

# C어플리케이션 포트폴리오

---



20173842      한근희

# <목차>

목차 .....	P.1
머릿말 .....	P.2
강의 계획서 .....	P.3-6

## <본 내용>

1주차 강의 .....	P.7-11
1주차 소감 .....	P.11
2주차 강의 .....	P.12-29
2주차 소감 .....	P.29
3주차 강의 .....	P.30-31
3주차 소감 .....	P.31
4주차 강의 .....	P.32-43
4주차 소감 .....	P.43
5주차 강의 .....	P.44-51
5주차 소감 .....	P.51

## <머릿말>

나는 문과 출신으로 컴퓨터와는 별로 친한 관계는 아니었다.

취미 생활을 향유 할 정도로만 컴퓨터를 활용하여 생활을 했지 컴퓨터 자체로 어떤 프로그램을 ‘개발’ 한다는 것은 나와는 거리가 먼 것인 줄만 알았다.

어쩌다 보니 진로를 정하며 프로그래밍에 흥미를 가지게 되었지만 그것이 나의 대학교 학부가 될 줄은 몰랐다. 물론 지금은 다양한 프로그래밍 언어와 툴 등을 배우며 프로그래머로서의 길을 걷고 있고 현재 2학년 1학기 까지 도달하였다.

구체적으로 어느 분야로 나아갈지 정하진 않았다. 하지만 내가 흥미를 느끼는 분야도 대략 파악 하였고 C언어를 배우며 여러 가지 어려운 점을 극복해 나가며 머릿속에 지식을 쌓아가며 보람을 느끼기도 한다.

그리고 지금 이렇게 나의 C어플리케이션 구현 과정을 담은 포트폴리오를 작성한다. 나는 이러한 방식으로 C언어를 익혔고 그 동안의 과정을 나열하며 과거 내가 공부했던 방식을 되돌아 보고 추후 나의 코딩 과정에 도움이 될 수 있으면 좋겠다.

# <강의 계획서>



동양미래대학교  
DONGYANG MIRAE UNIVERSITY

## 강 의 계 획 서

아시아 직업교육 허브대학

2020 학년도 1학기	전공 컴퓨터정보공학과(사회맞춤형 지능형컴퓨팅 과정)	학부	컴퓨터공학부	
과 목 명	C에플리케이션구현(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 : 2호관-706 + 전 화 : 02-2610-1941 + E-MAIL : hskang@dongyang.ac.kr + 면담가능기간 : 월 11시~12시 화 14시~17시			
학과 교육목표				
과목 개요	본 과목은 프로그래밍 언어 중 가장 널리 사용되고 있는 C언어를 학습하는 과목으로 C++, JAVA 등과 같은 언어의 기반이 된다. 본 과목에서는 지난 학기에서 배운 시스템프로그래밍1에 이어 C언어의 기본 구조 및 문법 체계 그리고 응용 프로그래밍 기법 등을 다룬다.  C언어에 대한 학습은 Windows상에서 이루어지며, 기본적인 이론 설명 후 실습문제를 프로그래밍하며 숙지하는 형태로 수업이 진행된다.			
학습목표 및 성취수준	대학 교육목표와 학과 교육목표를 달성하기 위하여 이 과목을 수강함으로써 학습자는 C언어의 문법 전반과 응용 프로그램 기법을 알 수 있다. 직전 학기의 수강으로 인한 C언어의 기초부터 함수, 포인터 등의 내용 이해를 바탕으로하여 이번 학기에는 지난 학기 내용의 전체적인 복습과 함께 C언어 전체를 학습하고, 특히 응용 능력을 배양하여 프로그래밍으로 문제를 해결하는 능력을 익히게 된다.			
	도서명	저자	출판사	비고
주교재	Perfect C	강환수, 강환일, 이동규	인피니티박스	
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: C언어 기초 및 조건문과 반복문 복습			
목표및 내용	C언어 기초 통합개발환경 테스트 기초적인 코드 실습			

미리읽어오기	교재 1~5장
과제,시험,기타	수업 중에 제시함
<b>2 주차</b>	<b>[2주]</b>
학습주제	C언어 기초 문법
목표및 내용	변수와 상수 연산자 l-value와 r-value
미리읽어오기	교재 1~5장
과제,시험,기타	수업 중에 제시함
<b>3 주차</b>	<b>[3주]</b>
학습주제	조건문
목표및 내용	6장 조건문 학습
미리읽어오기	교재 6장
과제,시험,기타	수업 중에 제시함
<b>4 주차</b>	<b>[4주]</b>
학습주제	반복문
목표및 내용	7장 반복문 학습
미리읽어오기	교재 7장
과제,시험,기타	수업 중에 제시함
<b>5 주차</b>	<b>[5주]</b>
학습주제	포인터
목표및 내용	8장 포인터 학습 단일포인터 다중포인터 여러가지 포인터
미리읽어오기	교재 8장
과제,시험,기타	수업 중에 제시함
<b>6 주차</b>	<b>[6주]</b>
학습주제	배열
목표및 내용	9장 배열
미리읽어오기	교재 9장
과제,시험,기타	수업 중에 제시함

<b>7 주차</b>	<b>[7주]</b>
학습주제	함수
목표및 내용	10장 함수
미리읽어오기	교재 10장
과제,시험,기타	수업 중에 제시함
<b>8 주차</b>	<b>[중간고사]</b>
학습주제	중간고사
목표및 내용	중간고사
미리읽어오기	
과제,시험,기타	
<b>9 주차</b>	<b>[9주]</b>
학습주제	문자열
목표및 내용	11장 문자열
미리읽어오기	교재 11장
과제,시험,기타	수업 중에 제시함
<b>10 주차</b>	<b>[10주]</b>
학습주제	변수 유효범위
목표및 내용	12장 변수 유효범위
미리읽어오기	교재 12장
과제,시험,기타	수업 중에 제시함
<b>11 주차</b>	<b>[11주]</b>
학습주제	구조체
목표및 내용	13장 구조체
미리읽어오기	교재 13장
과제,시험,기타	수업 중에 제시함
<b>12 주차</b>	<b>[12주]</b>
학습주제	함수와 포인터 활용
목표및 내용	14장 함수와 포인터활용
미리읽어오기	교재 14장
과제,시험,기타	수업 중에 제시함

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

# <1주차 3월 16일자 수업>

## -문자와 문자열-

### 1. 문자와 문자열 선언

#### 가. 문자와 문자열의 개념

- 1) 문자 : 영어의 알파벳이나 한글의 한 글자를 작은 따옴표로 둘러싸서 'A' 와 같이 표기하며 1바이트인 자료형 char로 지원 → 작은 따옴표에 의해 표기된 문자 상수
- 2) 문자열 : 문자의 모임인 일련의 문자, 문자 앞뒤로 큰 따옴표를 둘러싸서 "AAAA" 등으로 표기

(‘AAA’처럼 작은 따옴표를 문자열로 감싸면 오류)

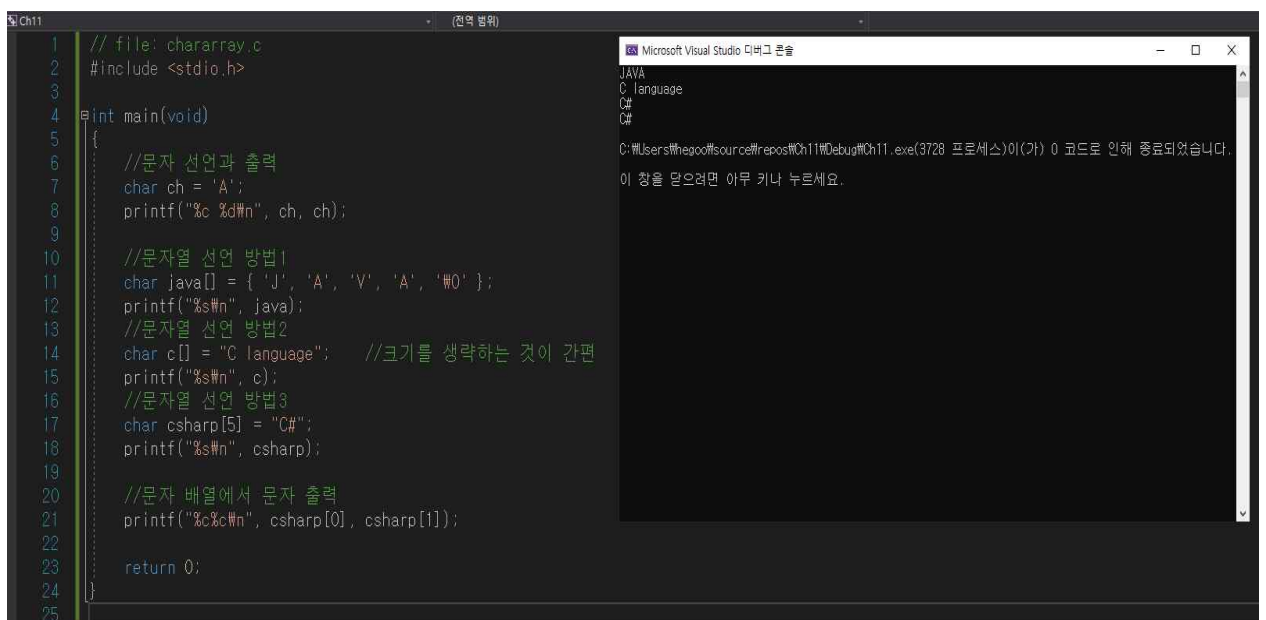
- 3) 문자는 C언어에서 char형 변수에 문자를 저장
- 4) 문자열을 저장하려면 문자의 모임인 ‘문자 배열’을 사용

(문자열의 마지막을 의미하는 NULL문자 ‘\0’가 마지막에 저장)

→ 문자열이 저장되는 배열 크기는 반드시 저장될 문자 수보다 1이 커야함

- 5) 배열 선언 시 저장할 큰 따옴표를 사용해 문자열 상수를 바로 대입하는 방법  
(배열 초기화시 배열 크기는 지정X, 만일 지정한 배열 크기가 (문자수+1) 보다 크면 나머지 부분은 모두 ‘\0’ 문자로 채워짐)

#### 예제 코드



```
1 // file: chararray.c
2 #include <stdio.h>
3
4 int main(void)
5 {
6     //문자 선언과 출력
7     char ch = 'A';
8     printf("%c %d\n", ch, ch);
9
10    //문자열 선언 방법1
11    char java[] = { 'J', 'A', 'V', 'A', '\0' };
12    printf("%s\n", java);
13    //문자열 선언 방법2
14    char c[] = "C language"; //크기를 생각하는 것이 간편
15    printf("%s\n", c);
16    //문자열 선언 방법3
17    char csharp[5] = "C#";
18    printf("%s\n", csharp);
19
20    //문자 배열에서 문자 출력
21    printf("%c%c\n", csharp[0], csharp[1]);
22
23    return 0;
24 }
25
```

Microsoft Visual Studio 디버그 콘솔

```
JAVA
C language
C#
C#

C:\Users\wngoo\source\repos\Ch11\Debug\Ch11.exe(3728 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

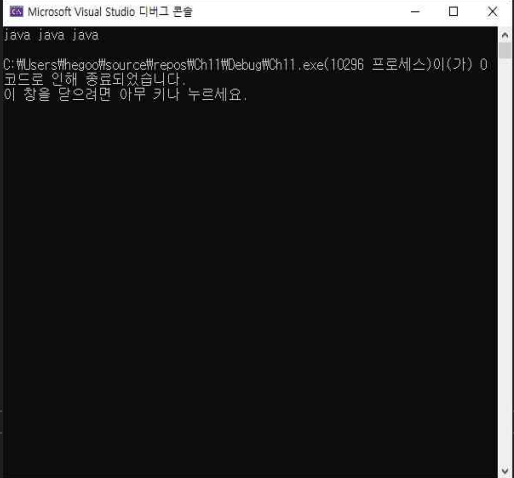


## 나. 문자열 구성하는 문자 참조

- 1) 문자열 상수를 문자 포인터에 저장하는 방식 존재
- 2) 문자열 출력도 함수 printf()에서 포인터 변수와 형식제어문자 %s로 간단히 처리  
but, 문자 포인터에 의한 선언으로는 문자 하나하나의 수정은 불가

### 예제 코드

