



C 언어

20193427 최권우 / 2020년 1학기

강의계획서

2020 학년도 1학기	전공	컴퓨터정보공학과(위)	학부	컴퓨터공학부
과 목 명	C에플리케이션선구현(2016003-PH)			
강의실 과 강의시간	월:11(3-217),12(3-217),13(3-217),14(3-217)		학점	4
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, 강환 일, 이동규	인피니티북스	
수업시 사용도구	Visual C++			
평가방법	중간고사 30%, 기말고사 30%, 과제를 및 퀴즈 20%, 출석 20%			
수강안내	C 언어를 활용하여 응용프로그램을 구현할 수 있다.			

11. 문자와 문자열



학습목표

- 문자와 문자열을 이해하고 설명할 수 있다.
- 문자와 문자열 입출력을 이해하고 설명할 수 있다.
- 문자열 관련 함수를 이해하고 설명할 수 있다.
- 여러 개의 문자열 처리하는 방법에 대해 이해하고 설명할 수 있다.



문자와 문자열의 개념

문자 C언어에서 문자는 알파벳이나 한글의 한 글자를 작은 따옴표로 감싼 ‘A’와 같이 표현

작은 따옴표에 의해 표기된 문자를 문자 상수라고 한다.

문자열 문자의 모임인 일련의 문자를 문자열 이라고 한다.

문자열은 일련의 문자 앞 뒤로 큰 따옴표로 둘러싸서 “c lang”로 표기한다

문자의 나열인 문자열은 ‘ABC’처럼 작은 따옴표로 둘러싸도 문자가 될 수 없으며 오류가 발생한다.



문자와 문자열의 선언

char형 변수에 문자를 저장한다.

문자열을 저장하려면 문자의 모임인 ‘문자 배열’을 사용한다.

문자열의 마지막을 의미하는 NULL 문자 ‘\0’가 마지막에 저장되어야 한다.

문자열이 저장되는 배열크기는 반드시 저장될 문자 수보다 1이 커야된다. (NULL 문자 때문에)



문자와 문자열의 선언

문자열을 선언하는 편리한 다른 방법을 살펴보면, 배열 선언시 저장할 큰 따옴표를 사용해 문자열 상수를 바로 대입하는 방법이 있다.

배열 초기화시 배열크기는 지정하지 않는 것이 더 편리, 만일 지정한다면 마지막 문자 ‘\0’을 고려해 문자 수보다 1크게 지정

만일 지정한 배열크기가 문자수+1 보다도 크다면 나머지는 ‘\0’로 채워진다.



문자와 문자열의 선언

```
char ch = 'A'; // 문자 저장
```

```
char csharp[3]; // 3개 지정
```

```
char csharp2[] = "C sharp"; // 컴파일러에서 지정
```

```
char csharp2_1[8] = "C sharp"; // 총문자 수 보다 1크게 지정
```

```
char csharp3[] = {'C', 'S', '\0'}; // 문자 하나 하나 저장 마지막에 NULL 문자 저장 해야 출력에 문제 없음
```




문자와 문자열 출력

printf 함수로 문자와 문자열을 출력할 수 있다.

문자는 %c 형식제어문자로 문자열은 %s 형식제어문자로 출력한다.

printf(c) 처럼 바로 배열이름을 인자로 사용해도 문자열 출력이 가능하다.

puts 함수로도 출력이 가능한데 puts(string) 처럼 사용하면 한줄에 문자열을 출력한 뒤 다음 줄에서 출력을 준비한다.



문자열 구성하는 문자 참조

문자열을 처리하는 다른 방법은 문자열 상수를 문자 포인터에 저장하는 방식이다.

문자 포인터에 의한 선언으로 문자 하나 하나의 수정은 할 수 없다.

`char *cpp = "cpp";` “cpp” 문자열 상수 처럼 선언된 것은 말 그대로 상수이므로 문자 하나 하나 수정이 불가능 하다.



‘\0’ 문자에 의한 문자열 분리

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

```
char c[] = "C C++ Java";
```

```
c[5] = '\0';
```

```
printf("%s\n%s\n", c, (c+6));
```

결과

C C++

Java



버퍼처리 함수 getchar()

문자 입력을 위한 함수 getchar는 라인 버퍼링(line buffering) 방식을 사용한다.

