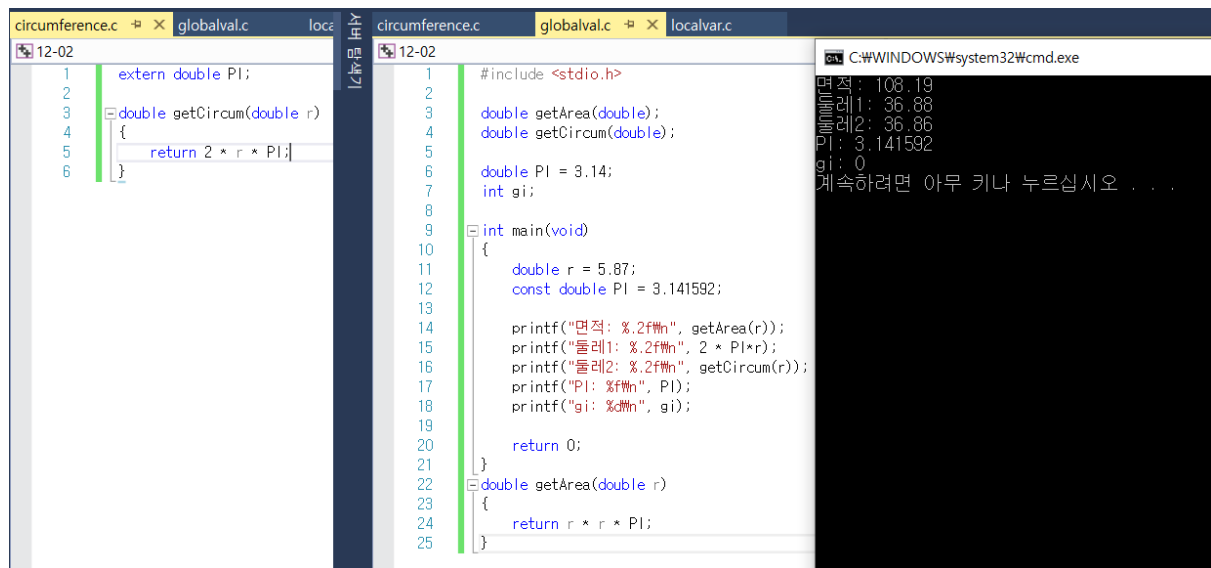
```
C:\WINDOWS\system32\cmd.exe
10
    0 0
    1 1
    2 3
10
    20 100
계속하려면 아무 키나 누르십시오 . . .
```

전역변수

- 함수 외부에서 선언되는 변수
- 외부변수라고도 불림
- 일반적으로 프로젝트의 모든 함수나 블록에서 참조 가능

-전역변수 PI 의 선언과 사용 (실습 예제 12-2)

-외부 전역변수 PI 를 참조 (실습 예제 12-3)



```
circumference.c
1 extern double PI;
2
3 double getCircum(double r)
4 {
5     return 2 * r * PI;
6 }

globalval.c
1 #include <stdio.h>
2
3 double getArea(double);
4 double getCircum(double);
5
6 double PI = 3.14;
7 int gi;
8
9 int main(void)
10 {
11     double r = 5.87;
12     const double PI = 3.141592;
13
14     printf("면적: %.2f\n", getArea(r));
15     printf("둘레1: %.2f\n", 2 * PI*r);
16     printf("둘레2: %.2f\n", getCircum(r));
17     printf("PI: %f\n", PI);
18     printf("gi: %d\n", gi);
19
20     return 0;
21 }
22 double getArea(double r)
23 {
24     return r * r * PI;
25 }

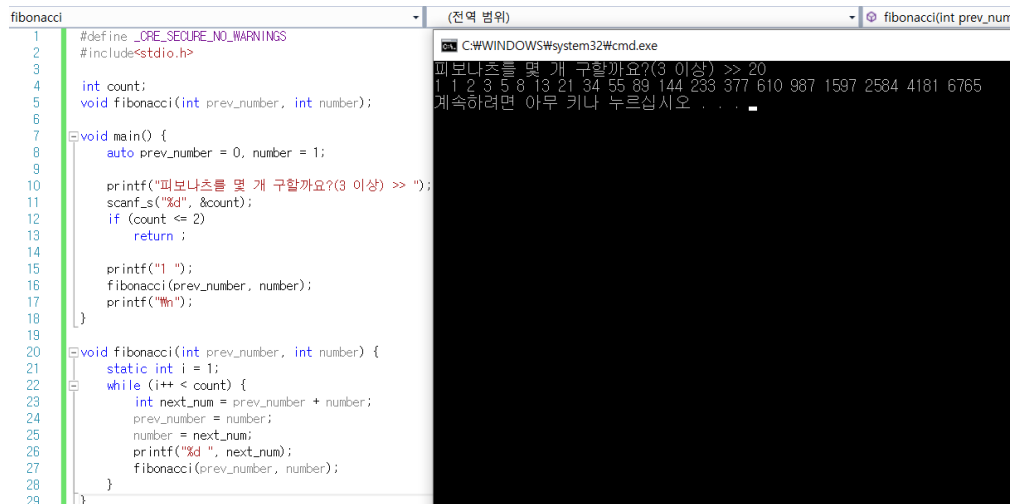
localvar.c
1 #include <stdio.h>
2
3 double getArea(double);
4 double getCircum(double);
5
6 double PI = 3.14;
7 int gi;
8
9 int main(void)
10 {
11     double r = 5.87;
12     const double PI = 3.141592;
13
14     printf("면적: %.2f\n", getArea(r));
15     printf("둘레1: %.2f\n", 2 * PI*r);
16     printf("둘레2: %.2f\n", getCircum(r));
17     printf("PI: %f\n", PI);
18     printf("gi: %d\n", gi);
19
20     return 0;
21 }
22 double getArea(double r)
23 {
24     return r * r * PI;
25 }
```

```
C:\WINDOWS\system32\cmd.exe
면적: 108.19
둘레1: 36.88
둘레2: 36.86
PI: 3.141592
gi: 0
계속하려면 아무 키나 누르십시오 . . .
```

