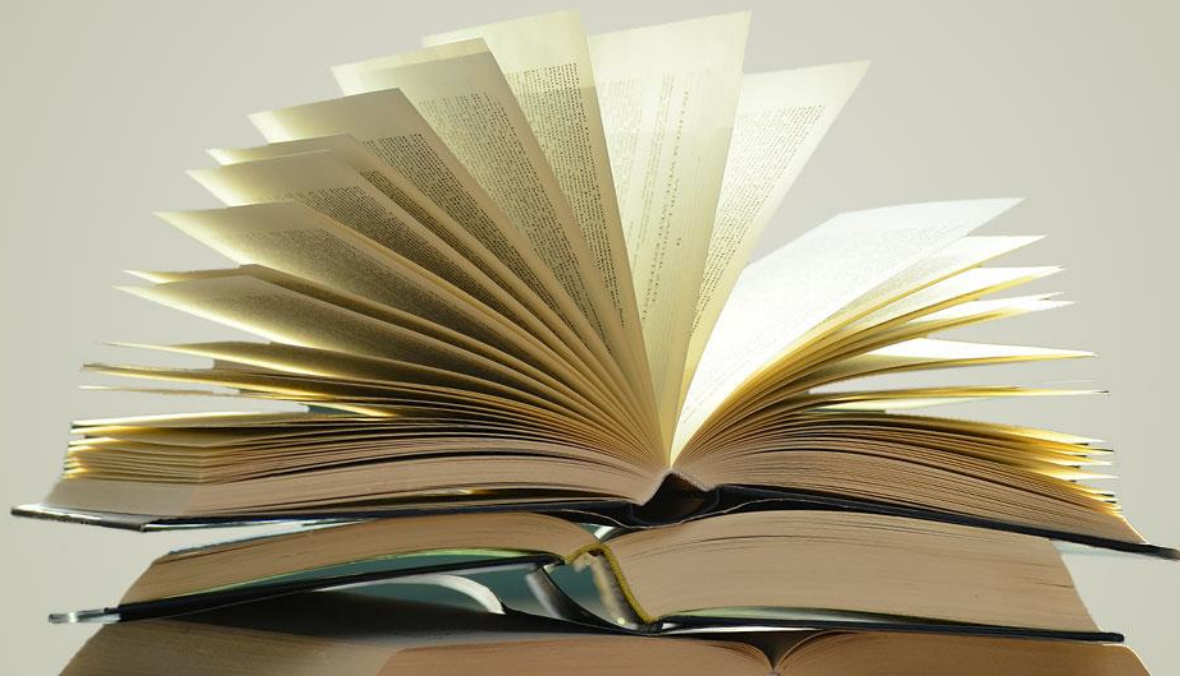


2020 C애플리케이션구현

C포트폴리오

PE반 20161886 강연승



목차

1

머릿말

2

11장 문자와 문자열

3

12장 변수 유효범위

4

13장 구조체와 공용체

5

꼬릿말

강의계획서



2020 학년도 1학기	전공	컴퓨터정보공학과(사회맞춤형 컴퓨팅과정)	학부	컴퓨터공학부
과 목 명	C애플리케이션구현(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	<p>본 과목은 프로그래밍 언어 중 가장 널리 사용되고 있는 C언어를 학습하는 과목으로 C++, JAVA 등과 같은 언어의 기반이 된다. 본 과목에서는 지난 학기에서 배운 시스템프로그래밍 1에 이어 C언어의 기본 구조 및 문법 체계 그리고 응용 프로그래밍 기법 등을 다룬다.</p> <p>C언어에 대한 학습은 Windows상에서 이루어지며, 기본적인 이론 설명 후 실습문제를 프로그래밍하며 숙지하는 형태로 수업이 진행된다.</p>



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 바탕으로 하여 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, 강한일, 이동규	인파니티북스	
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			
2 주차	[2주]			
학습주제	C언어 기초 문법			
목표및 내용	변수와 상수 연산자 l-value와 r-value			
미리읽어오기	교재 1~5장			
과제,시험,기타	수업 중에 제시함			
3 주차	[3주]			
학습주제	조건문			
목표및 내용	6장 조건문 학습			
미리읽어오기	교재 6장			
과제,시험,기타	수업 중에 제시함			

4 주차	[4주]
학습주제	반복문
목표및 내용	7장 반복문 학습
미리읽어오기	교재 7장
과제,시험,기타	수업 중에 제시함
5 주차	[5주]
학습주제	포인터
목표및 내용	8장 포인터 학습 단일포인터 다중포인터 여러가지 포인터
미리읽어오기	교재 8장
과제,시험,기타	수업 중에 제시함
6 주차	[6주]
학습주제	배열
목표및 내용	9장 배열
미리읽어오기	교재 9장
과제,시험,기타	수업 중에 제시함

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

10 주차	[10주]
학습주제	변수 유효범위
목표및 내용	12장 변수 유효범위
미리읽어오기	교재 12장
과제,시험,기타	수업 중에 제시함
11 주차	[11주]
학습주제	구조체
목표및 내용	13장 구조체
미리읽어오기	교재 13장
과제,시험,기타	수업 중에 제시함
12 주차	[12주]
학습주제	함수와 포인터 활용
목표및 내용	14장 함수와 포인터활용
미리읽어오기	교재 14장
과제,시험,기타	수업 중에 제시함

13 주차	[13주]
학습주제	파일처리
목표및 내용	15장 파일처리
미리읽어오기	교재 15장
과제,시험,기타	수업 중에 제시함
14 주차	[14주]
학습주제	향상심화강좌(동적할당)
목표및 내용	16장 동적할당
미리읽어오기	교재 16장
과제,시험,기타	수업 중에 제시함
15 주차	[기말고사]
학습주제	기말고사
목표및 내용	기말고사
미리읽어오기	
과제,시험,기타	..