문자 하나를 입력해도 반응을 보이지 않다가 엔터키를 누르면 그제서야 이전에 입력한 문자마다 입력이 실행된다.



함수 getch()

버퍼를 사용하지 않고 문자를 입력하는 함수

- 함수는 버퍼를 사용하지 않고 문자 하나를 바로 바로 입력할 수 있는 함수이다.
- 함수를 이용하려면 헤더 파일 `conio.h`를 삽입해야됨



함수 getch()

문자 입력을 위한 함수 getch()는 입력한 문자가 화면에 보이지 않는 특성이 있다.

- 함수 getch()도 버퍼를 사용하지 않는 문자 입력 함수다.
- 함수 getch()도 conio.h 헤더 파일을 삽입해야 사용할 수 있다.

현재는 여러 컴파일러에서 함수 getch(), getche()는 _getche(), _getch()로 이름이 수정되어 있다.



문자열 입력

문자배열 변수로 `scanf()`에서 입력은 `scanf()`로 받을 문자열의 크기 보다 충분한 문자배열을 선언한다.

함수 `scanf("%s", str)`에서 형식제어문자 `%s`를 사용하여 문자열 입력을 받을 수 있다.



gets()와 puts()

함수 gets()는 한 행의 문자열 입력에 유용한 함수이다.

- 마지막에 입력된 ‘\n’이 ‘\0’으로 교체되어 인자인 배열에 저장됨

함수 puts()는 한 행에 문자열을 출력하는 함수이다.

- 기호 상수 EOF(end of file)은 파일의 끝이라는 의미로 stdio.h 헤더 파일에 정수 -1로 정의되어 있음

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



다양한 문자열 라이브러리 함수

`void *memchr(const void *str, int c, size_t n)`, str에서 n 바이트까지 문자 c를 찾아 그 위치를 반환

`int memcmp(const void *str1, const void *str2, size_t n)` 메모리 str1과 str2를 첫 n바이트를 비교 검색하여 같으면 0, 다르면 음수 또는 양수 반환

`void *memcpy(void *dest, const void *src, size_t n)` 포인터 src 위치에서 dest에 n 바이트를 복사한 후 dest 위치 반환

`void *memmove(void *dest, const void *src, size_t n)` 포인터 src 위치에서 dest에 n 바이트를 복사한 후 dest 위치 반환

`void *memset(void *str, int c, size_t n)` 포인터 str 위치에서부터 n 바이트까지 문자 c를 지정한 후 str 위치 반환

`size_t strlen(const char *str)` 포인터 str 위치에서부터 널 문자를 제외한 문자열 길이 반환



다양한 문자열 라이브러리 함수

함수 `strcmp()` - 문자열 비교와 복사 그리고 문자열 연결 등과 같은 다양한 문자열 처리는 헤더파일 `string.h`에 함수원형으로 선언된 라이브러리 함수로 제공됨

함수 `strcpy()` - 함수 `strcpy()`와 `strncpy()`는 문자열을 복사하는 함수이다. 함수 `strcpy()`는 앞 인자 문자열 `dest`에 뒤 인자 문자열 `source`를 복사한다.

함수 `strcat()` - 함수 `strcat()`은 앞 문자열에 뒤 문자열의 NULL문자 까지 연결하여, 앞의 문자열 주소를 반환하는 함수이다.



다양한 문자열 라이브러리 함수

함수 `strtok()`은 문자열에서 구분자인 문자를 여러개 지정하여 토큰을 추출하는 함수이다.

함수 `strlen()`은 NULL문자를 제외한 문자열 길이를 반환하는 함수이다.



문자 포인터 배열과 이차원 문자 배열

문자포인터 배열

- 여러개의 문자열을 처리하는 하나의 방법은 문자 포인터 배열을 이용하는 방법이다.

이차원 문자 배열

- 여러개의 문자열을 처리하는 다른 방법은 문자의 이차원 배열을 이용하는 방법이다.