전역변수 장단점

- 전역변수는 어디에서든지 수정 가능해 사용이 편함
- 예상하지 못한 값이 저장된다면 프로그램 어느 부분에서 수정되었는지 알기 어려움

-피보나치 수의 출력 (Lab 12-1)



```
fibonacci
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3
4 int count;
5 void fibonacci(int prev_number, int number);
6
7 void main() {
8     auto prev_number = 0, number = 1;
9
10    printf("피보나치를 몇 개 구할까요?(3 이상) >> ");
11    scanf_s("%d", &count);
12    if (count <= 2)
13        return;
14
15    printf("1 ");
16    fibonacci(prev_number, number);
17    printf("\n");
18 }
19
20 void fibonacci(int prev_number, int number) {
21     static int i = 1;
22     while (i++ < count) {
23         int next_num = prev_number + number;
24         prev_number = number;
25         number = next_num;
26         printf("%d ", next_num);
27         fibonacci(prev_number, number);
28     }
29 }
```

```
C:\WINDOWS\system32\cmd.exe
피보나치를 몇 개 구할까요?(3 이상) >> 20
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
계속하려면 아무 키나 누르십시오 . . .
```

12-2 정적 변수와 레지스터 변수

Auto, register, static, extern

- 4가지의 기억부류에 따라 할당되는 메모리 영역이 결정되고 메모리의 할당과 제거 시기가 결정된다
- 키워드 extern을 제외하고 나머지 3개의 기억부류의 변수선언에서 초기값을 저장할 수 있다

기억부류 종류	전역	지역
auto	X	O
register	X	O
static	O	O
extern	O	X

키워드 register

- 레지스터 변수는 변수의 저장공간이 일반 메모리가 아니라 CPU 내부의 레지스터에 할당되는 변수이다
- 레지스터 변수는 키워드 register를 자료형 앞에 넣어 선언한다
- 레지스터 변수는 일반 메모리에 할당되는 변수가 아니므로 주소연산자 &를 사용할 수 없다
- 레지스터 변수는 처리 속도를 증가시키려는 변수에 이용한다. 특히 반복문의 횟수를 제어하는 제어변수에 이용하면 효과적이다

-레지스터 변수의 선언과 사용 (실습 예제 12-4)

The screenshot shows a C program in a code editor with tabs for 'registerval.c', 'circumference.c', 'globalval.c', and 'localvar.c'. The active file 'registerval.c' contains the following code:

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3
4 int main(void)
5 {
6     register int sum = 0;
7
8     int max;
9     printf("양의 정수 입력 >> ");
10    scanf("%d", &max);
11
12    for (register int count = 1; count <= max; count++)
13        sum += count;
14
15    printf("합: %d\n", sum);
16
17    return 0;
18 }
```

Overlaid on the code editor is a terminal window titled 'C:\WINDOWS\system32\cmd.exe'. It shows the program's execution with the following input and output:

```
양의 정수 입력 >> 8
합: 36
계속하려면 아무 키나 누르십시오 . . .
```

키워드 static

- 변수 선언에서 자료형 앞에 키워드 static을 넣어 정적변수를 선언
- 정적변수는 초기값을 지정하지 않으면 자동으로 자료형에 따라 '0' 또는 NULL 값이 저장된다

정적 지역변수

- 함수나 블록에서 정적으로 선언되는 변수
- 정적 지역변수는 함수나 블록을 종료해도 메모리에서 제거되지 않고 계속 메모리에 유지 관리되는 특성이 있음

-정적 지역변수와 자동 지역변수의 차이 (실습 예제 12-5)

```

staticlocal.c  x  registerval.c  circumference.c  globalval.c  localvar.c
12-05
1  #include <stdio.h>
2
3  void increment(void);
4
5  int main(void)
6  {
7      for (int count = 0; count < 3; count++)
8          increment();
9  }
10 void increment(void)
11 {
12     static int sindex = 1;
13     auto int aindex = 1;
14
15     printf("정적 지역변수 sindex: %2d,\n", sindex++);
16     printf("자동 지역변수 aindex: %2d,\n", aindex++);
17 }
  
```