머릿말

1년 휴학하면서 저만의 시간을 가지고
군대를 다녀와서 3년이라는 공백기가 있었더니
1학년때의 지식을 거의 다 잊고서 복학을 했습니다.

다시 공부하기에 상당한 어려움을 겪는 상황에
코로나19로 인하여 대면수업까지 불가능한 상황에 이르러
공부하기 힘든 상황에서 이 포트폴리오를 하면서
조금이라도 C언어에 대해 이해하고 활용할 수 있는
계기가 되었으면 좋겠습니다.



11장 문자와 문자열



11장 문자와 문자열

문자와 문자열 개념

문자

- 문자는 영어의 알파벳이나 한글의 한 글자를 작은 따옴표로 둘러싸서 'A'와 같이 표기
- 저장공간 크기 1바이트인 자료형 char로 지원
- 작은 따옴표에 의해 표기된 문자를 문자 상수라 한다.
- ex) char ch = 'A';

문자열

- 문자의 모임인 일련의 문자를 문자열이라 한다.
- 문자열은 일련의 문자 앞 뒤로 큰따옴표로 둘러싸서 "java"로 표기
- 문자의 나열인 문자열은 'ABC'처럼 작은 따옴표로 둘러싸도 문자가 될 수 없으며 오류가 발생한다.
- ex) char c[] = "C language";

11장 문자와 문자열

문자와 문자열 선언

- Char형 변수에 문자를 저장
- 문자열을 저장하려면 문자 배열을 사용한다
- 문자열의 마지막을 의미하는 NULL문자'\0'을 마지막에 저장해야 한다. 그러므로 문자열이 저장되는 배열크기는 반드시 저장될 문자 수보다 1이 커야 한다

```
//문자
```

```
char ch = 'A';
```

```
//문자열
```

```
char csharp[3];
```

```
csharp[0] = 'C'; csharp[1] = '#'; csharp[2] = '\0';
```

11장 문자와 문자열

문자와 문자열 선언

- 배열 선언 시 초기화 방법

```
char Java[] = { 'J', 'A', 'V', 'A', '\0' };
```

- 배열 선언 시 큰 따옴표를 사용하여 문자열 상수 바로 대입
- 배열 초기화 시 배열 크기는 지정하지 않는 것이 더 편리
- 크기 지정 시 마지막 문자인 '\0'을 고려해 문자 수 보다 1이 더 크게 배열크기를 지정
- 지정한 배열크기가 (문자수+1)보다 크면 나머지는 '\0'으로 채워진다

```
char c[] = "C language";  
char c[11] = "C language";
```


11장 문자와 문자열

함수 printf()를 사용한 문자와 문자열 출력

- 문자 선언과 출력

```
char ch = 'A';  
printf("%c", ch);
```

- 문자열 선언과 출력

```
char c[] = "C language";  
printf("%s", c);
```

11장 문자와 문자열

함수 printf()를 사용한 문자와 문자열 출력

- 문자열을 구성하는 문자 참조
 - 문자열 상수를 문자 포인터에 저장하는 방식

```
char* java = "java";  
printf("%s ", java);
```

- '\0' 문자에 의한 문자열 분리
 - 함수 printf()에서 %s는 문자 포인터가 가리키는 위치에서 NULL문자 까지 하나의 문자로 인식한다

```
char c[] = "C C++ Java";  
c[5] = '\0';  
printf("%s\n%s\n", c, (c + 6));
```

11장 문자와 문자열

다양한 문자 입출력

- 1) 버퍼처리 함수 `getchar()`
 - 함수 `getchar()`는 문자의 입력에 사용되고 `putchar()`는 출력에 사용된다
 - 문자입력을 위한 함수 `getchar()`는 라인 버퍼링 방식을 사용
- 2) 함수 `getche()`
 - 버퍼를 사용하지 않고 문자 하나를 바로 입력할 수 있는 함수
 - 헤더파일 `conio.h`를 삽입해야 한다
- 3) 함수 `getch()`
 - 문자 입력을 위한 함수 `getch()`는 입력한 문자가 화면에 보이지 않는 특성이 있다

함수	<code>scanf("%c", &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	

11장 문자와 문자열

다양한 문자 입출력

- 1) 버퍼처리 함수 getchar()

```
while ((ch = getchar()) != 'q')  
    putchar(ch);
```

- 2) 함수 getche()

```
while ((ch = _getche()) != 'q')  
    putchar(ch);
```

- 3) 함수 getch()

```
while ((ch = _getch()) != 'q')  
    _putch(ch);
```

11장 문자와 문자열

문자열 입력

1) 문자배열 변수로 scanf()에서 입력

- scanf()는 공백으로 구분되는 하나의 문자열을 입력 받을 수 있다.
- scanf()에서 형식제어문 %s를 통해 입력 받을 수 있고 printf를 통해 출력한다.

```
char name[20];  
printf("%s", "이름입력 >> ");  
scanf("%s", name);  
printf("출력: %s", name);
```

2) gets()와 puts()

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

```
char * gets(char * buffer);  
char * gets_s(char * buffer, size_t sizebuffer);  
int puts(const char * str);
```

함수 printf()와 scanf()는 다양한 입출력에 적합하며, 문자열 입출력 함수 puts()와 gets()는 처리속도가 빠르다는 장점이 있다