```
1 // file: charpointer.c
2 #include <stdio.h>
3
4 int main(void)
5 {
6     char* java = "java";
7     printf("%s", java);
8
9     //문자 포인터가 가리키는 문자 이후를 하나 하나 출력
10    int i = 0;
11    while (java[i]) //while (java[i] != '\0')
12        printf("%c", java[i++]);
13    printf(" ");
14
15    i = 0;
16    while (*(java + i) != '\0') //java[i]는 *(java + i)와 같음
17        printf("%c", *(java + i++));
18    printf("\n");
19
20
21    return 0;
22 }
23
```

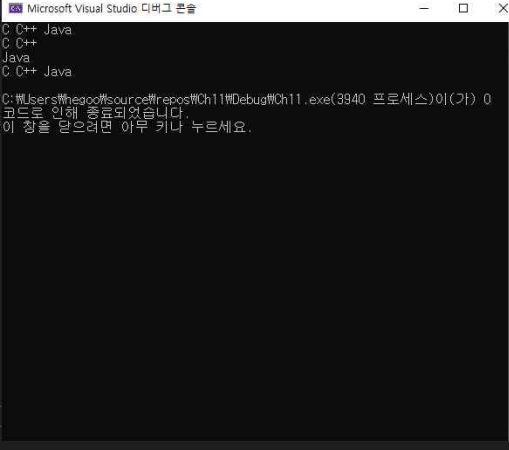


## 다. '/0' 문자에 의한 문자열 분리

- 1) 함수 printf()에서 %s는 문자 포인터가 가리키는 위치에서 NULL 문자까지를 하나의 문자열로 인식

### 예제 코드

```
1 // file: string.c
2 #include <stdio.h>
3
4 int main(void)
5 {
6     char c[] = "C C++ Java";
7     printf("%s\n", c);
8     c[5] = '\0'; //널 문자에 의해 문자열 분리
9     printf("%s\n%s\n", c, (c + 6));
10
11    //문자 배열의 각 원소를 하나 하나 출력하는 방법
12    c[5] = ' '; //널 문자를 빈 문자로 바꾸어 문자열 복원
13    char* p = c;
14    while (*p) //(*p != '\0')도 가능
15        printf("%c", *p++);
16    printf("\n");
17
18    return 0;
19 }
20
```



## 2. 다양한 문자 입출력

### 가. 버퍼처리 함수 getchar()

- 1) 함수 `getchar()`는 문자의 입력에 사용되고 `putchar()`는 문자의 출력에 사용  
(문자 입력을 위한 함수 `getchar()`는 라인 버퍼링 방식을 사용)

### 나. 함수 `getche()`

- 1) 함수 `getche()`는 버퍼를 사용하지 않고 문자를 입력하는 함수이다.  
(함수는 버퍼를 사용하지 않고 문자 하나를 바로 입력할 수 있는 함수.)  
(함수를 이용하려면 헤더파일 `conio.h`를 삽입해야함)

### 다. 함수 `getch()`

- 1) 문자 입력을 위한 함수 `getch()`는 입력한 문자가 화면에 보이지 않는 특성을 지님

함수	<code>scanf("%c", &amp;ch)</code>	<code>getchar()</code>	<code>getche()</code> <code>_getche()</code>	<code>getch()</code> <code>_getch()</code>
헤더파일	stdio.h		conio.h	
버퍼 이용	버퍼 이용함		버퍼 이용 안함	
반응	[enter] 키를 눌러야 작동		문자 입력마다 반응	
입력 문자의 표시(echo)	누르면 바로 표시		누르면 바로 표시	표시 안됨
입력문자 수정	가능		불가능	

#### 예제 코드

```

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 }

```

```

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

문자를 누를 때마다 두 번 출력 >>
jjaavvaq

문자를 누르면 한 번 출력 >>
java

```

### 3. 문자열 입력

#### 가. gets()와 puts()

- 1) 함수 gets()는 한 행의 문자열 입력에 유용한 함수이다.  
(함수 gets()는 마지막에 입력된 '\n'가 '\0'로 교체되어 인자인 배열에 저장)
- 2) 함수 puts()는 한 행에 문자열을 입력하는 함수이다.
- 3) 기호 상수 EOF는 파일의 끝이라는 의미로 stdio.h 헤더파일에 정수 -1로 정의됨

```
char * gets(char * buffer);
```

- 함수 gets()는 문자열을 입력 받아 buffer에 저장하고 입력 받은 첫 문자의 주소값을 반환한다.
- 함수 gets()는 표준입력으로 [enter] 키를 누를 때까지 공백을 포함한 한 행의 모든 문자열을 입력 받는다.
- 입력된 문자열에서 마지막 [enter] 키를 '\0' 문자로 대체하여 저장한다.

```
char * gets_s(char * buffer, size_t sizebuffer);
```

- 두 번째 인자인 sizebuffer는 정수형으로 buffer의 크기를 입력한다.
- Visual C++에서는 앞으로 gets() 대신 함수 gets\_s()의 사용을 권장한다.

```
int puts(const char * str);
```

- 인자인 문자열 str에서 마지막 '\0' 문자를 개행 문자인 '\n'로 대체하여 출력한다.
- 함수 puts()는 일반적으로 정수값 0을 반환하는데, 오류가 발생하면 EOF를 반환한다.

예제 코드

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3
4
5  int main(void)
6  {
7      char line[101];
8
9      printf("입력을 종료하려면 새로운 행에서 (ctrl + Z)를 누르십시오.\n");
10     while (gets(line))
11         puts(line);
12     printf("\n");
13
14     while (gets_s(line, 101))
15         puts_s(line);
16     printf("\n");
17
18
19     return 0;
20 }
```

```
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
```

## LAB) 한 행을 표준입력으로 받아 문자 하나 하나를 그대로 출력

```

1 // lineprint.c:
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4
5 int main()
6 {
7     char s[100];
8     //문자배열 s에 표준입력한 한 행을 저장
9     gets(s);
10
11     //문자배열에 저장된 한 행을 출력
12     char* p = s;
13     while (*p)
14         printf("%c", *p++);
15     printf("\n");
16
17     return 0;
18 }

```

Microsoft Visual Studio 디버그 콘솔

```

line
line
C:\Users\megoo\source\repos\Ch11\Debug\Ch11.exe(5612 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

```

## <1주차 과제>

### 과제정보

과제명	[과제1]
제출기간	2020-03-12 (00:00)~2020-03-22 (23:59)
참고자료	
과제내용	교재 실습예제 11-1, 3, 4, 5, 6의 구현, 코딩과 결과를 PDF 파일로 가상강좌 [과제1]에 제출

### 제출정보

과제설명	PE 20173842 한근희 C어플리케이션구현 과제 1주차 제출합니다
파일	PE 20173842 한근희 C어플리케이션구현 과제 1주차.pdf
점수	미평가
제출일시	2020-03-16 15:50:06

짧은 느낌 : 지난 학기에 배웠던 부분의 복습차 느낌  
배웠던 부분이 기억이 나면서 뿌듯한 느낌을 받았다.

# <2주차 3월 23일자 수업>

## 4. 문자열 복사와 연결

### 가. 함수 strcpy()

- 1) 함수 strcpy()와 strncpy()는 문자열을 복사하는 함수.
- 2) 함수 strcpy()는 앞 인자 문자열 dest에 뒤 문자열 source를 복사

문자열 복사 함수

```
char * strcpy(char * dest, const char * source);
```

- 앞 문자열 dest에 처음에 뒤 문자열 null 문자를 포함한 source를 복사하여 그 복사된 문자열을 반환한다.
- 앞 문자열은 수정되지만 뒤 문자열은 수정될 수 없다.

```
char * strncpy(char * dest, const char * source, size_t maxn);
```

- 앞 문자열 dest에 처음에 뒤 문자열 source에서 n개 문자를 복사하여 그 복사된 문자열을 반환한다.
- 만일 지정된 maxn이 source의 길이보다 길면 나머지는 모두 널 문자가 복사된다. 앞 문자열은 수정되지만 뒤 문자열은 수정될 수 없다.

```
errno_t strcpy_s(char * dest, size_t sizedest, const char * source);
```

```
errno_t strncpy_s(char * dest, size_t sizedest, const char * source, size_t maxn);
```

- 두 번째 인자인 sizedest는 정수형으로 dest의 크기를 입력한다.
- 반환형 errno\_t는 정수형이며 반환값은 오류번호로 성공하면 0을 반환한다.
- Visual C++에서는 앞으로 함수 strcpy\_s()와 strncpy\_s()의 사용을 권장한다.

### 예제 코드

```
Ch11 (전역 범위)
1 // file: strcpy.c
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <string.h>
5
6 int main(void)
7 {
8     char dest[80] = "Java";
9     char source[80] = "C is a language.";
10
11     printf("%s\n", strcpy(dest, source));
12     //printf("%d\n", strcpy_s(dest, 80, source));
13     //printf("%s\n", dest);
14     printf("%s\n", strncpy(dest, "C#", 2));
15
16     printf("%s\n", strncpy(dest, "C#", 3));
17     //printf("%d\n", strncpy_s(dest, 80, "C#", 3));
18     //printf("%s\n", dest);
19
20     return 0;
21 }
22
```

Microsoft Visual Studio 디버깅 콘솔

```
C is a language.
C# is a language.
C#
```

C:\Users\hegoo\source\repos\Ch11\Debug\Ch11.exe(9088 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.



## 나. 함수 strcat()

1) strcat()는 앞 문자열에 뒤 문자열의 null문자까지 연결하여, 앞의 문자열 주소를 반환

문자열 연결 함수

```
char * strcat(char * dest, const char * source);
```

- 앞 문자열 dest에 뒤 문자열 source를 연결(concatenate)해 저장하며, 이 연결된 문자열을 반환하고 뒤 문자열은 수정될 수 없다.

```
char * strncat(char * dest, const char * source, size_t maxn);
```

- 앞 문자열 dest에 뒤 문자열 source중에서 n개의 크기만큼을 연결(concatenate)해 저장하며, 이 연결된 문자열을 반환하고 뒤 문자열은 수정될 수 없다.
- 지정한 maxn이 문자열 길이보다 크면 null 문자까지 연결한다.

```
errno_t strcat_s(char * dest, size_t sizedest, const char * source);
```

```
errno_t strncat_s(char * dest, size_t sizedest, const char * source, size_t maxn);
```

- 두 번째 인자인 sizedest는 정수형으로 dest의 크기를 입력한다.
- 반환형 errno\_t는 정수형이며 반환값은 오류번호로 성공하면 0을 반환한다.
- Visual C++에서는 앞으로 함수strcat\_s()와 strncat\_s()의 사용을 권장한다.