```

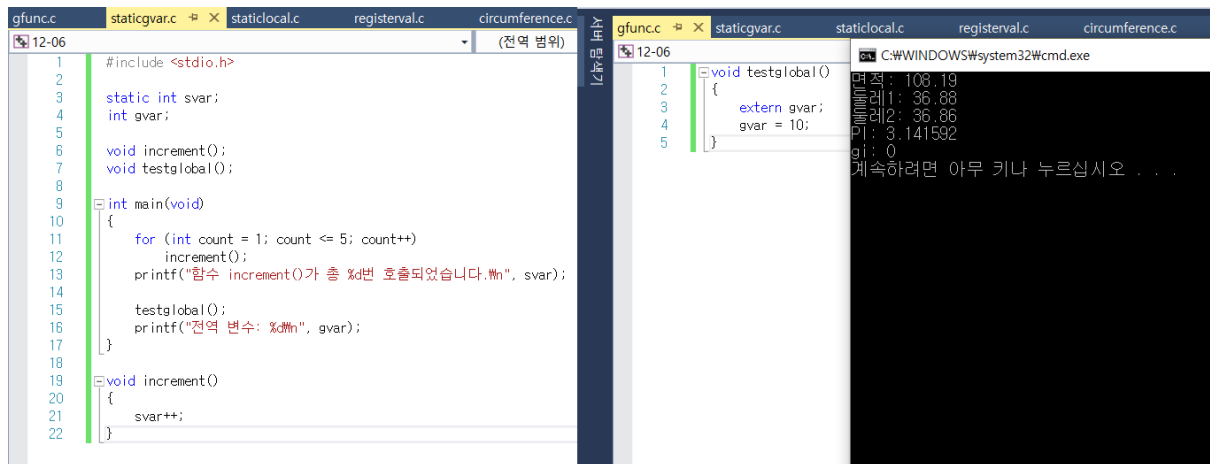
C:\WINDOWS\system32\cmd.exe
정적 지역변수 sindex: 1, 자동 지역변수 aindex: 1,
정적 지역변수 sindex: 2, 자동 지역변수 aindex: 1,
정적 지역변수 sindex: 3, 자동 지역변수 aindex: 1,
계속하려면 아무 키나 누르십시오 . . .
  
```

정적 전역변수

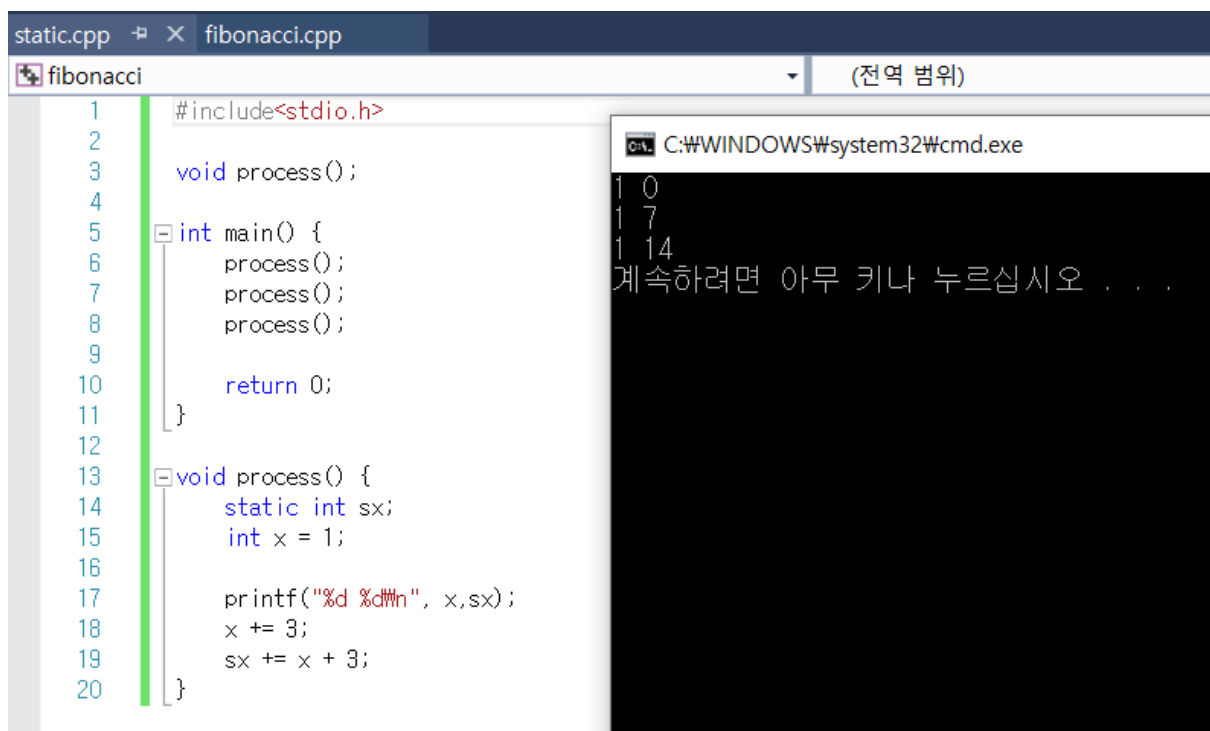
- 함수 외부에서 정적으로 선언되는 변수
- 선언된 파일 내부에서만 참조가 가능한 변수
- 프로그램이 크고 복잡하면 전역변수의 사용은 원하지 않는 전역변수의 수정과 같은 부작용의 위험성이 항상 존재함

정적 지역변수 선언과 사용 (실습예제 12-6)

-정적 전역변수 선언과 사용 (실습예제 12-7)



-지역변수와 정적변수의 사용 (Lab 12-2)



12-3 메모리 영역과 변수 이용

데이터, 스택, 힙 영역

- 메인 메모리의 영역은 프로그램 실행 과정에서 데이터 영역, 힙 영역, 스택 영역 세 부분으로 나뉜다.
- 메모리 영역은 변수의 유효범위와 생존기간에 결정적 역할을 함
- 변수는 기억부류에 따라 할당되는 메모리 공간이 달라짐
- 데이터 영역 : 전역 변수와 정적변수가 할당되는 저장공간

- 힙 영역 : 동적 할당되는 변수가 할당되는 저장공간
- 스택 영역은 함수 호출에 의한 형식 매개변수 그리고 함수내부의 지역변수가 할당되는 저장 공간이다
- 스택 영역은 메모리 주소가 높은 값에서 낮은 값으로 저장 장소가 할당되고 함수 호출과 종료에 따라 높은 주소에서 낮은 주소를 메모리가 할당되었다가 다시 제거되는 작업이 반복된다

변수의 이용 기준

- 실행 속도를 개선하고자 하는 경우에 제한적으로 특수한 지역변수인 레지스터 변수를 이용
- 함수나 블록 내부에서 함수나 블록이 종료되더라도 계속적으로 값을 저장하고 싶을 때는 정적 지역변수를 이용
- 해당 파일 내부에서만 변수를 공유하고자 하는 경우는 정적 전역변수를 이용
- 프로그램의 모든 영역에서 값을 공유하고자 하는 경우는 전역 변수를 이용

-전역변수와 지역변수의 선언과 참조 (실습 예제 12-8)

-외부 파일의 전역변수 참조 (실습 예제 12-9)

The screenshot displays two source files and their execution output. The left window shows the source code for 'storageclass.c' and 'gfunc.c'. The right window shows the execution output of 'storageclass.c'.