11장 문자와 문자열

문자열 관련 함수

- 1) 함수 strcmp()
 - 문자열 비교와 복사, 그리고 문자열 연결 등과 같은 다양한 문자열 처리는 헤더 파일 string.h에 함수원형으로 선언된 라이브러리 함수로 제공
- 2) 함수 strcpy()
 - 함수 strcpy()와 strncpy는 문자열을 복사하는 함수이다. 함수 strcpy()는 앞 인자 문자열 dest에 뒤 인자 문자열 source를 복사한다.
- 3) 함수 strcat()
 - 함수 strcat()는 앞문자열에 뒤 문자열 null 문자까지 연결하여, 앞의 문자열 주소를 반환하는 함수이다.
- 4) 함수 strtok()
 - 함수 strtok()은 문자열에서 구분자인 문자를 여러 개 지정하여 토큰을 추출하는 함수이다.
- 5) 함수 strlen()
 - 함수 strlen()은 NULL 문자를 제외한 문자열 길이를 반환하는 함수이다.
- 6) 함수 strlwr()
 - 인자를 모두 소문자로 변환하여 반환한다.
- 7) 함수 strupr()
 - 인자를 모두 대소문자로 변환하여 반환한다.



11장 문자와 문자열

여러 문자열 처리

1) 문자 포인터 배열

- 여러 개의 문자열을 처리하는 하나의 방법은 문자 포인터 배열을 이용하는 방법이다.

```
char* pa[] = { "JAVA", "C#", "C++" };
```

2) 이차원 문자 배열

- 여러 개의 문자열을 처리하는 다른 방법은 문자의 이차원 배열을 이용하는 방법이다.

```
char* ca[][5] = { "JAVA", "C#", "C++" };
```

3) 명령행 인자

- 명령행에서 입력하는 문자열을 프로그램으로 전달하는 방법이 명령행 인자를 사용하는 방법이다.
- 실행 프로그램 이름도 하나의 명령행 인자의 포함된다.

12장 변수 유효범위



12장 변수 유효범위

변수 범위와 지역변수

1) 변수 scope

- 변수의 참조가 유효한 범위를 변수의 유효범위(scope)라 한다.
- 지역 유효 범위와 전역 유효 범위로 나눌 수 있다
- 지역 유효 범위는 함수 또는 블록 내부에서 선언되어 그 지역에서 변수의 참조가 가능한 범위 이다

2) 지역변수

- 국내 전용 카드가 지역 변수라면 국내외 사용 카드는 전역 변수라 한다.
- 지역 변수는 함수 또는 블록에서 선언된 변수이다.
- 함수나 블록에서 지역변수는 선언 문장 이후에 함수나 블록의 내부에서만 사용이 가능하다.
- 함수의 매개 변수도 함수 전체에서 사용가능한 지역변수와 같다
- 지역변수는 선언 후 초기화하지 않으면 쓰레기 값이 저장되므로 주의해야 한다
- 지역변수가 할당되는 메모리 영역을 스택(stack)이라 한다
- 지역변수는 선언된 부분에서 자동으로 생성되고 함수나 블록이 종료되는 순간메모리에서 자동으로 제거 된다
- 지역변수는 자동변수라 한다

12장 변수 유효범위

전역 변수와 extern

- 1) 전역변수
 - 전역변수는 함수 외부에서 선언되는 변수이다
 - 전역변수는 외부 변수라고도 부른다
 - 전역변수는 일반적으로 프로젝트의 모든 함수나 블록에서 참조할 수 있다
- 2) 전역변수 장단점
 - 전역변수는 어디서나 수정할 수 있으므로 사용이 편한 장점이 있다
 - 전역변수에 예상하지 못한 값이 저장된다면 프로그램 어느 부분에서 수정되었는지 알기 어려운 단점이 있다

```
//전역변수 선언
double PI = 3.14;

int main(void) {
    //지역변수 선언
    double r = 5.87;
    //전역변수 PI와 같은 이름의 지역변수 선언
    const double PI = 3.141592;

    printf("PI: %f", PI); //지역변수 PI 참조
    return 0;
}

double getArea(double r) {
    return r * r * PI; //전역변수 PI참조
```

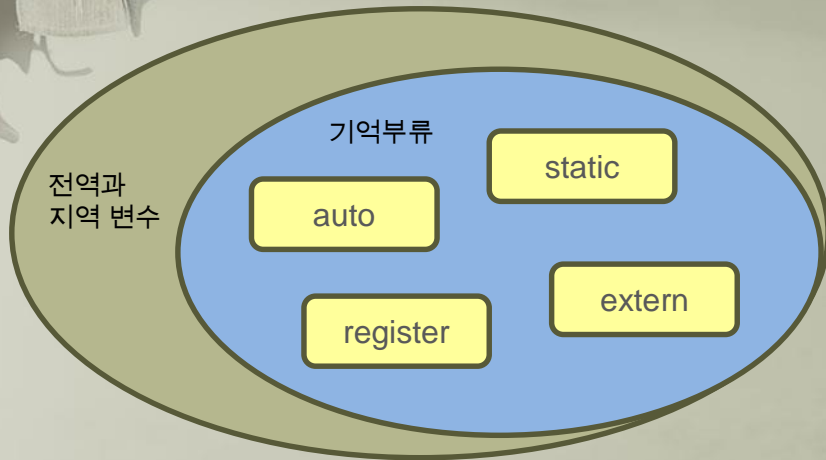
```
//이미 선언된 전역변수 선언
extern double PI;

double getCircum(double r) {
    //전역변수 PI 참조
    return 2 * r * PI;
}
```