그림 11-18 문자열 연결 함수

예제 코드

```
Ch11 (전역 범위)
1 // file: strcat.c
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <string.h>
5
6 int main(void)
7 {
8     char dest[80] = "C";
9
10    printf("%s\n", strcat(dest, " is "));
11    //printf("%d\n", strcat_s(dest, 80, " is "));
12    //printf("%s\n", dest);
13    printf("%s\n", strncat(dest, "a java", 2));
14    //printf("%d\n", strncat_s(dest, 80, "a proce", 2));
15    //printf("%s\n", dest);
16    printf("%s\n", strcat(dest, "procedural "));
17    printf("%s\n", strcat(dest, "language.));
18
19    return 0;
20 }
21
```

Microsoft Visual Studio 디버그 콘솔

```
C is
C is a
C is a procedural
C is a procedural language.
```

C:\Users\hegool\source\repos\Ch11\Debug\Ch11.exe(13480 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 5. 문자열 분리 및 다양한 문자열 관련 함수

### 가. 함수 strtok()

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

#### 문자열 분리 함수

```
char * strtok(char * str, const char * delim);
```

- 앞 문자열 str에서 뒤 문자열 delim을 구성하는 구분자를 기준으로 순서대로 토큰을 추출하여 반환하는 함수이며, 뒤 문자열 delim은 수정될 수 없다.

```
char * strtok_s(char * str, const char * delim, char ** context);
```

- 마지막 인자인 context는 함수 호출에 사용되는 위치 정보를 위한 인자이며, Visual C++에서는 앞으로 함수 strtok\_s()의 사용을 권장한다.

#### 예제 코드

```
1 // file: strtok.c
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <string.h>
5
6 int main(void)
7 {
8     char str1[] = "C and C++ language are best!";
9     char* delimiter = " ,#!";
10    //char *next_token;
11
12    printf("문자열 \"%s\"을 >>\n", str1);
13    printf("구분자 \"%s\"를 이용하여 토큰을 추출 >>\n", delimiter);
14    char* ptoken = strtok(str1, delimiter);
15    //ptoken = strtok_s(str, delimiter, &next_token);
16    while (ptoken) //(ptoken != NULL)
17    {
18        printf("%s\n", ptoken);
19        ptoken = strtok(NULL, delimiter); //다음 토큰을 반환
20        //ptoken = strtok_s(NULL, delimiter, &next_token); //다음 토큰을 반환
21    }
22
23    return 0;
24 }
25
```

Microsoft Visual Studio 디버그 콘솔

문자열 "C and C++ language are best!"을 >>  
구분자 " ,#!"를 이용하여 토큰을 추출 >>  
C  
and  
C++  
language  
are  
best

C:\Users\hegoc\source\repos\Ch11\Debug\Ch11.exe(14084 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 나. 함수 strlen(), 그 외 등등

- 1) 함수 strlen()은 NULL 문자를 제외한 문자열 길이를 반환하는 함수.
- 2) 함수 strlwr()은 인자를 모두 소문자로 변환하여 반환하는 함수
- 3) 함수 strupr()은 반대로 인자를 모두 대문자로 변환하여 반환하는 함수

함수원형	설명
char * strlwr(char * str); errno_t _strlwr_s(char * str, size_t strsize); //Visual C++ 권장함수	문자열 str을 모두 소문자로 변환하고 변환한 문자열을 반환하므로 str은 상수이면 오류가 발생하며, errno_t는 정수형의 오류번호이며, size_t도 정수형으로 strsize는 str의 길이
char * strupr(char * str); errno_t _strupr_s(char * str, size_t strsize); //Visual C++ 권장함수	문자열 str을 모두 대문자로 변환하고 변환한 문자열을 반환하므로 str은 상수이면 오류가 발생하며, errno_t는 정수형의 오류번호이며, size_t도 정수형으로 strsize는 str의 길이
char * strpbrk(const char * str, const char * charset);	앞의 문자열 str에서 뒤 문자열 charset에 포함된 문자가 나타나는 처음 위치를 찾아 그 주소값을 반환하며, 만일 찾지 못하면 NULL 포인터를 반환
char * strstr(const char * str, const char * strsearch);	앞의 문자열 str에서 뒤 문자열 strsearch이 나타나는 처음 위치를 찾아 그 주소값을 반환하며, 만일 찾지 못하면 NULL 포인터를 반환
char * strchr(const char * str, char ch);	앞의 문자열 str에서 뒤 문자 ch가 나타나는 처음 위치를 찾아 그 주소값을 반환하며, 만일 찾지 못하면 NULL 포인터를 반환

### 예제코드

```

1 // file: strfun.c
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <string.h>
5
6 int main(void)
7 {
8     char str[] = "JAVA 2017 go c#";
9     printf("%d\n", strlen("java")); //java의 길이: 4
10    printf("%s, ", _strlwr(str));    //모두 소문자로 변환
11    printf("%s\n", _strupr(str));    //모두 대문자로 변환
12
13    //문자열 VA가 시작되는 포인터 반환: VA 2013 GO C#
14    printf("%s, ", strstr(str, "VA"));
15    //문자 A가 처음 나타나는 포인터 반환: AVA 2013 GO C#
16    printf("%s\n", strchr(str, 'A'));
17
18    return 0;
19 }
20
21

```

Microsoft Visual Studio 디버그 콘솔

```

4
java 2017 go c#, JAVA 2017 GO C#
VA 2017 GO C#, AVA 2017 GO C#

C:\Users\hegoc\source\repos\Ch11\Debug\Ch11.exe(14364 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

```



## LAB) 문자열을 역순으로 저장하는 함수 reverse() 구현

```
1 // strreverse.c
2 #include <stdio.h>
3 #include <string.h>
4
5 void reverse(char str[]);
6
7 int main(void)
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 }
```

Microsoft Visual Studio 디버그 콘솔

C:\Programing!  
g:\mmarorP C

C:\Users\Whego\source\repos\Ch11\Debug\Ch11.exe(4148 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 6. 문자 포인터 배열과 이차원 문자 배열

### 가. 문자 포인터 배열

- 1) 여러 개의 문자열을 처리하는 하나의 방법은 문자 포인터 배열을 이용하는 방법.
- 2) 또 다른 방법으로는 문자의 이차원 배열을 이용하는 방법
- 3)

예제코드

```
1 // file: strarray.c
2 #include <stdio.h>
3
4 int main(void)
5 {
6     char* pa[] = { "JAVA", "C#", "C++" };
7     char ca[][5] = { "JAVA", "C#", "C++" };
8
9     //각각의 3개 문자열 출력
10    //pa[0][2] = '\v'; //실행 오류 발생
11    //ca[0][2] = '\v'; //수정 가능
12    printf("%s ", pa[0]); printf("%s ", pa[1]); printf("%s\n", pa[2]);
13    printf("%s ", ca[0]); printf("%s ", ca[1]); printf("%s\n", ca[2]);
14
15    //문자 출력
16    printf("%c %c %c\n", pa[0][1], pa[1][1], pa[2][1]);
17    printf("%c %c %c\n", ca[0][1], ca[1][1], ca[2][1]);
18
19    return 0;
20 }
```

Microsoft Visual Studio 디버그 콘솔

JAVA C# C++  
A # +  
A # +

C:\Users\Whego\source\repos\Ch11\Debug\Ch11.exe(15288 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## LAB) 여러 문자열 처리

```

1 // file: strprocess.c
2 #include <stdio.h>
3
4 int main(void)
5 {
6     char str1[] = "JAVA";
7     char str2[] = "C#";
8     char str3[] = "C++";
9
10    char* pstr[] = { str1, str2, str3 };
11
12    //각각의 3개 문자열 출력
13    printf("%s ", pstr[0]);
14    printf("%s ", pstr[1]);
15    printf("%s\n", pstr[2]);
16
17    //문자 출력
18    printf("%c %c %c\n", str1[0], str2[1], str3[2]);
19    printf("%c %c %c\n", pstr[0][1], pstr[1][1], pstr[2][1]);
20
21    return 0;
22 }
23
Microsoft Visual Studio 디버그 콘솔
C:\Users\whnegoo\source\repos\Ch11\Debug\Ch11.exe(13592 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

```

# <2주차 과제>

### 과제정보

과제명	[과제2]
제출기간	2020-03-12 (00:00)~2020-03-29 (23:59)
참고자료	
과제내용	교재 11장 프로그래밍 연습 1에서 10까지 중 5문제(학번의 홀짝에 따라 홀수 짝수 번호 선택) 구현, 코딩과 결과를 PDF 파일로 가상 강좌 [과제2]에 제출

### 제출정보

과제설명	PE 20173842 한
파일	PE 20173842 한근희 C어플리케이션구현 과제 2주차.pdf
점수	미평가
제출일시	2020-03-29 19:03:25

## 프로그래밍 연습 2번문제

```
second (전역 범위)
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <string.h>
4
5  void mystrcat(char s1[], const char s2[]);
6
7  int main(void)
8  {
9      char s1[50] = "C ";
10
11     mystrcat(s1, "programming language");
12     printf("%s\n", s1);
13
14     return 0;
15 }
16
17 void mystrcat(char s1[], const char s2[])
18 {
19     int a = strlen(s1);
20     for (int i = 0; s2[i] != '\0'; i++) {
21         s1[a] = s2[i];
22         a++;
23     }
24 }
25
```

## 2번문제 결과

Microsoft Visual Studio 디버그 콘솔

C programming language

C:\Users\whagoo\source\repos\second\Debug\second.exe(8296 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

#### 프로그래밍 연습 4번문제

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <string.h>
4
5  void delchar(char str[], const char ch);
6
7  int main(void)
8  {
9      char str[20];
10     char ch = 'a';
11     strcpy(str, "java");
12     delchar(str, ch);
13
14 }
15
16 void delchar(char str[], const char ch)
17 {
18     int count = 0;
19
20     while (str[count] != NULL) {
21         if (str[count] == ch)
22             for (int i = count; str[i] != NULL; i++) {
23                 str[i] = str[i + 1];
24             }
25         count++;
26     }
27     printf("%s\n", str);
28 }
```

#### 4번문제 결과

```
jv
C:\Users\hegoo\source\repos\second\Debug\second.exe(14632 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

### 프로그래밍 연습 6번문제

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <string.h>
4
5
6  int main(void)
7  {
8      char text[25];
9
10     printf("한 단어를 입력하세요. -> ");
11     gets_s(text, sizeof(text));
12
13     int size = strlen(text);
14
15     printf("입력한 단어를 반대로 출력합니다. -> ");
16     for (int i = size; i >= 0; i--) {
17         printf("%c", text[i]);
18     }
19     puts("");
20
21     return 0;
22 }
23
```

### 6번문제 결과

```
한 단어를 입력하세요. -> programming
입력한 단어를 반대로 출력합니다. -> gnimmargorp
C:\Users\hegoon\source\repos\second\Debug\second.exe(9560 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

## 프로그래밍 연습 8번문제

```

1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <string.h>
4
5
6  int main(void)
7  {
8      char text[50];
9      printf("한 줄의 문장을 입력하세요. ->\n");
10     gets_s(text, sizeof(text));
11
12     printf("입력한 각각의 단어를 반대로 출력합니다. ->\n");
13     char* t = strtok(text, " ");
14
15     while (t != NULL) {
16         int size = strlen(t) - 1;
17         for (size; size >= 0; size--) {
18             printf("%c", t[size]);
19         }
20         printf(" ");
21         t = strtok(NULL, " ");
22     }
23 }
24

```

## 8번문제 정답

```

한 줄의 문장을 입력하세요. ->
I've compiled with c++ powerpoint presentation
입력한 각각의 단어를 반대로 출력합니다. ->
ev'! delipmoc htiw ++c tnioprewop noitatneserp
C:\Users\hegoo\source\repos\second\Debug\second.exe(15082 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