```

storageclass.c
1 #include <stdio.h>
2
3 void infunction(void);
4 void outfunction(void);
5
6 int global = 10;
7 static int sglobal = 20;
8
9 int main(void)
10 {
11     auto int x = 100;
12
13     printf("%d, %d, %d\n", global, sglobal, x);
14     infunction(); outfunction();
15     infunction(); outfunction();
16     infunction(); outfunction();
17     printf("%d, %d, %d\n", global, sglobal, x);
18
19     return 0;
20 }
21
22 void infunction(void)
23 {
24     auto int fa = 1;
25     static int fs;
26     printf("in: %d, %d, %d\n", ++global, ++sglobal, fa, ++fs);
27 }
gfunc.c
1 #include <stdio.h>
2
3 void outfunction()
4 {
5     extern int global, sglobal;
6     printf("out: %d, %d\n", global, sglobal);
7 }

```

Execution output of storageclass.c:

```

10, 20, 100
11, 21, 1, 1
12
13, 22, 1, 2
14
15, 23, 1, 3
16
16, 23, 100
계속하려면 아무 키나 누르십시오 . . .

```

-은행계좌의 입출금 구현 (Lab 12-3)

The screenshot shows a C++ IDE with a file named `fibonacci.cpp` open. The code defines a bank account simulation with a `main` function and two helper functions, `save` and `withdraw`. The `main` function initializes a total balance of 100,000 and performs a series of deposits and withdrawals. The `save` function adds money to the total and prints the transaction details. The `withdraw` function subtracts money from the total and prints the transaction details. The output window shows the results of these operations, including a table of transactions and a final balance.

```

1 #include<stdio.h>
2
3 int total = 10000;
4
5 void save(int);
6 void withdraw(int);
7
8 int main() {
9     printf("입금액  출금액  총입금액  총출금액  잔고\n");
10    printf("=====*\n");
11    printf("%46d\n", total);
12
13    save(50000);
14    withdraw(30000);
15    save(60000);
16    withdraw(20000);
17    printf("=====*\n");
18
19    return 0;
20 }
21
22 void save(int money) {
23     static int amount;
24     total += money;
25     amount += money;
26     printf("%7d %17d %20d\n", money, amount, total);
27 }
28
29 void withdraw(int money) {
30     static int amount;
31     total -= money;
32     amount += money;
33     printf("%15d %20d %9d\n", money, amount, total);
34 }

```

The output window shows the following table:

입금액	출금액	총입금액	총출금액	잔고
50000		50000		10000
	30000		30000	60000
60000		110000		30000
	20000		50000	90000
				70000

Below the table, the output window displays the text: "계속하려면 아무 키나 누르십시오 . . ."

13-1 구조체와 공용체

구조체 개념

- 정수나 문자, 실수나 포인터 그리고 이들의 배열 등을 묶어 하나의 자료형으로 이용하는 것이 구조체
- 연관성이 있는 서로 다른 개별적인 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형
- 구조체는 연관된 멤버로 구성되는 통합 자료형으로 대표적인 유도 자료형
- 유도 자료형은 기존 자료형으로 새로이 만들어진 자료형

구조체 정의

- 구조체를 자료형으로 사용하려면 구조체를 정의해야 함
- 구조체를 사용하려면 구조체를 만들 구조체 틀(template)를 정의해야 함
- 구조체를 정의하는 방법은 키워드 `struct` 다음에 구조체 태그이름을 기술하고 중괄호를 이용하여 원하는 멤버를 여러 개의 변수로 선언하는 구조
- 구조체를 구성하는 하나 하나의 항목을 구조체 멤버 또는 필드라 함
- 구조체 정의는 변수의 선언과는 다른 것으로 변수선언에서 이용될 새로운 구조체 자료형을 정의하는 구문

- 구조체 멤버로는 일반변수, 포인터 변수, 배열, 다른 구조체 변수 및 구조체 포인터도 허용

구조체 변수 선언

- 새로운 자료형 struct account 형 변수 mine을 선언하려면 struct account mine;으로 선언

구조체 변수의 초기화

- 초기화 값은 다음과 같이 중괄호 내부에서 구조체의 각 멤버 정의 순서대로 초기값을 쉼표로 구분하여 기술
- 배열과 같이 초기값에 기술되지 않은 멤버값은 자료형에 따라 기본값인 0, 0.0, 'W0' 등으로 저장

구조체의 멤버 접근 연산자 . 와 변수 크기

- 삽입된 구조체형 변수는 접근연산자 .를 사용하여 멤버를 참조
- 실제 구조체의 크기는 멤버의 크기의 합보다 크거나 같다

-구조체 정의와 구조체 변수 선언 (실습 예제 13-1)