명령행 인자

```
main(int argc, char *argv[])
```

- 명령행에서 입력하는 문자열을 프로그램으로 전달하는 방법이 명령행 인자를 사용하는 방법이다.
- 주의할 점은 실행 프로그램 이름도 명령행 인자에 포함된다.

12. 전역변수와 지역변수



학습목표

변수 유효범위를 이해하고 설명할 수 있다.

- 지역변수, 자동변수를 선언하고 사용

전역변수를 선언하고 사용

- 키워드 extern의 필요성과 사용 방법

정적 변수와 레지스터 변수를 이해하고 설명할 수 있다.

- 기억 부류 auto, static, register, extern
- 지역 정적 변수와 전역 정적 변수의 필요성과 사용



변수 범위와 지역변수

변수 scope

- 변수의 참조가 유효한 범위를 변수의 유효 범위(scope)라고 한다.
- 종류는 지역 유효범위(local scope)와 전역 유효 범위(global scope)로 나뉜다.
 - 지역 유효 범위는 함수 또는 블록 내부에서 선언되어 그 지역에서 변수의 참조가 가능한 범위



지역변수

지역변수는 함수 또는 블록에서 선언된 변수다.

함수나 블록에서 지역변수는 선언 문자 이후에 함수나 블록의 내부에서만 사용 가능

함수의 매개변수도 함수 전체에서 사용 가능한 지역변수와 같다

지역변수는 선언 후 초기화하지 않으면 쓰레기값이 저장되므로 주의해야 한다.

지역변수가 할당되는 메모리 영역을 스택이라 한다, 지역변수는 선언된 부분에서 자동으로 생성되고 함수나 블록이 종료되는 순간 메모리에서 자동 제거됨, 이러한 특성때문에 지역변수는 자동변수라고도 한다.



전역변수

전역변수는 함수 외부에서 선언되는 변수다.

전역변수는 외부변수라고도 부른다.

전역변수는 일반적으로 프로젝트의 모든함수나 블록에서 참조 가능하다.

전역변수는 프로젝트의 다른 파일에서도 참조가 가능, 다른 파일에서 선언된 전역변수를 참조하려면 `extern` 키워드를 사용하여 다른 파일에서 선언되었음을 선언해야됨



전역변수의 장단점

- 동일한 파일에서도 extern을 사용해야 하는 경우가 발생할 수있다.
- 전역변수에 예상하지 못한 값이 저장된다면 프로그램 어느 부분에서 수정되었는지 알기 어려운 단점이 있다.



기억부류와 레지스터 변수

4가지의 기억부류인 auto, register, static, extern에 따라 할당되는 메모리 영역이 결정되고 메모리의 할당과 제거 시기가 결정된다.

- 기억 부류 auto와 register는 지역변수에만 이용이 가능
- static은 지역과 전역 모든 변수에 이용 가능
- extern은 전역변수에만 사용 가능

키워드 extern을 제외하고 나머지 3개의 기억부류의 변수선에서 초기값을 지정할 수 있다.



키워드 register

레지스터 변수는 변수의 저장공간이 일반 메모리가 아니라 PCU 내부의 레지스터에 할당되는 변수

- 레지스터 변수는 키워드 register를 자료형 앞에 넣어 선언한다.
- 레지스터 변수는 일반 메모리에 할당되는 변수와 다르므로 주소연산자 &를 사용할 수 없다.

주로 레지스터 변수는 처리 속도를 증가시키려는 변수에 이용한다. 특히 반복문의 횟수를 제어하는 제어변수에 이용하면 효과적이다.



정적변수, 정적 지역변수

키워드 static - 변수 선언에서 자료형 앞에 static을 넣어 정적변수를 선언할 수 있다.

- 정적변수는 초기값을 지정하지 않으면 자동으로 자료형에 따라 0이나 '\0' 또는 NULL 값이 들어간다.

함수나 블록에서 정적으로 선언되는 변수가 정적 지역변수다

- 정적 지역변수는 함수나 블록을 종료해도 메모리에서 제거되지 않고 계속 메모리에 유지 관리되는 특성이 있다.



정적 전역 변수

함수 외부에서 정적으로 선언되는 변수가 정적 전역변수다

- 정적 전역변수는 선언된 파일 내부에서만 참조가 가능한 변수다.
- 프로그램이 크고 복잡하면 전역변수의 사용은 원하지 않는 전역변수의 수정과 같은 부작용(side effect)의 위험성이 항상 존재한다.