```

## 프로그래밍 연습 10번문제

```

1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <string.h>
4  #include <stdlib.h>
5
6  int toint(const char *str);
7
8  int main(void)
9  {
10     char num[100];
11
12     printf("정수를 하나 입력하세요, ->");
13     gets(num);
14
15     printf("먼저 함수 atoi()를 이용한 정수 -> %d\n", atoi(num));
16
17     printf("직접 구현한 함수를 이용한 정수 -> %d\n", toint(num));
18
19     return 0;
20 }
21
22 int toint(const char *str)
23 {
24     int a = 0;
25
26     while (*str != NULL) {
27         a *= 10;
28         a += *str++ - '0';
29     }
30     return a;
31 }

```

## 10번문제 정답

```

정수를 하나 입력하세요, ->76843
먼저 함수 atoi()를 이용한 정수 -> 76843
직접 구현한 함수를 이용한 정수 -> 76843

C:\Users\hnegoo\source\repos\second\Debug\second.exe(9076 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

```

# -변수 유효범위-

## 7. 변수 범위와 지역 변수

### 가. 변수 scope

- 1) 변수의 참조가 유효한 범위를 변수의 유효 범위(scope) 라 함
- 2) 크게 지역 유효 범위 (local scope) 와 전역 유효 범위 (global scope) 로 나뉨  
(지역 유효 범위는 함수 또는 블록 내부에서 선언되어 그 지역에서 변수의 참조가 가능한 범위 임)

### 나. 지역 변수

국내 전용 카드가 지역 변수라면 국내외 사용 카드는 전역 변수임

- 1) 지역변수는 함수 또는 블록에서 선언된 변수  
(지역변수는 선언 문장 이후에 함수나 블록의 내부에서만 사용 가능)  
(함수의 매개변수도 함수 전체에서 사용 가능한 지역변수와 같음)  
(지역 변수는 선언 후 초기화하지 않으면 쓰레기 값이 저장됨)
- 2) 스택 :지역변수가 할당되는 메모리 영역
- 3) 지역변수는 선언된 부분에서 자동으로 생성되고 함수나 블록이 종료되는 순간 자동적으로 메모리에서 제거 → 이를 이유로 자동변수 라고도 함

#### 예제코드

```
1 // file: localvar.c
2 #include <stdio.h>
3
4 void sub(int param);
5
6 int main(void)
7 {
8     //지역변수 선언
9     auto int n = 10;
10    printf("%d\n", n);
11
12    //m, sum은 for 문 내부의 블록 지역변수
13    for (int m = 0, sum = 0; m < 3; m++)
14    {
15        sum += m;
16        printf("%t%d %d\n", m, sum);
17    }
18
19    printf("%d\n", n); //n 참조 가능
20    //printf("%d %d\n", m, sum); //m, sum 참조 불가능
21
22    //함수호출
23    sub(20);
24
25    return 0;
26
27    //매개변수인 param도 지역 변수와 같이 사용
28    void sub(int param)
29    {
30        //지역변수 local
31        auto int local = 100;
32        printf("%t%d %d\n", param, local); //param과 local 참조 가능
33        //printf("%d\n", n); //n 참조 불가능
34    }
```



## 8. 전역 변수와 extern

### 가. 전역변수

- 1) 전역변수 : 함수 외부에서 선언되는 변수
- 2) 일반적으로 프로젝트의 모든 함수나 블록에서 참조 가능 aka. 외부변수
- 3) 키워드 extern을 사용하여 전역변수임을 선언 가능. 이는 문장 맨 앞에 extern을 넣음
- 4) 전역변수에 예상하지 못한 값이 저장된다면 프로그램 어느 부분이 잘못된지 파악 난해

예제 코드

```
Ch11 (전역 범위)
1 // file: globalvar.c
2 #include <stdio.h>
3
4 double getArea(double);
5 double getCircum(double);
6
7 //전역변수 선언
8 double PI = 3.14;
9 int gi;
10
11 int main(void)
12 {
13     //지역변수 선언
14     double r = 5.87;
15     //전역변수 PI와 같은 이름의 지역변수 선언
16     const double PI = 3.141592;
17
18     printf("면적: %.2f\n", getArea(r));
19     printf("둘레1: %.2f\n", 2 * PI * r);
20     printf("둘레2: %.2f\n", getCircum(r));
21     printf("PI: %f\n", PI); //지역변수 PI 참조
22     printf("gi: %d\n", gi); //전역변수 gi 기본 값
23
24     return 0;
25 }
26
27 double getArea(double r)
28 {
29     return r * r * PI; //전역변수 PI 참조
30 }

Ch11 (전역 범위)
1 // circumference.c
2 //이미 외부에서 선언된 전역변수임을 알리는 선언
3 extern double PI;
4
5 double getCircum(double r)
6 {
7     //extern double PI; //함수 내부에서만 참조 가능
8     return 2 * r * PI; //전역변수 PI 참조
9 }
10

Microsoft Visual Studio 디버그 콘솔
면적: 108.19
둘레1: 36.88
둘레2: 36.86
PI: 3.141592
gi: 0
C:\Users\hegoc\source\repos\Ch11\Debug\Ch11.exe(15952 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

## LAB) 피보나치 수의 출력

```
Ch11 (전역 범위)
5 //전역변수
6 int count;
7 //함수원형
8 void fibonacci(int prev_number, int number);
9
10 int main(void)
11 {
12     //자동 지역변수
13     auto prev_number = 0, number = 1;
14     printf("피보나치를 몇 개 구할까요?(3 이상) >> ");
15     //전역변수를 표준입력으로 저장
16     scanf("%d", &count);
17     if (count <= 2)
18         return 0;
19
20     printf("1 ");
21     fibonacci(prev_number, number);
22     printf("\n");
23 }
24 void fibonacci(int prev_number, int number)
25 {
26     //정적 지역변수 i
27     static int i = 1;
28     //전역변수 count와 함수의 정적 지역변수를 비교
29     while (i++ < count)
30     {
31         //지역변수
32         int next_num = prev_number + number;
33         prev_number = number;
34         number = next_num;
35         printf("%d ", next_num);
36         fibonacci(prev_number, number);
37     }
38 }
```

Microsoft Visual Studio 디버그 콘솔

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

C:\Users\megoo\source\repos\Ch11\Debug\Ch11.exe(8908 프로세스)이(가) 0  
코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 9. 기억부류와 레지스터 변수

### 가. auto, register, static, extern

- 1) 변수는 4가지의 기억부류인 auto, register, static, extern 에 따라 할당되는 메모리 영역이 결정되고 메모리의 할당과 제거 시기가 결정됨
- 2) 기억부류 auto와 register는 지역변수에만 이용 가능
- 3) static은 지역과 전역 모든 변수에 이용 가능
- 4) extern은 전역변수에만 이용 가능

표 12-1 기억부류 종류와 유효 범위

기억부류 종류	전역	지역
auto	×	○
register	×	○
static	○	○
extern	○	×

- 5) 키워드 extern을 제외하고 나머지 3개의 기억부류의 변수선언에서 초기값 저장 가능

### 나. 키워드 register

- 1) 레지스터 변수 : 변수의 저장공간이 일반 메모리가 아니라 내부의 레지스터에 할당됨
- 2) 레지스터 변수는 키워드 register를 자료형 앞에 넣어 선언
- 3) 레지스터 변수는 일반 메모리에 할당되는 변수가 아니므로 주소연산자 & 사용 불가
- 4) 레지스터 변수는 처리 속도를 증가시키려는 변수에 이용 (ex : 반복문 횟수 제어 변수)

#### 예제코드

```
Ch11 (전역 범위)
1 // file: registervar.c
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4
5 int main(void)
6 {
7     //레지스터 지역변수 선언
8     register int sum = 0;
9
10    //메모리에 저장되는 일반 지역변수 선언
11    int max;
12    printf("양의 정수 입력 >> ");
13    scanf("%d", &max);
14
15    //레지스터 블록 지역변수 선언
16    for (register int count = 1; count <= max; count++)
17        sum += count;
18
19    printf("합: %d\n", sum);
20
21    return 0;
22
23
```

Microsoft Visual Studio 디버그 콘솔

양의 정수 입력 >> 8  
합: 36

C:\Users\hegoc\source\repos\Ch11\Debug\Ch11.exe(14572 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 10. 정적 변수

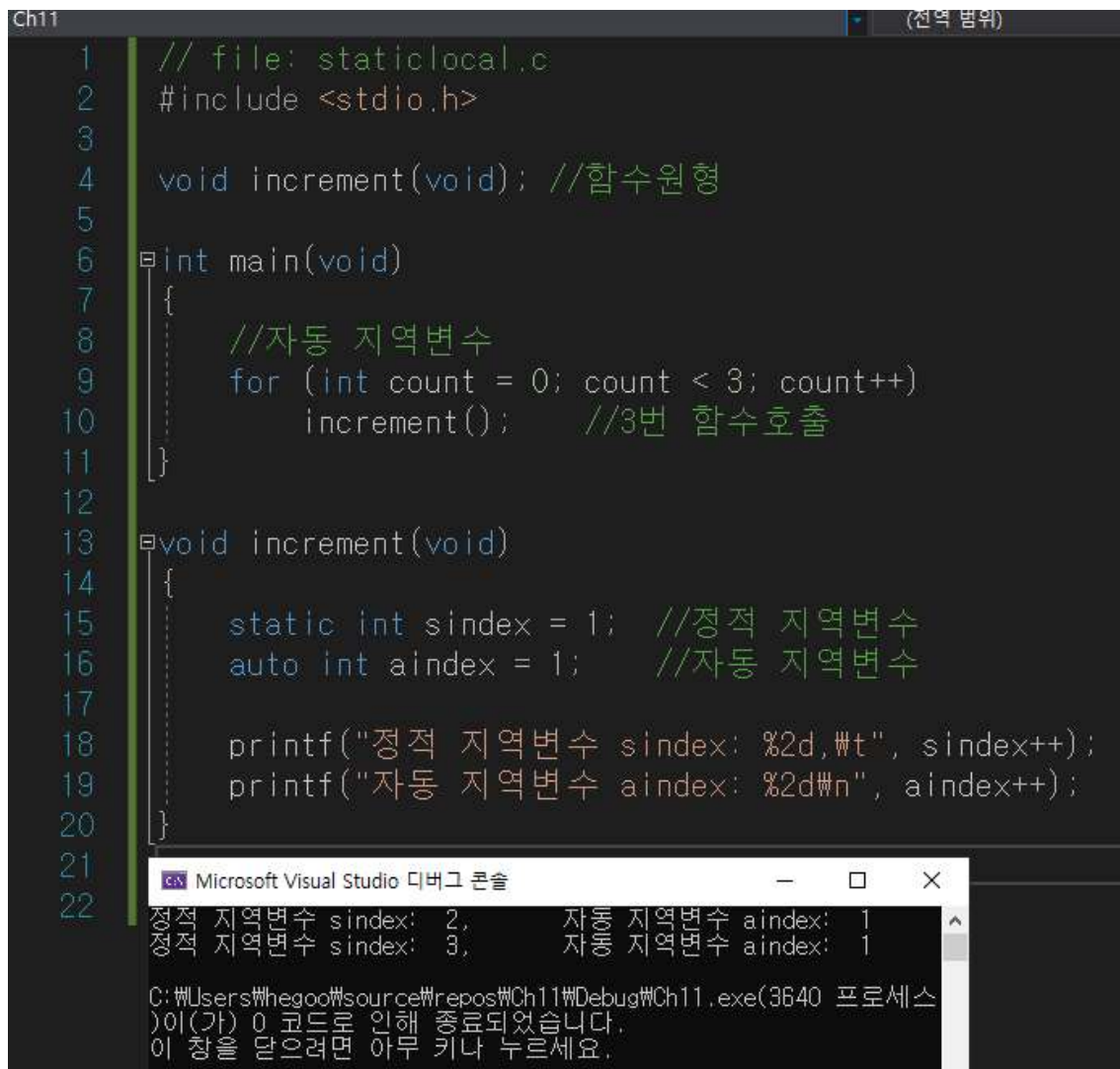
### 가. 키워드 static

- 1) 변수 선언에서 자료형 앞에 키워드 static을 넣어 정적변수를 선언 가능
- 2) 정적변수는 초기값을 지정하지 않으면 자동으로 자료형에 따라 0이나 '\0' 또는 NULL 값이 저장됨

### 나. 정적 지역변수

- 1) 정적 지역변수 : 함수나 블록에서 정적으로 선언되는 변수
- 2) 함수나 블록을 종료해도 메모리에서 제거되지 않고 계속 메모리에 유지, 관리됨

예제코드