```

1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <string.h>
4
5  struct account
6  {
7      char name[20];
8      int actnum;
9      double balance;
10 };
11
12 int main(void)
13 {
14     struct account mine = { "홍길동", 1001, 300000 };
15     struct account yours;
16
17     strcpy(yours.name, "이동원");
18     yours.actnum = 1002;
19     yours.balance = 500000;
20
21     printf("구조체 크기: %d\n", sizeof(mine));
22     printf("%s %d %.2f\n", mine.name, mine.actnum, mine.balance);
23     printf("%s %d %.2f\n", yours.name, yours.actnum, yours.balance);
24
25     return 0;
26 }

```

Output (C:\WINDOWS\system32\cmd.exe):

```

구조체 크기: 32
홍길동 1001 300000.00
이동원 1002 500000.00
계속하려면 아무 키나 누르십시오 . . .

```

구조체 멤버로 사용되는 구조체

- 구조체 멤버로 이미 정의된 다른 구조체 형 변수와 자기 자신을 포함한 구조체 포인터 변수를 사용할 수 있음

-구조체 멤버로 다른 구조체 허용 (실습 예제 13-2)

```

1  #include <stdio.h>
2  #include <string.h>
3
4  struct date
5  {
6      int year;
7      int month;
8      int day;
9  };
10
11 struct account
12 {
13     struct date open;
14     char name[12];
15     int actnum;
16     double balance;
17 };
18
19 int main(void)
20 {
21     struct account me = { {2018, 3, 9}, "홍길동", 1001, 300000 };
22
23     printf("구조체 크기: %d\n", sizeof(me));
24     printf("[%d, %d, %d]\n", me.open.year, me.open.month, me.open.day);
25     printf("%s %d %.2f\n", me.name, me.actnum, me.balance);
26 }
    
```

Output (C:\WINDOWS\system32\cmd.exe):

```

구조체 크기: 40
[2018, 3, 9]
홍길동 1001 300000.00
계속하려면 아무 키나 누르십시오 . . .
    
```

구조체 변수의 대입과 동등비교

- 구조체 멤버마다 모두 대입할 필요 없이 변수 대입으로 한번에 모든 멤버의 대입이 가능

-(실습 예제 13-3)

```

1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <string.h>
4
5  int main(void)
6  {
7      struct student
8      {
9          int snum;
10         char *dept;
11         char name[12];
12     };
13
14     struct student hong = { 201800001, "컴퓨터정보공학과", "홍길동" };
15     struct student na = { 201800002 };
16     struct student bae = { 201800003 };
17
18     scanf("%s", na.name);
19
20     na.dept = "컴퓨터정보공학과";
21     bae.dept = "기계공학과";
22     memcpy(bae.name, "배상문", 7);
23     strcpy_s(bae.name, 7, "배상문");
24     strcpy_s(bae.name, 7, "배상문");
25
26     printf("[%d, %s, %s]\n", hong.snum, hong.dept, hong.name);
27     printf("[%d, %s, %s]\n", na.snum, na.dept, na.name);
28     printf("[%d, %s, %s]\n", bae.snum, bae.dept, bae.name);
29
30     struct student one;
31     one = bae;
32     if (one.snum == bae.snum)
33         printf("학번이 %d으로 동일합니다.\n", one.snum);
34     if (one.snum == bae.snum && !strcmp(one.name, bae.name) && !strcmp(one.dept, bae.dept))
35         printf("내용이 같은 구조체입니다.\n");
36
37     return 0;
38 }
    
```

Output (C:\WINDOWS\system32\cmd.exe):

```

나한국
[201800001, 컴퓨터정보공학과, 홍길동]
[201800002, 컴퓨터정보공학과, 나한국]
[201800003, 기계공학과, 배상문]
학번이 201800003으로 동일합니다.
내용이 같은 구조체입니다.
계속하려면 아무 키나 누르십시오 . . .
    
```

공용체 개념

- 동일한 저장 장소에 여러 자료형을 저장하는 방법

Union 을 사용한 공용체 정의 및 변수 선언

- 공용체는 서로 다른 자료형의 값을 동일한 저장공간에 저장하는 자료형
- 공용체 변수의 크기는 멤버 중 가장 큰 자료형의 크기로 정해짐
- 공용체의 멤버는 모든 멤버가 동일한 저장 공간을 사용하므로 동시에 멤버의 값을 저장하여 이용할 수 없으며, 마지막에 저장된 단 하나의 멤버 자료값만을 저장함
- 공용체의 초기화 값은 공용체 정의 시 처음 선언한 멤버의 초기값으로만 저장이 가능하다

공용체 멤버 접근

- 공용체 변수로 멤버를 접근하기 위해서는 구조체와 같이 접근연산자 .를 사용한다

-공용체 정의와 변수 선언 및 사용 (실습 예제 13-4)

```

1  #include <stdio.h>
2
3  union data
4  {
5      char ch;
6      int cnt;
7      double real;
8  } data1;
9
10 int main(void)
11 {
12     union data data2 = { 'A' };
13     union data data3 = data2;
14
15     printf("%d %d\n", sizeof(union data), sizeof(data3));
16
17     data1.ch = 'a';
18     printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
19
20     data1.cnt = 100;
21     printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
22
23     data1.real = 3.156759;
24     printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
25
26     return 0;
27 }
  
```