메모리 영역

데이터, 스택, 힙 영역

- 메인메모리의 영역은 프로그램 실행 과정에서 데이터영역, 힙 영역, 스택 영역 세 부분으로 나뉜다. 메모리 영역은 변수의 유효범위 생존기간에 결정적인 역할을 하며 변수는 기억부류에 따라 할당되는 메모리 공간이 달라진다.
- 힙 영역은 동적할당되는 변수가 할당되는 저장공간이다.
- 스택영역은 함수 호출에 의한 형식 매개변수 그리고 함수 내부의 지역변수가 할당되는 저장공간이다.
- 즉 스택 영역은 메모리 주소가 높은 값에서 낮은 값으로 저장 장소가 할당, 그러므로 함수 호출과 조율에 따라 높은 주소에서 낮은 주소로 메모리가 할당되었다가 다시 제거되는 작업이 반복됨



변수의 이용

이용 기준

- 함수나 블록 내부에서 함수나 블록이 종료되더라도 계속적으로 값을 저장하고 싶을 때는 정적 지역변수를 이용한다
- 해당 파일 내부에서만 변수를 공유하고자 하는 경우 정적 전역변수를 이용한다.

13. 구조체와 공용체



학습목표

구조체와 공용체를 이해하고 설명할 수 있다.

- 구조체의 개념과 정의 방법
- 필요한 구조체 변수의 선언 방법
- 구조체 변수의 접근연산자 .의 사용 방법
- 공용체 정의와 변수 선언 및 활용 방법

자료형 재정의를 위한 typedef를 사용할 수 있다.

- 키워드 typedef를 사용한 자료형 재정의 방법과 필요성
- 구조체 정의를 새로운 자료형으로 재정의

구조체 포인터와 배열을 활용할 수 있다.

- 구조체의 주소를 저장하는 포인터의 선언과 활용
- 구조체 포인터의 접근연산자 -> 의 사용방법
- 구조체 배열의 선언과 활용방법



구조체 개념

정수나 문자, 실수나 포인터 그리고 이들의 배열 등을 묶어 하나의 자료형으로 이용하는 것이 구조체다.

연관성이 있는 서로 다른 개별적인 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형을 구조체라 한다.

구조체는 연관된 멤버로 구성되는 통합 자료형으로 대표적인 유도 자료형이다.

즉 기초 자료형으로 새로 만들어진 자료형을 유도 자료형이라 한다.



구조체 정의

구조체를 자료형으로 사용하려면 먼저 구조체를 정의해야 한다.

구조체를 사용하려면 먼저 구조체를 만들 구조체 틀을 정의해야 함

구조체를 정의하는 방법은 키워드 `struct` 다음에 구조체 태그이름을 기술하고 중괄호를 이용하여 원하는 멤버를 여러 개의 변수로 선언하는 구조다. 구조체를 구성하는 하나 하나의 항목을 구조체 멤버 또는 필드라고한다.

- 구조체 정의는 변수의 선언과는 다른 것으로 변수선언에서 이용될 새로운 구조체 자료형을 정의하는 구문이다



구조체 변수 선언과 초기화

구조체 변수 선언

- 새로운 자료형 `struct account` 형 변수 `mine`을 선언하려면 `struct account mine;`으로 선언한다

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, `'\0'` 등으로 저장된다.



구조체의 멤버 접근 연산자 . 와 변수 크기

선언된 구조체형 변수는 접근연산자 . 을 사용하여 멤버를 참조할 수 있다.

실제 구조체의 크기는 멤버의 크기의 합보다 크거나 같다.



공용체 활용

공용체 개념

- 동일한 저장 장소에 여러 자료형을 저장하는 방법

union을 사용한 공용체 정의 및 변수 선언

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



공용체 활용

공용체 멤버 접근

- 공용체 변수로 멤버를 접근하기 위해서 구조체와 같이 접근 연산자 .을 사용한다.