```
Ch11 (선택 범위)
1 // file: staticlocal.c
2 #include <stdio.h>
3
4 void increment(void); //함수원형
5
6 int main(void)
7 {
8     //자동 지역변수
9     for (int count = 0; count < 3; count++)
10         increment(); //3번 함수호출
11 }
12
13 void increment(void)
14 {
15     static int sindex = 1; //정적 지역변수
16     auto int aindex = 1; //자동 지역변수
17
18     printf("정적 지역변수 sindex: %2d,\t", sindex++);
19     printf("자동 지역변수 aindex: %2d\n", aindex++);
20 }
21
22
```

Microsoft Visual Studio 디버그 콘솔

```
정적 지역변수 sindex: 2,      자동 지역변수 aindex: 1
정적 지역변수 sindex: 3,      자동 지역변수 aindex: 1

C:\Users\hegooo\source\repos\Ch11\Debug\Ch11.exe(3640 프로세스)
이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

## 다. 정적 전역변수

- 1) 정적 전역변수 : 함수 외부에서 정적으로 선언되는 변수
- 2) 선언된 파일 내부에서만 참조가 가능한 변수
- 3) 프로그램이 크고 복잡하면 전역변수의 사용은 원하지 않는 전역변수의 수정과 같은 부작용 위험성 존재

예제코드

```
Ch11 (전역 범위)
1 // file: staticvar.c
2 #include <stdio.h>
3
4 //정적 전역변수 선언
5 static int svar;
6 //전역변수 선언
7 int gvar;
8
9 //함수 원형
10 void increment();
11 void testglobal();
12 //void teststatic();
13
14 int main(void)
15 {
16     for (int count = 1; count <= 5; count++)
17         increment();
18     printf("함수 increment()가 총 %d번 호출되었습니다.\n", svar);
19
20     testglobal();
21     printf("전역 변수: %d\n", gvar);
22     //teststatic();
23 }
24
25 //함수 구현
26 void increment()
27 {
28     svar++;
29 }
30
```

```
Ch11 (전역 범위)
1 // file: gfunc.c
2
3 void teststatic()
4 {
5     //정적 전역변수는 선언 및 사용 불가능
6     //extern svar;
7     //svar = 5;
8 }
9
10 void testglobal()
11 {
12     //전역변수는 선언 및 사용 가능
13     extern gvar;
14     gvar = 10;
15 }
16
```

Microsoft Visual Studio 디버그 콘솔

함수 increment()가 총 5번 호출되었습니다.  
전역 변수: 10

C:\Users\hegoc\source\repos\Ch11\Debug\Ch11.exe(9652 프로세스)  
(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## LAB) 지역변수와 정적변수의 사용

```
Ch11
1 //file: static.c
2 #include <stdio.h>
3
4 void process();
5
6 int main()
7 {
8     process();
9     process();
10    process();
11
12    return 0;
13 }
14
15 void process()
16 {
17     //정적 변수
18     static int sx;
19     //지역 변수
20     int x = 1;
21
22     printf("%d %d\n", x, sx);
23
24     x += 3;
25     sx += x + 3;
26 }
27
```

짧은 느낌 : 상대적으로 진도의 진행이 많아 습득할 지식이 많았던  
주. strcat 등의 문자열 관련 함수의 반복 습득이 중요할 듯

# <3주차 3월 30일자 수업>

## 11. 메모리 영역

### 가. 데이터, 스택, 힙 영역

- 1) 메인메모리의 영역은 프로그램 실행 과정에서 데이터, 힙, 스택 영역으로 나뉨
- 2) 메모리 영역은 변수의 유효범위와 생존기간에 결정적 역할
- 3) 변수는 기억부류에 따라 할당되는 메모리 공간이 달라짐
- 4) 데이터 영역 : 전역변수와 정적변수가 할당되는 저장공간
- 5) 힙 영역 : 동적 할당 되는 변수가 할당되는 공간
- 6) 스택 영역 : 함수 호출에 의한 형식 매개변수 그리고 함수 내부의 지역변수가 할당되는 공간 → 메모리 주소가 높은 값에서 낮은 값으로 저장장소가 할당 됨

### LAB) 은행계좌의 입출금 구현

```
1 // file: bank.c
2 #include <stdio.h>
3
4 //전역변수
5 int total = 10000;
6
7 //입금 함수원형
8 void save(int);
9 //출금 함수원형
10 void withdraw(int);
11
12 int main(void)
13 {
14     printf("입금액  출금액  총입금액  총출금액  잔고\n");
15     printf("=====\n");
16     printf("%4d\n", total);
17     save(50000);
18     withdraw(30000);
19     save(60000);
20     withdraw(20000);
21     printf("=====\n");
22
23     return 0;
24 }
```

```
26 //입금액을 매개변수로 사용
27 void save(int money)
28 {
29     //총입금액이 저장되는 정적 지역변수
30     static int amount;
31     total += money;
32     amount += money;
33     printf("%7d %17d %20d\n", money, amount, total);
34 }
35
36 //출금액을 매개변수로 사용
37 void withdraw(int money)
38 {
39     //총출금액이 저장되는 정적 지역변수
40     static int amount;
41     total -= money;
42     amount += money;
43     printf("%15d %20d %9d\n", money, amount, total);
44 }
45
```

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

C:\Users\hego\source\repos\Ch11\Debug\Ch11.exe(12784 프로세스)이(가) 0 코드를 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.



# <3주차 과제>

과제정보	
과제명	[과제3]
제출기간	2020-03-29 (00:00)~2020-04-05 (23:59)
참고자료	
과제내용	Lab 12-1, 2, 3의 구현.
제출정보	
과제설명	과제설명 내용 없음
파일	PE 20173842 한근희 C어플리케이션구현 3주차 과제.pdf
점수	미평가
제출일시	2020-04-04 18:28:09

짧은 느낌 : 이전에 배운 내용을 복습하며 기초를 다진 주간



# <4주차 4월 6일자 강의>

## -구조체와 공용체-

### 12. 구조체 개념과 정의

#### 가. 구조체 개념

- 1) 선물세트 : 인기가 있거나 관련 있는 상품들을 묶어 하나의 구성 제품으로 만든 것
- 2) 구조체 : 선물세트와 같이 정수, 문자, 실수, 포인터 그리고 이들의 배열 등을 묶어 하나의 자료형으로 이용하는 것
- 3) 연관성이 있는 서로 다른 개별적인 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형을 뜻함
- 4) 구조체는 연관된 멤버로 구성되는 통합 자료형으로 대표적인 유도 자료형
- 5) 유도 자료형 : 기존 자료형으로 새로이 만들어진 자료형

#### 나. 구조체 정의

- 1) 구조체를 사용하려면 먼저 구조체를 정의하여야함
- 2) 구조체를 사용하려면 먼저 구조체를 만들 구조체 틀을 정의
- 3) 구조체 정의 방법
  - 가) 키워드 `struct` 다음에 구조체 태그 이름을 기술하고 중괄호를 이용하여 원하는 멤버를 여러 개의 변수로 선언
  - 나) 구조체를 구성하는 하나 하나의 항목을 구조체 멤버 또는 필드라 함
- 4) 구조체 정의는 변수의 선언과는 다른 것으로 변수선언에서 이용될 새로운 구조체 자료형을 정의하는 구문
- 5) 다른 구조체 변수 및 구조체 포인터도 구조체 멤버로 허용됨

### 13. 구조체 변수 선언과 초기화

#### 가. 구조체 변수 선언

- 1) 새로운 자료형 `struct account` 형 변수 `mine`을 선언하려면 = `struct account mine;`

#### 나. 구조체 변수의 초기화

- 1) 초기화 값은 다음과 같이 중괄호 내부에서 구조체의 각 멤버 정의 순서대로 초기값을 쉼표로 구분하여 기술
- 2) 배열과 같이 초기값에 기술되지 않은 멤버값은 자료형에 따라 기본값으로 저장

```
struct 구조체태그이름 변수명 = {초기값1, 초기값2, 초기값3, ...};

struct account
{
    char name[12];        //계좌주이름
    int actnum;           //계좌번호
    double balance;       //잔고
};

struct account mine = {"홍길동", 1001, 300000 };
```

## 다. 구조체의 멤버 접근 연산자 . 와 변수 크기

- 1) 선언된 구조체형 변수는 접근 연산자 .를 이용하여 멤버 참조 가능

```
구조체변수이름.멤버
mine.actnum = 1002;   mine.balance = 300000;
```

그림 13-10 구조체 멤버 접근 연산자

- 2) 실제 구조체의 크기는 멤버의 크기의 합보다 크거나 같음

### 예제코드

```
1 // file: structbasic.c
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <string.h>
5
6 //은행 계좌를 위한 구조체 정의
7 struct account
8 {
9     char name[12]; //계좌주 이름
10    int actnum;     //계좌번호
11    double balance; //잔고
12 };
13
14 int main(void)
15 {
16     //구조체 변수 선언 및 초기화
17     struct account mine = { "홍길동", 1001, 300000 };
18     struct account yours;
19
20     strcpy(yours.name, "이동원");
21     //strcpy_s(yours.name, 12, "이동원"); //가능
22     //yours.name = "이동원"; //오류
23     yours.actnum = 1002;
24     yours.balance = 500000;
25
26     printf("구조체 크기: %d\n", sizeof(mine));
27     printf("%s %d %.2f\n", mine.name, mine.actnum, mine.balance);
28     printf("%s %d %.2f\n", yours.name, yours.actnum, yours.balance);
29
30     return 0;
31 }
32
```

Microsoft Visual Studio 디버그 콘솔

```
구조체 크기: 24
홍길동 1001 300000.00
이동원 1002 500000.00

C:\Users\hngoo\source\repos\Ch11\Debug\Ch11.exe(1324 프로세스)
이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

## 14. 구조체 활용

### 가. 구조체 멤버로 사용되는 구조체

예제코드

```
1 // file: nestedstruct.c
2 #include <stdio.h>
3 #include <string.h>
4
5 //날짜를 위한 구조체
6 struct date
7 {
8     int year;    //년
9     int month;   //월
10    int day;     //일
11 };
12
13 //은행계좌를 위한 구조체
14 struct account
15 {
16     struct date open;    //계좌 개설일자
17     char name[12];       //계좌주 이름
18     int actnum;          //계좌번호
19     double balance;      //잔고
20 };
21
22 int main(void)
23 {
24     struct account me = { { 2018, 3, 9 }, "홍길동", 1001, 300000 };
25
26     printf("구조체 크기: %d\n", sizeof(me));
27     printf("[%d, %d, %d]\n", me.open.year, me.open.month, me.open.day);
28     printf("%s %d %.2f\n", me.name, me.actnum, me.balance);
29 }
30
```

Microsoft Visual Studio 디버그 콘솔

```
구조체 크기: 40
[2018, 3, 9]
홍길동 1001 300000.00

C:\Users\heego\source\repos\Ch11\Debug\Ch11.exe(16880 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