```

C:\WINDOWS\system32\cmd.exe
8 8
a 97 0.000000
d 100 0.000000
N -590162866 3.156759
계속하려면 아무 키나 누르십시오 . . .
  
```

-도시의 이름과 위치를 표현하는 구조체 (Lab 13-1)

```

structcity.c  structarray.c  unionpointer.c  structpointer.c  typedefstruct.c  typedef.c  13-4.cpp  13-3.cpp
LAB 13-1  (전역 범위)
1  #include <stdio.h>
2  #include <string.h>
3
4  struct position
5  {
6      double latitude;
7      double longitude;
8  };
9
10 int main(void)
11 {
12     struct city
13     {
14         char *name;
15         struct position place;
16     };
17     struct city seoul, newyork;
18     seoul.name = "서울";
19     seoul.place.latitude = 37.33;
20     seoul.place.longitude = 126.58;
21     newyork.name = "뉴욕";
22     newyork.place.latitude = 40.8;
23     newyork.place.longitude = 73.9;
24
25     printf("[%s] 위도= %.1f 경도= %.1f\n",
26           seoul.name, seoul.place.latitude, seoul.place.longitude);
27     printf("[%s] 위도= %.1f 경도= %.1f\n",
28           newyork.name, newyork.place.latitude, newyork.place.longitude);
29
30     return 0;
31 }
C:\WINDOWS\system32\cmd.exe
[서울] 위도= 37.3 경도= 126.6
[뉴욕] 위도= 40.8 경도= 73.9
계속하려면 아무 키나 누르십시오 . . .

```

13-2 자료형 재정의

typedef 구문

- Typedef : 이미 사용되는 자료 유형을 다른 새로운 자료형 이름으로 재정의 할 수 있도록 하는 키워드
- 일반적으로 자료형을 재정의하는 이유는 프로그램의 시스템 간 호환성과 편의성을 위해 필요
- 문장 typedef도 일반 변수와 같이 그 사용 범위를 제한

-자료형 재정의 이용 (실습 예제 13-5)

```

typedef.c  13-4.cpp  13-3.cpp  13-2.cpp  13-1.cpp
13-05  (전역 범위)
1  #include <stdio.h>
2
3  typedef unsigned int budget;
4
5  int main(void)
6  {
7      budget year = 24500000;
8      typedef int profit;
9      profit month = 4600000;
10
11     printf("올 예산은 %d, 이달의 이익은 %d 입니다.\n", year, month);
12
13     return 0;
14 }
15 void test(void)
16 {
17     budget year = 24500000;
18 }
C:\WINDOWS\system32\cmd.exe
올 예산은 24500000, 이달의 이익은 4600000 입니다.
계속하려면 아무 키나 누르십시오 . . .

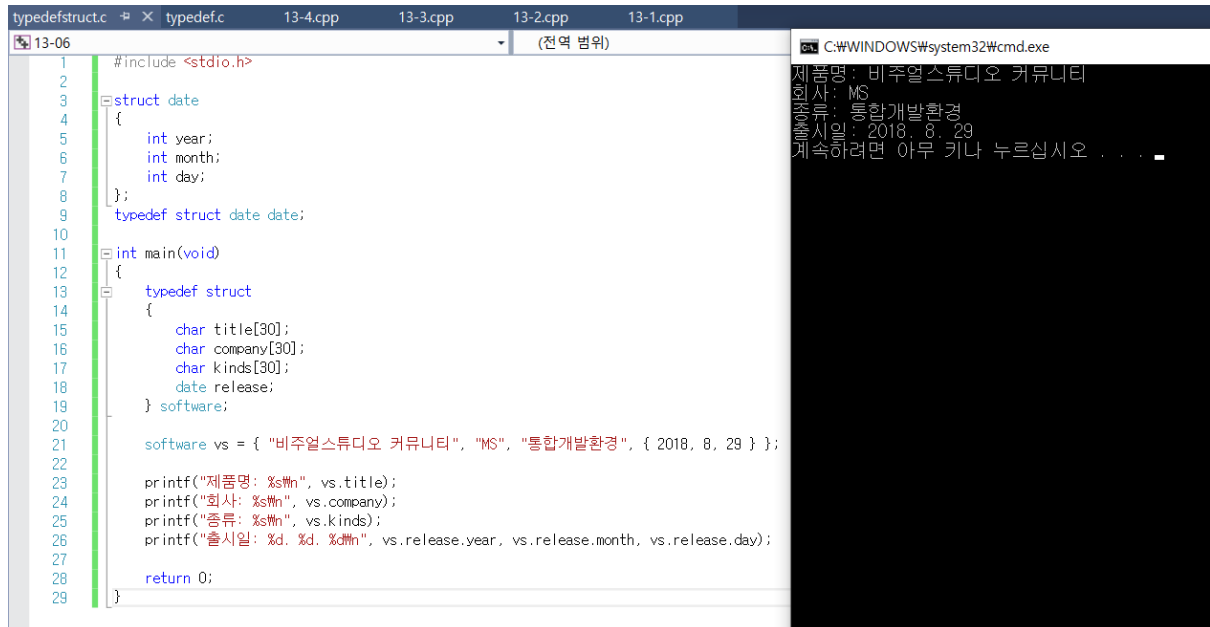
```

Struct 를 생략한 새로운 자료형

- 구조체 struct date 가 정의된 상태에서 typedef 사용하여 구조체 struct date를 date로 재정의할 수 있음

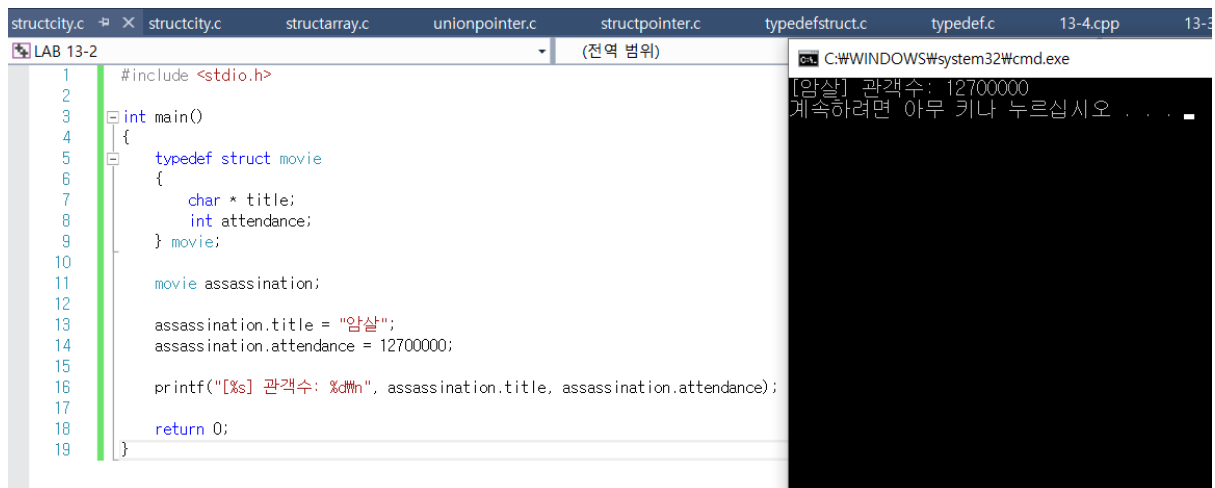
-문장 typedef 를 이용하여 구조체의 자료유형을 다른 이름으로 재정의하여 이용

(실습 예제 13-6)