12장 변수 유효범위

기억부류

- 1) auto, register, static, extern
- 4가지 기억부류인 auto, register, static, extern 에 따라 할당되는 메모리 영역이 결정되고 메모리의 할당과 제거 시기가 결정된다
 - 기억부류 auto와 register는 지역변수에만 이용 가능하고 static은 지역과 전역모든 변수에 시용 가능하다. extern은 전역변수에만 사용이 가능하다
 - 키워드 extern을 제외하고 나머지 3개의 기억부류의 변수선언에서 초기값을 저장할 수 있다



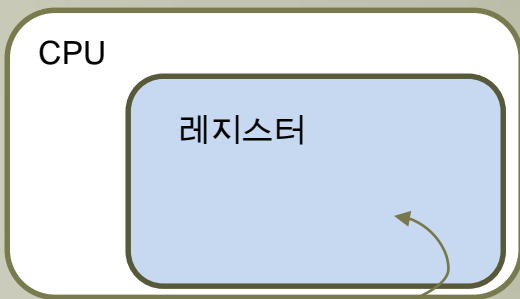
기억부류 종류	전역	지역
auto	X	O
register	X	O
static	O	O
extern	O	X

12장 변수 유효범위

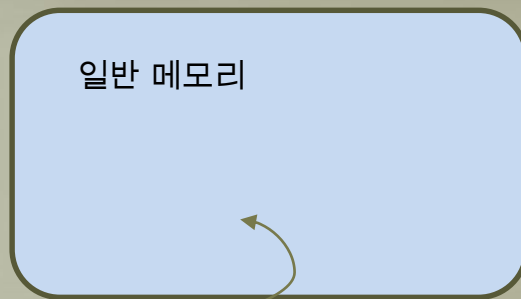
레지스터 변수

1) 키워드 register

- 레지스터 변수는 변수의 저장공간이 일반 메모리가 아니라 CPU내부의 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



register int **count**;



int **data**;

A person's hand holding a book and a brown leather bag.

12장 변수 유효범위

정적변수

- 1) 키워드 static
 - 변수 선언에서 자료형 앞에 키워드 static을 넣어 정적변수를 선언할 수 있다
 - 정적변수는 초기값을 지정하지 않으면 자동으로 자료형에 따라 0이나 '\0'또는 NULL값이 저장된다
- 2) 정적 지역변수
 - 함수나 블록에서 정적으로 선언되는 변수가 정적 지역변수이다.
 - 정적 지역변수는 함수나 블록을 종료해도 메모리에서 제거되지 않고 계속 메모리에 유지 관리되는 특성이 있다.
- 3) 정적 전역변수
 - 함수 외부에서 정적으로 선언되는 변수가 정적 전역변수이다.
 - 정적 전역변수는 선언된 파일 내부에서만 참조가 가능한 변수이다.
 - 프로그램이 크고 복잡하면 전역변수의 사용은 원하지 않는 전역변수의 수정과 같은 부작용의 위험성이 항상 존재한다.

12장 변수 유효범위

메모리 영역

1) 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 하며, 변수는 기억부류에 따라 할당되는 메모리 공간이 달라진다.
- 힙 영역은 동적 할당 되는 변수가 할당되는 저장공간이다.
- 스택 영역은 함수 호출에 의한 형식 매개변수 그리고 함수 내부의 지역변수가 할당 되는 저장공간이다.
- 스택 영역은 메모리 주소가 높은 값에서 낮은 값으로 저장 장소가 할당 된다. 그러므로 함수 호출과 종료에 따라 높은 주소에서 낮은 주소로 메모리가 할당되었다가 다시 제거되는 작업이 반복된다.

12장 변수 유효범위

변수의 이용

- 변수의 이용 기준

1. 실행 속도를 개선하고자 하는 경우에 제한적으로 특수한 지역변수인 레지스터 변수를 이용한다
2. 함수나 블록 내부에서 함수나 블록이 종료되더라도 계속적으로 값을 저장하고 싶을 때는 정적 지역변수를 이용한다
3. 해당 파일 내부에서만 변수를 공유하고자 하는 경우는 정적 지역변수를 이용한다
4. 프로그램의 모든 영역에서 값을 공유하고자 하는 경우는 전역변수를 이용한다



12장 변수 유효범위

변수의 종류

선언위치	상세 종류	키워드		유효범위	기억장소	생존기간
전역	전역 변수	참조선언	extern	프로그램 전역	메모리 (데이터 영역)	프로그램 실행시간
	정적 전역변수	static		파일 내부		
지역	정적 지역변수	static		함수나 블록 내부	레지스터	함수 또는 블록 실행시간
	레지스터 변수	register			메모리 (스택 영역)	
	자동 지역변수	auto (생략가능)				

12장 변수 유효범위

변수의 유효범위

구분	종류	메모리 할당 시기	동일 파일 외부 함수에서의 이용	다른 파일 외부 함수에서의 이용	메모리제거 시기
전역	전역 변수	프로그램 시작	O	O	프로그램종료
	정적 전역변수	프로그램 시작	O	X	프로그램종료
지역	정적 지역변수	프로그램 시작	X	X	프로그램종료
	레지스터 변수	함수(블록) 시작	X	X	함수(블록) 종료
	자동 지역변수	함수(블록) 시작	X	X	함수(블록) 종료

12장 변수 유효범위

변수의 초기값

지역, 전역	종류	자동 저장되는 기본 초기값	초기값 저장
전역	전역 변수	자료형에 따라 0이나 '0' 또는 NULL 값이 저장됨	프로그램 시작 시
	정적 전역변수		
지역	정적 지역변수	쓰레기 값이 저장됨	함수나 블록이 실행될 때마다
	레지스터 변수		
	자동 지역변수		

13장 구조체와 공용체



13장 구조체와 공용체

구조체 개념

1) 구조체 개념

- 정수나 문자, 실수나 포인터 그리고 이들의 배열 등을 묶어 하나의 자료형으로 이용하는 것이 구조체이다.
- 연관성이 있는 서로 다른 개별적인 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형을 구조체라 한다.
- 구조체는 연관된 멤버로 구성되는 통합 자료형으로 대표적인 유도 자료형이다.
- 기존 자료형으로 새로이 만들어진 자료형을 유도 자료형이라 한다.
(유도자료형은 사용자정의 자료형이라 부르며, 구조체, 공용체, 열거형 등이 있으며 배열과 포인터도 유도 자료형이다.)