## 나. 구조체 변수의 대입과 동등비교

예제코드

```
Ch11 (전역 범위)
1 // file: structstudent.c
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <string.h>
5
6 int main(void)
7 {
8     //학생을 위한 구조체
9     struct student
10     {
11         int snum;        //학번
12         char* dept;      //학과 이름
13         char name[12];   //학생 이름
14     };
15     struct student hong = { 201800001, "컴퓨터정보공학과", "홍길동" };
16     struct student na = { 201800002 };
17     struct student bae = { 201800003 };
18
19     //학생이름 입력
20     scanf("%s", na.name);
21     //na.name = "나한국"; //오류
22     //scanf("%s", na.dept); //오류
23
24     na.dept = "컴퓨터정보공학과";
25     bae.dept = "기계공학과";
26     memcpy(bae.name, "배상문", 7);
27     strcpy(bae.name, "배상문");
28     strcpy_s(bae.name, 7, "배상문");
29
30     printf("[%d, %s, %s]\n", hong.snum, hong.dept, hong.name);
31     printf("[%d, %s, %s]\n", na.snum, na.dept, na.name);
32     printf("[%d, %s, %s]\n", bae.snum, bae.dept, bae.name);
33
34     struct student one;
35     one = bae;
36     if (one.snum == bae.snum)
37         printf("학번이 %d로 동일합니다.\n", one.snum);
38     //if ( one == bae ) //오류
39     if (one.snum == bae.snum && !strcmp(one.name, bae.name) && !strcmp(one.dept, bae.dept))
40         printf("내용이 같은 구조체입니다.\n");
41
42     return 0;
43 }
```

Microsoft Visual Studio 디버그 콘솔

```
나한국
[201800001, 컴퓨터정보공학과, 홍길동]
[201800002, 컴퓨터정보공학과, 나한국]
[201800003, 기계공학과, 배상문]
학번이 201800003로 동일합니다.
내용이 같은 구조체입니다.
C:\Users\heegoo\source\repos\Ch11\Debug\Ch11.exe(14620 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.
```

## 15. 공용체 활용

### 가. 공용체 개념

- 1) 공용체란 동일한 저장 장소에 여러 자료형을 저장하는 방법
- 2) 공용체를 구성하는 멤버에는 한번에 한 종류만 저장하고 참조 가능
- 3) 즉 공용체는 서로 다른 자료형의 값을 동일한 저장공간에 저장하는 자료형
- 4) 공용체 변수의 크기는 멤버 중 가장 큰 자료형의 크기로 정해짐

가) 공용체의 멤버는 모든 멤버가 동일한 저장공간을 사용하므로 동시에 여러 멤버의 값을 동시에 저장하여 이용할 수는 없으며, 마지막에 저장된 단 하나의 멤버 자료값만을 저장한다.

나) 공용체의 초기화 값은 공용체 정의 시 처음 선언한 멤버의 초기값으로만 저장 가능

### 나. 공용체 멤버 접근

- 1) 공용체 변수로 멤버를 접근하기 위해서는 구조체와 같이 접근연산자 .을 사용

예제코드

```
Ch11 (전역 범위)
1 // file: union.c
2 #include <stdio.h>
3
4 //유니온 구조체를 정의하면서 변수 data1도 선언한 문장
5 union data
6 {
7     char ch;    //문자형
8     int cnt;    //정수형
9     double real; //실수형
10 } data1; //data1은 전역변수
11
12 int main(void)
13 {
14     union data data2 = { 'A' }; //첫 멤버인 char형으로만 초기화 가능
15     //union data data2 = {10,3}; //컴파일 시 경고 발생
16     union data data3 = data2; //다른 변수로 초기화 가능
17
18     printf("%d %d\n", sizeof(union data), sizeof(data3));
19
20     //멤버 ch에 저장
21     data1.ch = 'a';
22     printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
23     //멤버 cnt에 저장
24     data1.cnt = 100;
25     printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
26     //멤버 real에 저장
27     data1.real = 3.156759;
28     printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
29
30     return 0;
31 }
32
```

Microsoft Visual Studio 디버그 콘솔

```
8 8
a 97 0.000000
d 100 0.000000
N -590162866 3.156759
```

C:\Users\hhegoof\source\repos\Ch11\Debug\Ch11.exe(16300 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## LAB) 도시의 이름과 위치를 표현하는 구조체

```
Ch11
1 // file: structcity.c
2 #include <stdio.h>
3 #include <string.h>
4
5 //지구 위치 구조체
6 struct position
7 {
8     double latitude; //위도
9     double longitude; //경도
10 };
11
12 int main(void)
13 {
14     //도시 정보 구조체
15     struct city
16     {
17         char* name; //이름
18         struct position place; //위치
19     };
20     struct city seoul, newyork;
21
22     seoul.name = "서울";
23     seoul.place.latitude = 37.33;
24     seoul.place.longitude = 126.58;
25
26     newyork.name = "뉴욕";
27     newyork.place.latitude = 40.8;
28     newyork.place.longitude = 73.9;
```

```
30     printf("[%s] 위도= %.1f 경도= %.1f\n",
31           seoul.name, seoul.place.latitude, seoul.place.longitude);
32     printf("[%s] 위도= %.1f 경도= %.1f\n",
33           newyork.name, newyork.place.latitude, newyork.place.longitude);
34
35     return 0;
36 }
37
```

Microsoft Visual Studio 디버그 콘솔

[서울] 위도= 37.3 경도= 126.6  
[뉴욕] 위도= 40.8 경도= 73.9

C:\Users\hegoo\source\repos\Ch11\Debug\Ch11.exe(16612 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.



## 16. 자료형 재정의 typedef

### 가. typedef 구문

- 1) typedef : 사용되는 자료 유형을 다른 자료형 이름으로 재정의 할수 있도록하는 키워드
- 2) 일반적으로 자료형을 재정의 하는 이유는 프로그램의 시스템 간 호환성과 편의성 위함
- 3) typedef도 일반 변수와 같이 그 사용 범위를 제한함

예제코드

```
Ch11 (전역 범위)
1 // file: typedef.c
2 #include <stdio.h>
3
4 //함수 외부에서 정의된 자료형은 이후 파일에서 사용 가능
5 typedef unsigned int budget;
6
7 int main(void)
8 {
9     //새로운 자료형 budget 사용
10    budget year = 24500000;
11
12    //함수 내부에서 정의된 자료형은 이후 함수내부에서만 사용 가능
13    typedef int profit;
14    //새로운 자료형 profit 사용
15    profit month = 4600000;
16
17    printf("올 예산은 %d, 이달의 이익은 %d 입니다.\n", year, month);
18
19    return 0;
20 }
21
22 void test(void)
23 {
24     //새로운 자료형 budget 사용
25     budget year = 24500000;
26
27     //profit은 이 함수에서는 사용 불가, 오류 발생
28     //profit year;
29 }
30
```

Microsoft Visual Studio 디버그 콘솔

올 예산은 24500000, 이달의 이익은 4600000 입니다.

C:\Users\hegoo\source\repos\Ch11\Debug\Ch11.exe(14316 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 17. 구조체 자료형 재정의

### 가. struct를 생략한 새로운 자료형

- 1) 구조체 struct date가 정의된 상태에서 typedef를 사용하여 구조체 struct date를 date로 재정의 가능

예제코드

```
1 // file: typedefstruct.c
2 #include <stdio.h>
3
4 struct date
5 {
6     int year; //년
7     int month; //월
8     int day; //일
9 };
10
11 //struct date 유형을 간단히 date 형으로 사용하기 위한 구문
12 typedef struct date date;
13
14 int main(void)
15 {
16     //구조체를 정의하면서 바로 자료형 software 로 정의하기 위한 구문
17     typedef struct
18     {
19         char title[30]; //제목
20         char company[30]; //제작회사
21         char kinds[30]; //종류
22         date release; //출시일
23     } software;
24
25     software vs = { "비주얼스튜디오 커뮤니티", "MS", "통합개발환경", { 2018, 8, 29 } };
26
27     printf("제품명: %s\n", vs.title);
28     printf("회사 : %s\n", vs.company);
29     printf("종류 : %s\n", vs.kinds);
30     printf("출시일: %d, %d, %d\n", vs.release.year, vs.release.month, vs.release.day);
31
32     return 0;
33 }
```

### LAB) 영화 정보를 표현하는 구조체

```
1 // file: typemovie.c
2 #include <stdio.h>
3
4 int main(void)
5 {
6     //영화 정보 구조체
7     typedef struct movie
8     {
9         char* title; //영화제목
10        int attendance; //관객수
11    } movie;
12
13    movie assassination;
14
15    assassination.title = "암살";
16    assassination.attendance = 12700000;
17
18    printf("[%s] 관객수: %d\n", assassination.title, assassination.attendance);
19
20    return 0;
21 }
```



## 18. 구조체 포인터

### 가. 포인터 변수 선언

- 1) 포인터는 각각의 자료형 저장 공간의 주소를 저장
- 2) 이와 마찬가지로 구조체 포인터는 구조체의 주소값을 저장 할 수 있는 변수

### 나. 포인터 변수의 구조체 멤버 접근 연산자 ->

- 1) 구조체 포인터 멤버 접근 연산자 -> 는 `p->name` 과 같이 사용함
- 2) 연산식 `p->name`은 포인터`p`가 가리키는 구조체 변수의 멤버 `name`을 접근하는 연산식  
(연산식 `*p.name`은 접근연산자(.)가 간접연산자(\*)보다 우선순위가 빠르므로 `*(p.name)`과 같은 연산식임)

접근 연산식	구조체 변수 <code>os</code> 와 구조체 포인터변수 <code>p</code> 인 경우의 의미
<code>p-&gt;name</code>	포인터 <code>p</code> 가 가리키는 구조체의 멤버 <code>name</code>
<code>(*p).name</code>	포인터 <code>p</code> 가 가리키는 구조체의 멤버 <code>name</code>
<code>*p.name</code>	<code>*(p.name)</code> 이고 <code>p</code> 가 포인터이므로 <code>p.name</code> 은 문법오류가 발생
<code>*os.name</code>	<code>*(os.name)</code> 를 의미하며, 구조체 변수 <code>os</code> 의 멤버 포인터 <code>name</code> 이 가리키는 변수로, 이 경우는 구조체 변수 <code>os</code> 멤버 <code>강좌명</code> 의 첫 문자임, 다만 한글인 경우에는 실행 오류
<code>*p-&gt;name</code>	<code>*(p-&gt;name)</code> 을 의미하며, 포인터 <code>p</code> 이 가리키는 구조체의 멤버 <code>name</code> 이 가리키는 변수로 이 경우는 구조체 포인터 <code>p</code> 이 가리키는 구조체의 멤버 <code>강좌명</code> 의 첫 문자임, 마찬가지로 한글인 경우에는 실행 오류

### 예제코드

```
1 // file: structpointer.c
2 #include <stdio.h>
3
4 struct lecture
5 {
6     char name[20]; //강좌명
7     int type; //강좌구분 0: 교양, 1: 일반선택, 2: 전공필수, 3: 전공선택
8     int credit; //학점
9     int hours; //시수
10 };
11 typedef struct lecture lecture;
12
13 //제목을 위한 문자열
14 char* head[] = { "강좌명", "강좌구분", "학점", "시수" };
15 //강좌 종류를 위한 문자열
16 char* lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
17
18 int main(void)
19 {
20     lecture os = { "운영체제", 2, 3, 3 };
21     lecture c = { "C프로그래밍", 3, 3, 4 };
22     lecture* p = &os;
23
24     printf("구조체크기: %d, 포인터크기: %d\n", sizeof(os), sizeof(p));
25     printf("%10s %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
26     printf("%12s %10s %5d %5d\n", p->name, lectype[p->type], p->credit, p->hours);
27
28     //포인터 변경
29     p = &c;
30     printf("%12s %10s %5d %5d\n", (*p).name, lectype[(*p).type], (*p).credit, (*p).hours);
31     printf("%12c %10s %5d %5d\n", *c.name, lectype[c.type], c.credit, c.hours);
32
33     return 0;
34 }
```