```
1 #include <stdio.h>
2
3 struct date
4 {
5     int year;
6     int month;
7     int day;
8 };
9 typedef struct date date;
10
11 int main(void)
12 {
13     typedef struct
14     {
15         char title[30];
16         char company[30];
17         char kinds[30];
18         date release;
19     } software;
20
21     software vs = { "비주얼 스튜디오 커뮤니티", "MS", "통합개발환경", { 2018, 8, 29 } };
22
23     printf("제품명: %s\n", vs.title);
24     printf("회사: %s\n", vs.company);
25     printf("종류: %s\n", vs.kinds);
26     printf("출시일: %d, %d, %d\n", vs.release.year, vs.release.month, vs.release.day);
27
28     return 0;
29 }
```

-영화 정보를 표현하는 구조체 (Lab 13-2)



```
1 #include <stdio.h>
2
3 int main()
4 {
5     typedef struct movie
6     {
7         char * title;
8         int attendance;
9     } movie;
10
11     movie assassination;
12
13     assassination.title = "암살";
14     assassination.attendance = 12700000;
15
16     printf("[%s] 관객수: %d\n", assassination.title, assassination.attendance);
17
18     return 0;
19 }
```

13-3 구조체와 공용체의 포인터와 배열

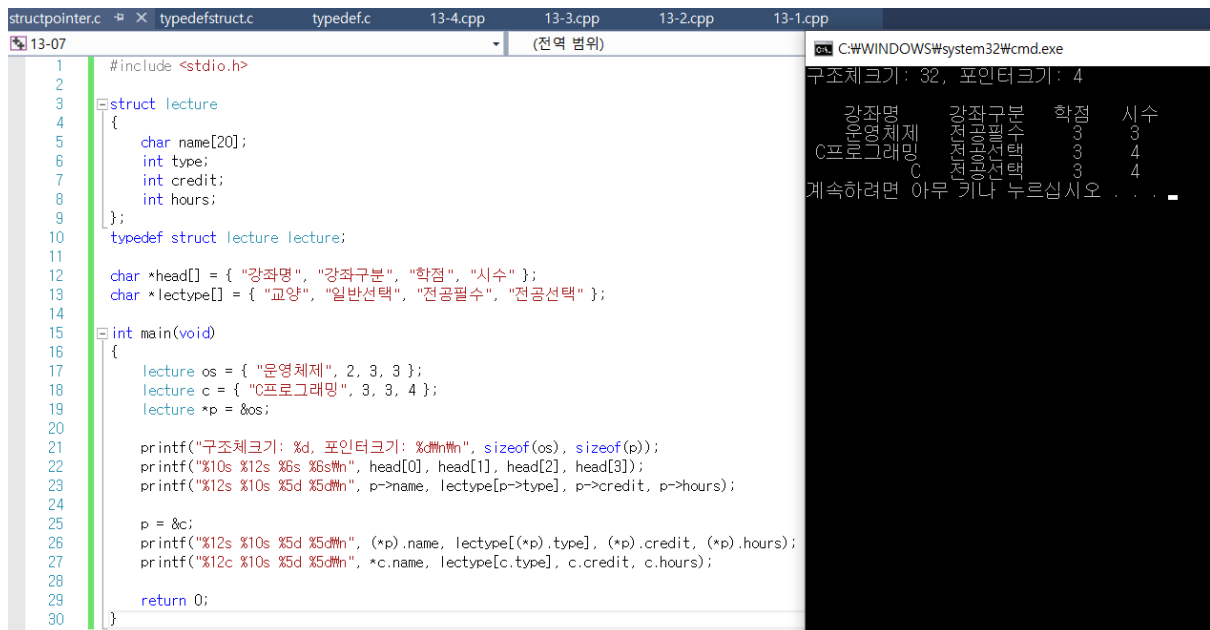
포인터 변수 선언

- 포인터는 각각의 자료형 저장 공간의 주소를 저장하듯이 구조체 포인터는 구조체의 주소값을 저장 할 수 있는 변수이다
- 구조체 포인터 변수의 선언은 일반 포인터 변수 선언과 동일

포인터 변수의 구조체 멤버 접근 연산자 ->

- 구조체 포인터 멤버 접근연산자 ->는 p -> name과 같이 사용한다
- 연산식 p -> name은 포인터 p가 가리키는 구조체 변수의 멤버 name을 접근하는 연산식
- 연산식 *p.name은 접근연산자(.)가 간접연산자 (*)보다 우선순위가 빠르므로 *(p.name)과 같은 연산식이다-

-구조체 포인터의 선언과 사용 (실습 예제 13 - 7)