2) 구조체 정의

- 구조체를 자료형으로 사용하려면 먼저 구조체를 정의해야 한다.
- 구조체를 사용하려면 먼저 구조체를 만들 구조체 틀을 정의해야 한다.
- 구조체를 구성하는 하나 하나의 항목을 구조체 멤버 또는 필드라 한다
- 구조체 정의는 변수의 선언과는 다른 것으로 변수선언에서 이용될 새로운 구조체 자료형을 정의하는 구문이다.
- 구조체 멤버로는 일반변수, 포인터 변수, 배열, 다른 구조체 변수 및 구조체 포인터도 허용된다.

13장 구조체와 공용체

구조체 개념

1) 구조체 개념

- 정수나 문자, 실수나 포인터 그리고 이들의 배열 등을 묶어 하나의 자료형으로 이용하는 것이 구조체이다.
- 연관성이 있는 서로 다른 개별적인 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형을 구조체라 한다.
- 구조체는 연관된 멤버로 구성되는 통합 자료형으로 대표적인 유도 자료형이다.
- 기존 자료형으로 새로이 만들어진 자료형을 유도 자료형이라 한다.
(유도자료형은 사용자정의 자료형이라 부르며, 구조체, 공용체, 열거형 등이 있으며 배열과 포인터도 유도 자료형이다.)

```
//구조체 틀: 정의
struct lecture {
    char name[20]; //강좌명
    int credit;    //학점
    int hour;      //시수
};
```

2) 구조체 정의

- 구조체를 자료형으로 사용하려면 먼저 구조체를 정의해야 한다.
- 구조체를 사용하려면 먼저 구조체를 만들 구조체 틀을 정의해야 한다.
- 구조체를 구성하는 하나 하나의 항목을 구조체 멤버 또는 필드라 한다.
- 구조체 정의는 변수의 선언과는 다른 것으로 변수선언에서 이용될 새로운 구조체 자료형을 정의하는 구문이다.
- 구조체 멤버로는 일반변수, 포인터 변수, 배열, 다른 구조체 변수 및 구조체 포인터도 허용된다.

```
//구조체를 자료형으로 사용
struct lecture datastructure;
```


13장 구조체와 공용체

구조체 변수 선언

1) 구조체 자료형 변수 선언 및 초기화 구문

```
struct 구조체태그이름 변수명;  
//여러 변수의 선언도 가능  
struct 구조체태그이름 변수명1, 변수명2, 변수명3;
```

2) 구조체 정의와 변수 선언을 함께하는 방법

```
struct account {  
    char name[12];    //계좌주이름  
    int actnum;       //계좌번호  
    double balance;   //잔고  
} myaccount;         //struct account형 변수로 선언  
  
struct account youraccount;
```

3) 구조체 태그이름이 없는 변수 선언 방법

- 구조체 변수 선언 구문에서 구조체 태그이름을 생략할 수 있다.
- 태그이름이 없는 구조체 정의에서는 바로 변수가 나오지 않는다면 아무 의미 없는 문장이 된다.

```
struct account {  
    char name[12];    //계좌주이름  
    int actnum;       //계좌번호  
    double balance;   //잔고  
} youraccount;  
//이후에 동일한 구조체의 변수선언 불가능
```

13장 구조체와 공용체

구조체 변수의 초기화

- 초기화 값은 다음과 같이 중괄호 내부에서 구조체의 각 멤버 정의 순서대로 초기값을 쉼표로 구분하여 기술한다.
- 배열과 같이 초기값에 기술되지 않은 멤버 값은 자료형에 따라 기본값인 0, 0.0, '\0' 등으로 저장된다.
- `struct 구조체태그이름 변수명 = { 초기값1, 초기값2, 초기값3, ...};`

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

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


13장 구조체와 공용체

구조체 멤버 접근 연산자 .와 변수 크기

- 선언된 구조체형 변수는 접근연산자 .를 사용하여 멤버를 참조할 수 있다.
- 구조체변수이름.멤버
- mine.actnum = 1002; mine.balance = 300000;
- 실제 구조체의 크기는 멤버의 크기의 합보다 크거나 같다.

```
strcpy(yours.name, "강연승");  
//접근연산자 .를 사용한 멤버의 참조  
yours.actnum = 1002;  
yours.balance = 500000;  
  
printf("구조체크기 : %d\n", sizeof(mine));
```

13장 구조체와 공용체

구조체 멤버로 사용되는 구조체

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

구조체 멤버로
다른 구조체 변수를 허용한다.