Microsoft Visual Studio 디버그 콘솔

구조체크기: 32, 포인터크기: 4

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

C:\Users\hego\source\repos\Ch11\Debug\Ch11.exe (15172 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 다. 공용체 포인터

- 1) 공용체 변수도 포인터 변수 사용이 가능함

예제코드

```
Ch11 (전역 범위)
1 // file: unionpointer.c
2 #include <stdio.h>
3
4 int main(void)
5 {
6     //유니온 union data 정의
7     union data
8     {
9         char ch;
10        int cnt;
11        double real;
12    };
13
14    //유니온 union data를 다시 자료형 udata로 정의
15    typedef union data udata;
16
17    //udata 형으로 value와 포인터 p 선언
18    udata value, * p;
19
20    p = &value;
21    p->ch = 'a';
22    printf("%c %c\n", p->ch, (*p).ch);
23    p->cnt = 100;
24    printf("%d ", p->cnt);
25    p->real = 3.14;
26    printf("%.2f\n", p->real);
27
28    return 0;
29 }
```

Microsoft Visual Studio 디버그 콘솔

```
a a
100 3.14
```

C:\Users\hegoo\source\repos\Ch11\Debug\Ch11.exe(1716 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 19. 구조체 배열

### 가. 구조체 배열 변수 선언

- 1) 다른 배열과 같이 동일한 구조체 변수가 여러 개 필요하면 구조체 배열을 선언하여 이용 할 수 있음

예제코드

```
1 // file: structarray.c
2 #include <stdio.h>
3
4 struct lecture
5 {
6     char name[20]; //강좌명
7     int type; //강좌구분
8     int credit; //학점
9     int hours; //시수
10 };
11 typedef struct lecture lecture;
12
13 char* lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
14 char* head[] = { "강좌명", "강좌구분", "학점", "시수" };
15
16 int main(void)
17 {
18     //구조체 lecture의 배열 선언 및 초기화
19     lecture course[] = { { "인간과 사회", 0, 2, 2 },
20     { "경제학개론", 1, 3, 3 },
21     { "자료구조", 2, 3, 3 },
22     { "모바일프로그래밍", 2, 3, 4 },
23     { "고급 C프로그래밍", 3, 3, 4 } };
24
25     int arysize = sizeof(course) / sizeof(course[0]);
26
27     printf("배열크기: %d\n\n", arysize);
28     printf("%12s %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
29     printf("=====n");
30
31     for (int i = 0; i < arysize; i++)
32         printf("%16s %10s %5d %5d\n", course[i].name,
33             lectype[course[i].type], course[i].credit, course[i].hours);
34
35     return 0;
36 }
37
```

Microsoft Visual Studio 디버그 콘솔

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\hegoo\source\repos\Ch11\Debug\Ch11.exe(16976 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## LAB) 영화 정보를 표현하는 구조체의 배열

```

1 // file: structmovie.c
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4 #include <string.h>
5
6 int main(void)
7 {
8     //영화 정보 구조체
9     typedef struct movie
10     {
11         char* title;        //영화제목
12         int attendance;     //관객수
13         char director[20]; //감독
14     } movie;
15
16     movie box[] = {
17         { "명량", 17613000, "김한민" },
18         { "국제시장", 14257000, "윤제균" },
19         { "베테랑", 13383000 };
20
21     //영화 베테랑의 감독을 류승완으로 저장
22     strcpy(box[2].director, "류승완");
23
24     printf("   제목      감독      관객수\n");
25     printf("=====n");
26     for (int i = 0; i < 3; i++)
27         printf("[%8s] %6s %d\n",
28             box[i].title, box[i].director, box[i].attendance);
29
30     return 0;
31 }

```



제목	감독	관객수
[ 명량 ]	김한민	17613000
[ 국제시장 ]	윤제균	14257000
[ 베테랑 ]	류승완	13383000

## <4주차 과제>

### 과제정보

과제명	[과제4]
제출기간	2020-03-29 (00:00)~2020-04-12 (23:59)
참고자료	
과제내용	<p><b>일납에세</b></p> <p><b>13-1,</b></p> <p><b>2, 3,</b></p> <p><b>4의</b></p>

### 제출정보

과제설명	과제설명 내용 없음
파일	PE 20173842 한글회 C언어리케이선구현 4주차 과제.pdf
점수	미평가
제출일시	2020-04-10 17:24:32

짧은 느낌 : 생각보다 구조체는 쉽게 다가왔으나 공용체 개념 이해를 좀더 확실히 보완해야 겠다는 생각이 듭

# <5주차 4월 13일자 강의>

## -함수와 포인터 활용-

### 20. 값에 의한 호출과 참조에 의한 호출

#### 가. 함수에서 값의 전달

- 1) C언어는 함수의 인자 전달 방식이 기본적으로 값에 의한 호출 (call by value) 방식임
- 2) 값에 의한 호출 방식이란 함수 호출시 실인자의 값이 형식인자에 복사되어 저장을 의미
- 3) 값에 의한 호출 방식을 사용해서는 함수 외부 변수를 함수 내부에서 수정X 특징 지님

예제코드

```
1 // file: callbyvalue.c
2 #include <stdio.h>
3
4 void increase(int origin, int increment);
5
6 int main(void)
7 {
8     int amount = 10;
9     //amount가 20 증가하지 않음
10    increase(amount, 20);
11    printf("%d\n", amount);
12
13    return 0;
14 }
15
16 void increase(int origin, int increment)
17 {
18     origin += increment;
19 }
20
21
```

Microsoft Visual Studio 디버그 콘솔

10

C:\Users\hegoo\source\repos\Ch11\Debug\Ch11.exe(16812 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

#### 나. 함수에서 주소의 전달

- 1) C언어에서 포인터를 매개변수로 사용하면 함수로 전달될 실인자의 주소를 이용하여 그 변수를 참조 가능
- 2) 이와 같이 함수에서 주소의 호출을 참조에 의한 호출 이라 함

## 예제코드

```
1 // file: callbyreference.c
2 #include <stdio.h>
3
4 void increase(int* origin, int increment);
5
6 int main(void)
7 {
8     int amount = 10;
9     //&amount: amount의 주소로 호출
10    increase(&amount, 20);
11    printf("%d\n", amount);
12
13    return 0;
14 }
15
16 void increase(int* origin, int increment)
17 {
18     /**origin은 origin이 가리키는 변수 자체
19     *origin += increment; //그러므로 origin이 가리키는 변수 값이 20 증가
20 }
21
22 Microsoft Visual Studio 디버그 콘솔
23 30
24 C:\Users\hegoo\source\repos\Ch11\Debug\Ch11.exe(14420 프로세스)이(가) 0 코드로 인해 종료되었습니다.
25 이 창을 닫으려면 아무 키나 누르세요.
```

## 21. 배열의 전달

### 가. 배열 이름으로 전달

- 1) 함수의 매개변수로 배열을 전달하는 것은 배열의 첫 원소를 참조 매개변수로 전달하는 것과 동일함
- 2) 배열 크기에 관계없이 배열 원소의 합을 구하는 함수를 만들려면 배열크기도 하나의 인자로 사용
- 3) 함수원형에서 매개변수는 배열 이름을 생략하고 double [] 과 같이 기술 가능
- 4) 함수호출에서 배열 인자에는 반드시 배열 이름으로 sum (data,5) 와 같이 기술



## 예제 코드

```

1 // file: arrayparameter.c
2 #include <stdio.h>
3
4 #define ARYSIZE 5
5 double sum(double g[], int n); //배열 원소 값을 모두 더하는 함수원형
6
7 int main(void)
8 {
9     //배열 초기화
10    double data[] = { 2.3, 3.4, 4.5, 6.7, 9.2 };
11
12    //배열원소 출력
13    for (int i = 0; i < ARYSIZE; i++)
14        printf("%5.1f", data[i]);
15    puts("");
16
17    //배열 원소 값을 모두 더하는 함수호출
18    printf("합: %5.1f\n", sum(data, ARYSIZE));
19
20    return 0;
21 }
22
23 //배열 원소 값을 모두 더하는 함수정의
24 double sum(double ary[], int n)
25 {
26     double total = 0.0;
27     for (int i = 0; i < n; i++)
28         total += ary[i];
29
30     return total;
31 }

```

Microsoft Visual Studio 디버그 콘솔

```

2.3 3.4 4.5 6.7 9.2
합: 26.1

```

C:\Users\hegoo\source\repos\Ch11\Debug\Ch11.exe(15108 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 나. 다양한 배열원소 참조 방법

```

int i, sum = 0;
int point[] = {95, 88, 76, 54, 85, 33, 65, 78, 99, 82};
int *address = point;
int aryLength = sizeof(point) / sizeof(int);

```

가능	가능	오류
for (i=0; i<aryLength; i++) sum += *(point+i);	for (i=0; i<aryLength; i++) sum += *(address++);	for (i=0; i<aryLength; i++) sum += *(point++);

그림 14-5 간접연산자 \*를 사용한 배열원소의 참조방법

(함수헤더에 int ary[]로 기술하는 것은 int \*ary로 대체 가능)



## 예제 코드

```

1  #include <stdio.h>
2
3  int sumary(int* ary, int SIZE); //int sumary(int ary[], int SIZE)도 가능
4
5  int main(void)
6  {
7      int point[] = { 95, 88, 76, 54, 85, 33, 65, 78, 99, 82 };
8      //배열크기 구하기
9      int aryLength = sizeof(point) / sizeof(int);
10
11     //address는 포인터 변수이며 point는 배열 상수
12     int* address = point;
13     //메인에서 직접 배열 합 구하기
14     int sum = 0;
15     for (int i = 0; i < aryLength; i++)
16         sum += *(point + i);
17     //sum += *(point++); //오류발생
18     //sum += *(address++); //가능
19     printf("메인에서 구한 합은 %d\n", sum);
20
21     //함수호출하여 합 구하기
22     printf("함수sumary() 호출로 구한 합은 %d\n", sumary(point, aryLength));
23     printf("함수sumary() 호출로 구한 합은 %d\n", sumary(&point[0], aryLength));
24     printf("함수sumary() 호출로 구한 합은 %d\n", sumary(address, aryLength));
25
26     return 0;
27 }

```

```

29 int sumary(int* ary, int SIZE) //int sumary(int ary[], int SIZE)도 가능
30 {
31     int sum = 0;
32     for (int i = 0; i < SIZE; i++)
33     {
34         //sum += ary[i]; //가능
35         //sum += *(ary + i); //가능
36         sum += *ary++;
37         //sum += *(ary++); //가능
38     }
39
40     return sum;
41 }

```

Microsoft Visual Studio 디버그 콘솔

```

43 메인에서 구한 합은 755
44 함수sumary() 호출로 구한 합은 755
함수sumary() 호출로 구한 합은 755
함수sumary() 호출로 구한 합은 755

```

C:\Users\hegoo\source\repos\Ch11\Debug\Ch11.exe(16412 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 다. 배열 크기 계산방법

1) 연산자 sizeof를 이용한 식 ( sizeof(배열이름) / sizeof(배열원소) )

예제코드

```
1 // file: arrayfunction
2 #define _CRT_SECURE_NO_WARNINGS
3 #include <stdio.h>
4
5 void readarray(double[], int); //배열 원소값을 모두 표준입력 받는 함수원형
6 void printarray(double[], int); //배열 원소값을 모두 출력하는 함수원형
7 double sum(double[], int); //배열 원소값을 모두 더하는 함수원형
8
9 int main(void)
10 {
11     double data[5];
12     int arraysize = sizeof(data) / sizeof(data[0]);
13
14     printf("실수 5개의 값을 입력하세요. \n");
15     readarray(data, arraysize);
16     printf("\n입력한 자료값은 다음과 같습니다.\n");
17     printarray(data, arraysize);
18     printf("함수에서 구한 합은 %.3f 입니다.\n", sum(data, arraysize));
19
20     return 0;
21 }
22
23 //배열 원소값을 모두 표준입력 받는 함수
24 void readarray(double data[], int n)
25 {
26     for (int i = 0; i < n; i++)
27     {
28         printf("data[%d] = ", i);
29         scanf("%lf", &data[i]); //(data + i)로도 가능
30     }
31     return;
32 }
33
34 //배열 원소값을 모두 출력하는 함수
35 void printarray(double data[], int n)
36 {
37     for (int i = 0; i < n; i++)
38         printf("data[%d] = %.2lf ", i, *(data + i));
39     printf("\n");
40     return;
41 }
42
43 //배열 원소값을 모두 더하는 함수
44 double sum(double data[], int n)
45 {
46     double total = 0;
47     for (int i = 0; i < n; i++)
48         total += data[i]; //*(data + i)
49     return total;
50 }
51
52
```

Microsoft Visual Studio 디버그 콘솔

실수 5개의 값을 입력하세요.

data[0] = 3.22

data[1] = 4.22

data[2] = 5.33

data[3] = 9.63

data[4] = 5.98

입력한 자료값은 다음과 같습니다.

data[0] = 3.22 data[1] = 4.22 data[2] = 5.33 data[3] = 9.63 data[4] = 5.98

함수에서 구한 합은 28.380 입니다.