자료형 재정의 typedef

typedef 구문

- typedef는 이미 사용되는 자료 유형을 다른 새로운 자료형 이름으로 재정의할 수 있도록 하는 키워드다
- 일반적으로 자료형을 재정의하는 이유는 프로그램의 시스템 간 호환성과 편의성을 위해 필요하다.
- 문자 typedef도 일반 변수와 같이 그 사용 범위를 제한함



구조체 자료형 재정의

struct를 생략한 새로운 자료형

- 구조체 struct date가 정의된 상태에서 typedef 사용하여 구조체 struct date를 date로 재정의



구조체 포인터

포인터 변수 선언

- 포인터는 각각의 자료형 저장 공간의 주소를 저장하듯이 구조체 포인터는 구조체의 주소값을 저장할 수 있는 변수이다.

포인터 변수의 구조체 멤버 접근 연산자 ->

- 구조체 포인터 멤버 접근연산자 ->는 `p->name`과 같이 사용한다. 연산식 `p->name`은 포인터 `p`가 가리키는 구조체 변수의 멤버 `name`을 접근하는 연산식이다.
- 연산식 `*p.name`은 접근연산자(`.`)가 간접연산자(`*`)보다 우선순위가 빠르므로 `*(p.name)`과 같은 연산식이다.



공용체 포인터

공용체 변수도 포인터 변수 사용이 가능

공용체 포인터 변수로 멤버를 접근하려면 접근 연산자 `->`를 이용한다.



구조체 배열

구조체 배열 변수 선언

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

14. 함수와 포인터 활용



학습목표

함수의 인자전달 방식을 이해하고 설명할 수 있다.

- 값의 의한 호출과 참조에 의한 호출 방식
- 함수에서 인자와 반환값으로 배열의 주고 받는 활용
- 가변 인자의 필요성과 사용 방법

함수에서 인자로 포인터의 전달과 반환으로 포인터 형의 사용을 이해하고 설명할 수 있다.

- 매개변수 전달과 반환으로 포인터 사용
- 포인터 인자전달 시 키워드 const의 이용
- 함수에서 구조체 전달과 반환

함수 포인터를 이해하고 설명할 수 있다.

- 함수 포인터의 필요성과 사용
- 함수 포인터 배열의 사용
- void 포인터의 필요성과 사용



함수에서 값의 전달

C 언어는 함수의 인자 전달 방식이 기본적으로 값에 의한 호출 방식

값에 의한 호출 방식이란 함수 호출 시 실인자의 값이 형식인자에 복사되어 저장된다는 의미

값에 의한 호출 방식을 사용해서는 함수 외부의 변수를 함수 내부에서 수정할 수 없는 특징이 있다.



함수에서 주소의 전달

C 언어에서 포인터를 매개변수로 사용하면 함수로 전달된 실인자의 주소를 이용하여 그 변수를 참조
이와 같이 함수에서 주소의 호출을 참조에 의한 호출이라 함



배열이름으로 전달

함수의 매개변수로 배열을 전달하는 것은 배열의 첫 원소를 참조 매개변수로 전달하는 것과 동일

배열크기에 관계없이 배열 원소의 합을 구하는 함수를 만들려면 배열크기도 하나의 인자로 사용해야됨

함수원형에서 매개변수는 배열이름을 생략하고 `double[]`와 같이 기술할 수 있다. 함수호출에서 배열 인자에는 반드시 배열이름으로 `sum(data, 5)`와 같이 기술



다양한 배열원소 참조 방법 및 크기 계산방법

다양한 배열원소 참조 방법

- 함수헤더에 `int ary[]`로 기술하는 것은 `int * ary`로 대체 가능

배열크기 계산방법

- 연산자 `sizeof`를 이용한 식 (`sizeof(배열이름)/sizeof(배열원소)`)의 결과는 배열크기이다.



다차원 배열 전달

다차원 배열을 인자로 이용하는 경우, 함수원형과 함수정의의 헤더에서 첫 번째 대괄호 내부의 크기를 제외한 다른 모든 크기는 반드시 기술해야됨

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



가변인자

가변인자가 있는 함수머리

- 출력할 인자의 수와 자료형은 인자 `_Format`에 `%d`등으로 표현되어 있다
- 함수에서 인자의 수와 자료형이 결정되지 않은 함수 인자 방식을 가변 인자라 함

가변인자가 있는 함수 구현

- 가변 인자를 구현하려면 가변인자 선언, 가변인자 처리 시작, 가변인자 얻기, 가변인자 처리 종료 4단계가 필요
- 헤더파일 `stdarg.h`가 필요하다.