```
1 #include <stdio.h>
2
3 struct lecture
4 {
5     char name[20];
6     int type;
7     int credit;
8     int hours;
9 };
10 typedef struct lecture lecture;
11
12 char *head[] = { "강좌명", "강좌구분", "학점", "시수" };
13 char *lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
14
15 int main(void)
16 {
17     lecture os = { "운영체제", 2, 3, 3 };
18     lecture c = { "C프로그래밍", 3, 3, 4 };
19     lecture *p = &os;
20
21     printf("구조체크기: %d, 포인터크기: %d\n", sizeof(os), sizeof(p));
22     printf("%10s %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
23     printf("%12s %10s %5d %5d\n", p->name, lectype[p->type], p->credit, p->hours);
24
25     p = &c;
26     printf("%12s %10s %5d %5d\n", (*p).name, lectype[(*p).type], (*p).credit, (*p).hours);
27     printf("%12c %10s %5d %5d\n", *c.name, lectype[c.type], c.credit, c.hours);
28
29     return 0;
30 }
```

C:\WINDOWS\system32\cmd.exe

구조체크기: 32, 포인터크기: 4

강좌명	강좌구분	학점	시수
운영체제	전공필수	3	3
C프로그래밍	전공선택	3	4
C	전공선택	3	4

계속하려면 아무 키나 누르십시오 . . .

공용체 포인터

- 포인터 변수 사용이 가능
- 공용체 포인터 변수로 멤버를 접근하려면 접근연산자 ->를 이용

-공용체 정의와 선언 및 사용 (실습 예제 13-8)

```

1  #include <stdio.h>
2
3  int main(void)
4  {
5      union data
6      {
7          char ch;
8          int cnt;
9          double real;
10     };
11     typedef union data udata;
12
13     udata value, *p;
14
15     p = &value;
16     p->ch = 'a';
17     printf("%c %c\n", p->ch, (*p).ch);
18     p->cnt = 100;
19     printf("%d ", p->cnt);
20     p->real = 3.14;
21     printf("%.2f\n", p->real);
22
23     return 0;
24 }

```

Terminal Output:

```

a a
100 3.14
계속하려면 아무 키나 누르십시오 . . .

```

구조체 배열 변수 선언

- 다른 배열과 같이 동일한 구조체 변수가 여러 개 필요하면 구조체 배열을 선언하여 이용할 수 있다

-구조체 배열을 선언한 후 출력 처리 (실습 예제 13-9)

```

1  #include <stdio.h>
2
3  struct lecture
4  {
5      char name[20];
6      int type;
7      int credit;
8      int hours;
9  };
10 typedef struct lecture lecture;
11
12 char *lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
13 char *head[] = { "강좌명", "강좌구분", "학점", "시수" };
14
15 int main(void)
16 {
17     lecture course[] = { { "인간과 사회", 0, 2, 2 },
18     { "경제학개론", 1, 3, 3 },
19     { "자료구조", 2, 3, 3 },
20     { "모바일프로그래밍", 2, 3, 4 },
21     { "고급 C프로그래밍", 3, 3, 4 } };
22
23     int arysize = sizeof(course) / sizeof(course[0]);
24
25     printf("배열 크기: %d\n", arysize);
26     printf("%12s %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
27     printf("=====");
28     for (int i = 0; i < arysize; i++)
29         printf("%16s %10s %5d %5d\n", course[i].name,
30             lectype[course[i].type], course[i].credit, course[i].hours);
31
32     return 0;
33 }

```

Terminal Output:

```

배열 크기: 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

계속하려면 아무 키나 누르십시오 . . .

-영화 정보를 표현하는 구조체의 배열 (Lab 13-3)

The screenshot shows the Microsoft Visual Studio IDE with a C program in the main editor and its execution output in the console window.

Source Code (structure.c):

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3 #include <string.h>
4
5 int main(void)
6 {
7     typedef struct movie
8     {
9         char *title;
10        int attendance;
11        char director[20];
12    } movie;
13
14    movie box[] = {
15        {"영웅", 17813000, "김한민"},
16        {"국제시장", 14257000, "윤제균"},
17        {"베테랑", 13383000, ""},
18    };
19    strcpy(box[2].director, "류승완");
20
21    printf(" 제목   감독   관객수\n");
22    printf("=====");
23    for (int i = 0; i < 3; i++)
24        printf("%8s %8s %8s\n",
25               box[i].title, box[i].director, box[i].attendance);
26
27    return 0;
28 }
```

Execution Output:

```
제목   감독   관객수
영웅   김한민  17813000
국제시장 윤제균  14257000
베테랑  류승완  13383000
계속하려면 아무 키나 누르십시오 . . .
```