C:\Users\hogeoo\source\repos\Ch11\Debug\Ch11.exe(4176 프로세스)이(가) 0 코드로 인해 종료되었습니다.

이 창을 닫으려면 아무 키나 누르세요.

## 라. 다차원 배열 전달

- 1) 다차원 배열을 인자로 사용하는 경우, 함수원형과 함수정의의 헤더에서 첫 번째 대괄호 내부의 크기를 제외한 다른 모든 크기는 반드시 기술되어야함

함수 sum()을 호출하려면 배열이름과 함께 행과 열의 수가 필요하다.

- 이차원 배열의 행의 수는 다음과 같이 ( $\text{sizeof}(x) / \text{sizeof}(x[0])$ )로 계산할 수 있다.
- 또한 이차원 배열의 열의 수는 다음과 같이 ( $\text{sizeof}(x[0]) / \text{sizeof}(x[0][0])$ )로 계산할 수 있다.
- 여기서  $\text{sizeof}(x)$ 는 배열 전체의 바이트 수를 나타내며  $\text{sizeof}(x[0])$ 는 1행의 바이트 수,  $\text{sizeof}(x[0][0])$ 은 첫 번째 원소의 바이트 수를 나타낸다.

### 예제코드

```
1 // file: twodarrayfunction.c
2 #include <stdio.h>
3
4 //2차원 배열값을 모두 더하는 함수원형
5 double sum(double data[][3], int, int);
6 //2차원 배열값을 모두 출력하는 함수원형
7 void printarray(double data[][3], int, int);
8
9 int main(void)
10 {
11     //4 x 3 행렬
12     double x[][3] = { { 1, 2, 3 }, { 7, 8, 9 }, { 4, 5, 6 }, { 10, 11, 12 } };
13
14     int rowsize = sizeof(x) / sizeof(x[0]);
15     int colsize = sizeof(x[0]) / sizeof(x[0][0]);
16     printf("2차원 배열의 자료값은 다음과 같습니다.\n");
17     printarray(x, rowsize, colsize);
18     printf("2차원 배열 원소합은 %.3lf 입니다.\n", sum(x, rowsize, colsize));
19
20     return 0;
21 }
22
23 //배열값을 모두 출력하는 함수
24 void printarray(double data[][3], int rowsize, int colsize)
25 {
26     for (int i = 0; i < rowsize; i++)
27     {
28         printf("%d행원소: ", i + 1);
29         for (int j = 0; j < colsize; j++)
30             printf("x[%d][%d] = %.2lf ", i, j, data[i][j]);
31         printf("\n");
32     }
33     printf("\n");
34 }
35
36 //배열값을 모두 더하는 함수
37 double sum(double data[][3], int rowsize, int colsize)
38 {
39     double total = 0;
40     for (int i = 0; i < rowsize; i++)
41         for (int j = 0; j < colsize; j++)
42             total += data[i][j];
43     return total;
44 }
```

Microsoft Visual Studio 디버그 콘솔

2차원 배열의 자료값은 다음과 같습니다.  
1행원소: x[0][0] = 1.00 x[0][1] = 2.00 x[0][2] = 3.00  
2행원소: x[1][0] = 7.00 x[1][1] = 8.00 x[1][2] = 9.00  
3행원소: x[2][0] = 4.00 x[2][1] = 5.00 x[2][2] = 6.00  
4행원소: x[3][0] = 10.00 x[3][1] = 11.00 x[3][2] = 12.00

2차원 배열 원소합은 78.000 입니다.

C:\Users\hegoos\source\repos\Ch11\Debug\Ch11.exe(16048 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## 22. 가변인자

### 가. 가변 인자가 있는 함수머리

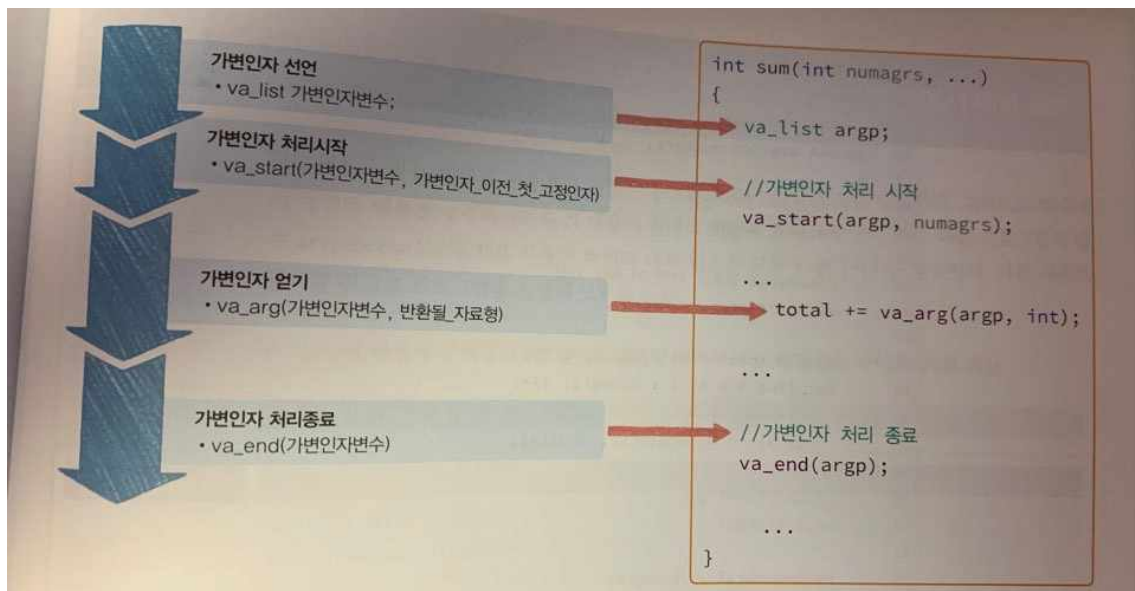
- 1) 함수 printf()를 호출하는 경우를 살펴보면, 출력할 인자의 수와 자료형이 결정되지 않은 채 함수를 호출함
- 2) 출력할 인자의 수와 자료형은 인자 \_Format에 %d 등으로 표현되어 있음
- 3) 함수에서 인자의 수와 자료형이 결정되지 않은 함수 인자 방식을 가변인자 라함

### 나. 가변인자가 있는 함수 구현

- 1) 가변 인자를 구현하려면 4단계가 필요
- 2) 또한 헤더파일 stdarg.h 필요

표 14-1 가변인자 처리를 위한 네 가지 절차

구문	처리 절차	설명
<code>va_list argp;</code>	❶ 가변인자 선언	<code>va_list</code> 로 변수 <code>argp</code> 을 선언
<code>va_start(va_list argp, prevarg)</code>	❷ 가변인자 처리 시작	<code>va_start()</code> 는 첫 번째 인자로 <code>va_list</code> 로 선언된 변수이름 <code>argp</code> 과 두 번째 인자는 가변인자 앞의 고정인자 <code>prevarg</code> 를 지정하여 가변인자 처리 시작
<code>type va_arg(va_list argp, type)</code>	❸ 가변인자 얻기	<code>va_arg()</code> 는 첫번째 인자로 <code>va_start()</code> 로 초기화한 <code>va_list</code> 변수 <code>argp</code> 를 받으며, 두번째 인자로 가변 인자로 전달된 값의 type을 기술
<code>va_end(va_list argp)</code>	❹ 가변인자 처리 종료	<code>va_list</code> 로 선언된 변수이름 <code>argp</code> 의 가변인자 처리 종료



## 예제코드

```

1 // file: vararg.c
2 #include <stdio.h>
3 #include <stdarg.h>
4
5 double avg(int count, ...); //int count 이후는 가변인자 ...
6
7 int main(void)
8 {
9     printf("평균 %.2f\n", avg(5, 1.2, 2.1, 3.6, 4.3, 5.8));
10
11     return 0;
12 }
13
14 //가변인자 ... 시작 전 첫 고정 매개변수는 이후의 가변인자를 처리하는데 필요한 정보를 지정
15 //여기서에서는 가변인자의 수를 지정
16 double avg(int numargs, ...)
17 {
18     //가변인자 변수 선언
19     va_list argp;
20
21     //numargs 이후의 가변인자 처리 시작
22     va_start(argp, numargs);
23
24     double total = 0; //합이 저장될 변수
25     for (int i = 0; i < numargs; i++)
26         //지정하는 double 형으로 가변인자 하나 하나를 반환
27         total += va_arg(argp, double);
28
29     //가변인자 처리 종료
30     va_end(argp);
31
32     return total / numargs;
33 }
34

```

Microsoft Visual Studio 디버그 콘솔

평균 3.40

C:\Users\hegoc\source\repos\Ch11\Debug\Ch11.exe(16204 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

## LAB) 함수에서 배열 활용

```

1 // file: aryprocess.c
2 #include <stdio.h>
3
4 void aryprocess(int* ary, int SIZE);
5
6 int main(void)
7 {
8     int data[] = { 1, 3, 5, 7, 9 };
9
10    int aryLength = sizeof(data) / sizeof(int);
11    aryprocess(data, aryLength);
12    for (int i = 0; i < aryLength; i++)
13        printf("%d ", *(data + i));
14    printf("\n");
15
16    return 0;
17 }
18
19 void aryprocess(int* ary, int SIZE)
20 {
21     for (int i = 0; i < SIZE; i++)
22         (*ary++)++; //(*ary++)++;
23         // ++(*ary++); ++(*ary++); ++*ary++; //동일한 기능
24 }
25

```

Microsoft Visual Studio 디버그 콘솔

2 4 6 8 10

C:\Users\hegoc\source\repos\Ch11\Debug\Ch11.exe(17044 프로세스)이(가) 0 코드로 인해 종료되었습니다.  
이 창을 닫으려면 아무 키나 누르세요.

짧은 느낌 : 이전에 추상적으로만 배웠던 callbyreference 방식 등을 익힐 수 있던 주, 꾸준한 복습이 필요함

감사합니다!