포인터 전달과 반환

주소 연산자 &

- 함수에서 매개변수를 포인터로 이용하면 결국 참조에 의한 호출이 됨

주소값 반환

- 함수의 결과를 포인터로 반환
- 지역변수 주소값의 반환은 문제를 발생시킬 수 있다.



상수를 위한 const 사용

키워드 const

- 수정을 원하지 않는 함수의 인자 앞에 키워드 const 삽입하여 참조되는 변수가 수정될 수 없게 한다.



함수의 구조체 전달과 반환

복소수를 위한 구조체

- 구조체 complex를 이용하여 복소수 연산에 이용되는 함수를 만들어 보자
- 구조체 complex는 실수부와 허수부를 나타내는 real과 img를 멤버로 구성한다

```
struct complex
```

```
{  
double real; // 실수  
double img; // 허수  
};
```

```
typedef struct complex complex;
```



인자와 반환형으로 구조체 사용

함수 `paircomplex1()`은 인자인 복소수의 켤레 복소수 (pair complex number)를 구하여 반환하는 함수

- 구조체는 함수의 인자와 반환값으로 이용 가능하다.



함수 주소 저장 변수

포인터의 장점은 다른 변수를 참조하여 읽거나 쓰는 것도 가능하다

하나의 함수이름으로 필요에 따라 여러 함수를 사용하면 편리하다

함수 포인터는 함수의 주소값을 저장하는 포인터 변수다.



함수 포인터 배열

함수 포인터 배열 개념

- 함수 포인터 배열 (Array of function pointer)은 함수 포인터가 원소인 배열이다.

함수포인터 선언

- 반환자료형 (*배열이름[배열크기])(자료형1 매개변수이름1, 자료형2 매개변수이름2, ...);
- 반환자료형 (*배열이름[배열크기])(자료형1, 자료형2, ...);



void 포인터

void 포인터 개념

- 주소값이란 참조를 시작하는 주소에 불과하며 자료형을 알아야 참조할 범위와 내용을 해석할 방법을 알 수 있는 것

void 포인터 활용

- void 포인터는 모든 주소를 저장할 수 있지만 가리키는 변수를 참조하거나 수정이 불가능하다
- void 포인터는 자료형 정보없이 임시로 주소만 저장하는 포인터다.
- void 포인터로 변수를 참조하기 위해서는 자료형 변환이 필요하다.

15. 파일 처리



학습목표

텍스트 파일과 이진 파일의 차이를 이해하고 설명할 수 있다.

- 주기억장치와 파일의 차이
- 텍스트 파일과 이진 파일의 정의와 예
- 파일 스트림과 파일모드의 이해
- 함수 `fopen()`과 `fopen_s()`, `fclose()`의 사용

텍스트 파일의 입출력 함수를 이해하고 설명할 수 있다.

- 함수 `fprintf()`와 `fscanf()`, `fscanf_s()`의 사용
- 함수 `fputc()`와 `fgetc()`의 사용

이진 파일의 입출력 함수를 이해하고 설명할 수 있다.

- 함수 `fwrite()`와 `fread()`의 사용
- 함수 `getw()`와 `putw()`의 사용

파일 삭제 및 이름 바꾸기 함수를 이해하고 설명할 수 있다.

- 함수 `remove()`와 `rename()`의 사용



파일의 필요성

보조기억장치인 디스크에 저장되는 파일은 직접 삭제하지 않은 한 프로그램이 종료되더라도 계속 저장할 수 있다.

프로그램에서 사용하던 정보를 종료 후에도 계속 사용하고 싶다면 프로그램에서 파일에 그 내용을 저장해야 됨



텍스트 파일과 이진 파일

파일은 텍스트 파일과 이진 파일 두가지로 나뉜다.