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

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

struct account me = { {2018,3,9}, "홍길동", 1001, 300000 };
```

13장 구조체와 공용체

구조체 정의의 위치

- 구조체 정의는 변수의 선언처럼 그 정의 위치에 따라 구조체의 유효 범위가 결정된다
- 구조체의 정의도 변수 선언처럼 유효범위는 전역 또는 지역으로 모두 가능하다.

전역

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

```
int main(void) {  
    struct account {  
        char name[12];    //계좌주 이름  
        int actnum;       //계좌번호  
        double balance;   //잔고  
    };  
  
    //구조체 변수 선언 및 초기화  
    struct account mine = { "홍길동", 1001, 300000 };  
  
    return 0;  
}
```

지역

13장 구조체와 공용체

구조체 변수의 대입과 동등비교

- 구조체 변수의 대입

```
struct student
{
    int snum;        //학번
    char* dept;      //학과 이름
    char name[12];   //학생 이름
};

struct student hong = { 201800001, "컴퓨터정보공학과", "홍길동" };
struct student one;

one = hong;
```

- 구조체의 동등 비교

```
if (one == bae) //오류
    printf("내용이 같은 구조체입니다.");

if (one.snum == bae.snum)
    printf("학번이 %d으로 동일합니다.", one.snum);

if (one.snum == bae.snum && !strcmp(one.name, bae.name) && !strcmp(one.dept, bae.dept))
    printf("내용이 같은 구조체입니다.");
```

13장 구조체와 공용체

문자열을 처리하기 위한 포인터 char *와 배열 char []

char 포인터

```
char *dept;           //학과 이름
```

```
char *dept = "컴퓨터정보공학과";
```



변수 dept는 포인터로 단순히 문자열 상수를 다루는 경우 효과적

```
dept = "컴퓨터정보공학과";
```

단지 문자열 상수의 첫 주소를 저장하므로 문자열 자체를 저장하거나 수정하는 것은 불가능 하므로 다음 구문은 사용 불가능

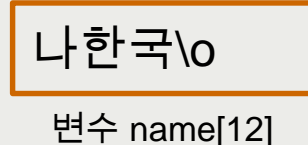
```
strcpy(dept, "컴퓨터정보공학과"); //오류
```

```
scanf("%s", dept); ; //오류
```

char 배열

```
char name[12];        //학생 이름
```

```
char name[12] = "나한국";
```



변수 name은 배열로 12바이트 공간을 가지며 문자열을 저장하고 수정 등이 필요한 경우 효과적

```
name = "나한국"; //오류
```

문자열 자체를 저장하는 배열이므로 문자열의 저장 및 수정이 가능하고 문자열 자체를 저장하는 다음 구문 사용도 가능

```
strcpy(name, "배상문");
```

```
scanf("%s", name);
```


13장 구조체와 공용체

공용체 활용

- 공용체 개념
 - 동일한 저장 장소에 여러 자료형을 저장하는 방법

```
union share {  
    int count;  
    float value;  
};
```

```
union share a;
```

a.count = 55;

a.value = 243.5;

count는 첫 4바이트

동일한 저장 공간

value는 전체 8바이트

13장 구조체와 공용체

Union을 사용한 공용체 정의 및 변수 선언

- 공용체(union)는 서로 다른 자료형의 값을 동일한 저장공간에 저장하는 자료형이다.
- 공용체 정의 구문

union 공용체태그이름

```
{  
  자료형 멤버변수명1;  
  자료형 멤버변수명2;  
  ...  
}
```

공용체 구성요소인 멤버(struct member)이다

} [변수명1] [,변수명2] ;

```
union data {  
    char ch;           //문자형  
    int cnt;           //정수형  
    double real;       //실수형  
} data1;
```

```
union udata {  
    char name[4];      //char형 배열  
    int n;             //정수형  
    double val;        //실수형  
};
```

13장 구조체와 공용체

Union을 사용한 공용체 정의 및 변수 선언

- 공용체 변수의 크기는 멤버 중 가장 큰 자료형의 크기로 정해진다
- 공용체의 멤버는 모든 멤버가 동일한 저장 공간을 사용하므로 동시에 여러 멤버의 값을 동시에 저장하여 이용할 수 없으며, 마지막에 저장된 단 하나의 멤버 자료 값 만을 저장한다.
- 공용체의 초기화 값은 공용체 정의 시 처음 선언한 멤버의 초기값으로만 저장이 가능하다.
- 공용체 초기화 및 경고

```
typedef union data uniondata;
```

```
uniondata data2 = { 'A' };    //첫 멤버인 char형으로만 초기화 가능
```

```
//uniondata data2 = { 10.3 };    //컴파일 시 경고 발생
```

```
//warning C4244: '초기화중' : 'double'에서'char'(으)로 변환하면서 데이터가 손실될 수 있습니다.
```

```
uniondata data3 = data2;
```

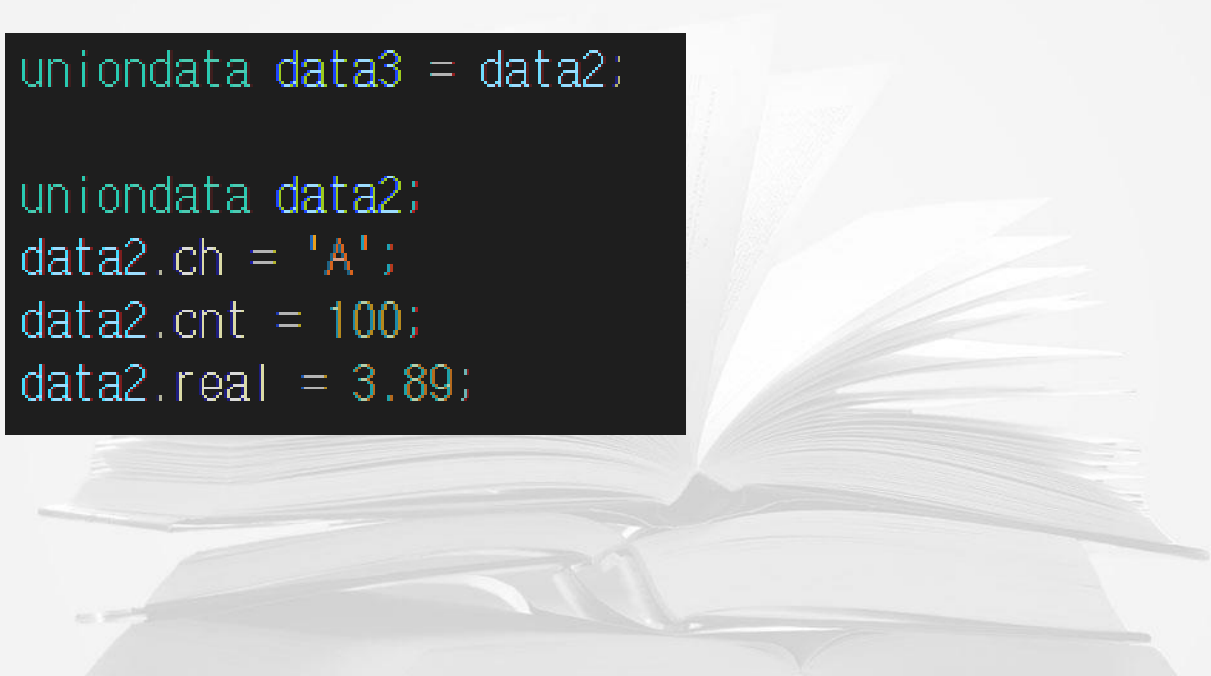
13장 구조체와 공용체

공용체 멤버 접근

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

```
uniondata data3 = data2;
```

```
uniondata data2;  
data2.ch = 'A';  
data2.cnt = 100;  
data2.real = 3.89;
```



13장 구조체와 공용체

자료형 재정의 typedef

- Typedef는 이미 사용되는 자료 유형을 다른 새로운 자료형 이름으로 재정의할 수 있도록 하는 키워드이다.

```
typedef 기존자료유형이름 새로운자료형1, 새로운자료형2, ...;  
//여러 이름으로도 재정의할 수 있다.  
  
typedef int profit;  
typedef unsigned int budget;  
typedef unsigned int size_t;  
//새로운 이름 size_t도 자료형 unsigned int와 동일하게 이용 가능
```

- 일반적으로 자료형을 재정의하는 이유는 프로그램의 시스템 간 호환성과 편의성을 위해 필요하다.

Visual C++: 4바이트

```
int salary = 2000000;
```

Turbo C++: 2바이트

```
int salary = 2000000;
```

오버플로 발생(데이터 손실)

Visual C++: 4바이트

```
typedef int myint;  
...  
myint salary = 2000000;
```

Turbo C++: 2바이트

```
typedef long myint;  
...  
myint salary = 2000000;
```

이 typedef 문장만 수정하면 터보 C++에서 이소스를 그대로 이용 가능

```
typedef int integer, word;  
  
integer myAge;      //int myAge와 동일  
word yourAge        //int yourAge와 동일
```

- 하나의 자료형을 여러 자료형으로 재정의

- 문장 typedef도 일반 변수와 같이 그 사용 범위를 제한한다.

13장 구조체와 공용체

Struct를 생략한 새로운 자료형

- 구조체 struct date가 정의된 상태에서 typedef 사용하여 구조체 struct date를 date로 재정의 할 수 있다.

```
struct date {  
    int year;      //년  
    int month;     //월  
    int day;       //일  
};  
//자료유형인 date는 struct date와 함께 동일한 자료유형으로 이용이 가능하다.  
typedef struct date date;
```

- 새로운 자료유형 date의 재정의와 이용

```
typedef struct {  
    char title[30];      //제목  
    char company[30];    //제작회사  
    char kinds[30];      //종류  
    date release;        //출시일  
} software;              //software는 변수가 아니라 새로운 자료형이다.
```

- 구조체 정의와 typedef를 함께 이용한 자료형의 정의

13장 구조체와 공용체

포인터 변수 선언

- 포인터는 각각의 자료형 저장 공간의 주소를 저장하듯이 구조체 포인터는 구조체의 주소 값을 저장 할 수 있는 변수이다.

- 구조체 포인터 변수 선언

```
struct lecture {  
    char name[20];    //강좌명  
    int type;         //강좌구분  
    int credit;       //학점  
    int hours;        //시수  
};  
typedef struct lecture lecture;  
lecture* p;
```

- 구조체 변수 os와 포인터 변수 p의 구조체 멤버 참조 방법

```
lecture os = { "운영체제", 2, 3, 3 };  
lecture *p = &os;
```

13장 구조체와 공용체

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 연산식이다.
- 연산식 *p.name은 접근연산자(.)가 간접연산자(*)보다 우선순위가 빠르므로 *(p.name)과 같은 연산식이다.

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

13장 구조체와 공용체

공용체 포인터

- 공용체 변수도 포인터 변수 사용이 가능하다.
- 공용체 포인터 변수로 멤버를 접근하려면 접근연산자 ->를 이용한다.

```
union data {  
    char ch;  
    int cnt;  
    double real;  
} value, *p;  
  
p = &value;      //포인터 p에 value의 주소값을 저장  
p->ch = 'a';     //value.ch = 'a';와 동일한 문장
```

변수 value는 union data형이며 p는 union data포인터 형으로 선언

13장 구조체와 공용체

구조체 배열 변수 선언

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

```
lecture c[] = { {"인간과사회", 0, 2, 2},  
                {"경제학개론", 1, 3, 3},  
                {"자료구조", 2, 3, 3} };
```

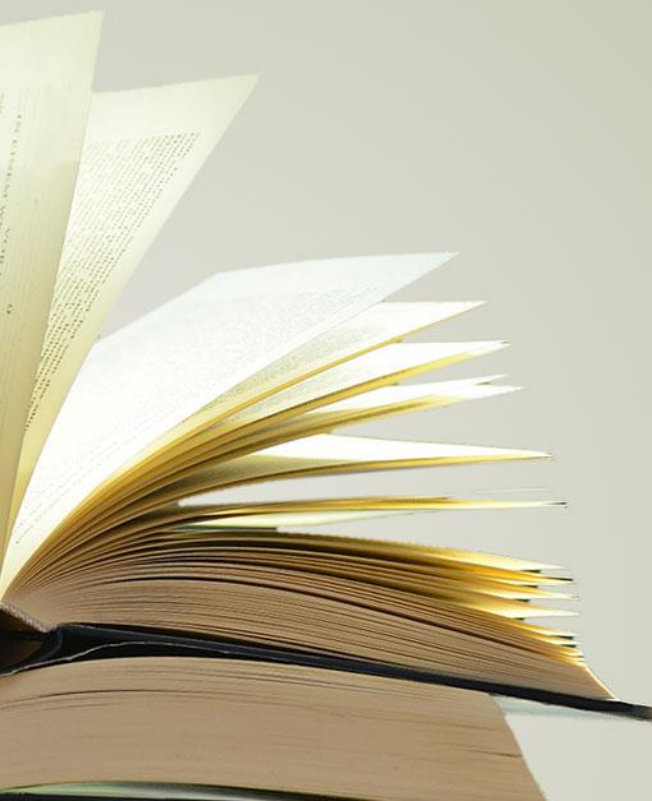
- 배열크기: `sizeof(c) / sizeof(c[0])` 또는 `sizeof(c) / sizeof(lecture)`
- 배열의 주소를 저장

```
lecture* p = c;
```

```
//p로도 참조가능
```

```
for (i = 0; i < arysize; i++)  
    printf("%16s %10s %5d %5d", p[i].name, lectype[p[i].type], p[i].crdeit, p[i].hours);
```

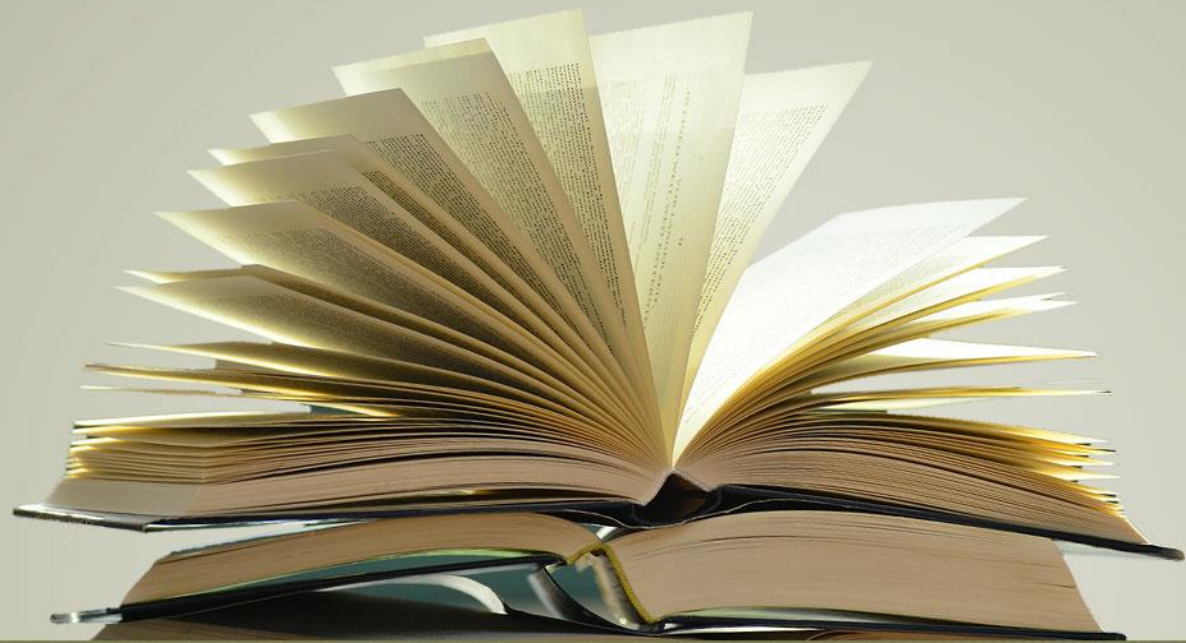

포트폴리오를 마치며



이번 C포트폴리오를 하면서
C언어의 개념을 정리하면서
학습하는 계기가 되어 뿌듯하고
평소에 그냥 넘겨 짚어 놓쳤던 부분도 놓치지 않고
확인 할 수 있었을 뿐더러
이미 알고 있던 부분은 다시 복습하면서
더 확실하게 알아가게 되지 않았나 싶습니다.

앞으로 있을 기말고사 준비에도
도움이 될 것이라 생각이 됩니다.

비록 이번 포트폴리오에서는 모든 부분을 담을 수 없었지만
C언어를 능숙하게 다룰 수 있도록 다른 부분들도
스스로 공부하면서 노력하도록 하겠습니다.
감사합니다.



감사합니다.