텍스트 파일은 문자기반의 파일로서 내용이 ASCII CODE와 같은 문자 코드값으로 저장됨

이진 파일은 텍스트 파일과 다르게 그림 파일, 동영상 파일, 실행 파일과 같이 각각의 목적에 알맞은 자료가 이진 형태(binary format)로 저장되는 파일이다.

이진 파일은 컴퓨터 내부 형식으로 저장되는 파일이다.

이진 파일에서 자료는 메모리 자료 내용에서 어떤 변환도 거치지 않고 그대로 파일에 기록된다. 그러므로 입출력 속도도 텍스트 파일에 비해 빠르다.



입출력 스트림

자료의 입력과 출력은 자료의 이동이라고 볼수 있다.

자료가 이동하려면 이동 경로가 필요하다, 입출력 시 이동 통로가 바로 입출력 스트림이다.

다른 곳에서 프로그램으로 들어오는 경로가 입력 스트림이다.

프로그램에서 다른 곳으로 나가는 경로가 출력 스트림이다.



파일 스트림 이해

프로그램에서 보조기억장치에 파일로 정보를 저장하거나 파일에서 정보를 참조하려면 파일에 대한 파일 스트림을 먼저 연결해야 함

파일스트림이란 보조기억장치의 파일과 프로그램을 연결하는 전송경로다.

파일스트림은 입력을 위한 파일 입력 스트림과 출력을 위한 파일 출력 스트림으로 나뉨



파일 스트림 열기

`fopen()`으로 파일 스트림 열기, 프로그램에서 특정한 파일과 파일 스트림을 연결하기 위해서는 함수 `fopen()`또는 `fopen_s()`를 이용해야 한다.

`FILE`은 헤더 파일 `stdio.h`에 정의되어 있는 구조체 유형이다, 구조체 `FILE`은 파일을 표현하는 C 언어의 유도 자료형이다. 함수 `fopen()`은 인자가 파일이름과 파일열기 모드이며, 파일스트림 연결에 성공하면 파일 포인터를 반환하며 실패하면 `NULL`을 반환함

함수 `fopen_s()`는 파일 스트림 연결에 성공하면 정수 0을 반환, 만일 스트림 연결에 실패하면 양수를 반환

인잔인 파일열기 종류에는 텍스트 파일인 경우 “r”, “w”, “a”등의 종류가 있다.



파일 스트림 열기

- 읽기 모드 r은 읽기가 가능한 모드이며, 쓰기 불가능
- 쓰기모드 w는 파일 어디에든 쓰기가 가능한 모드이나 읽기는 불가능하다.
- 추가모드 a는 파일 중간에 써 쓸 수 없으며 파일 마지막에 추가적으로 쓰는 것만 가능한 모드이며 읽기는 불가능하다.



함수 `fclose()`로 파일 스트림 닫기

함수 `fclose()`는 `fopen()`으로 연결한 파일 스트림을 닫는 기능을 수행

그러므로 파일 스트림을 연결한 후 파일 처리가 모두 끝났으면 파일 포인터 `f`를 인자로 함수 `fclose()`를 호출하여 반드시 파일을 닫도록 한다.



파일에 서식화된 문자열 입출력

함수 `fprintf()`와 `fscanf()`


- 테스트 파일에 자료를 쓰거나 읽기 위하여 함수 `fprintf()`와 `fscanf()` 또는 `fscanf_s()`를 이용한다



파일 문자열 입출력

함수 `fgets()`와 `fputs()`

- 함수 `fgets()`는 파일로부터 한 행의 문자열을 입력받는 함수다.
- 함수 `fputs()`는 파일로 한행의 문자열을 출력하는 함수다.
- 함수 `gets()`는 파일로부터 문자열을 개행문자까지 읽어 마지막 개행문자를 '\0'문자로 바꾸어 입력 버퍼 문자열에 저장한다.
- 마찬가지로 함수 `fputs()`는 문자열을 한 행에 출력한다



함수 feof()와 ferror()

함수 feof()는 파일 스트림의 EOF 표시를 검사하는 함수다

함수 ferror()는 파일 처리에서 오류가 발생했는지 검사하는 함수다.



파일 문자 입출력

함수 `fgetc()`와 `getc()`는 파일로부터 문자 하나를 입력 받는 함수

함수 `fputc()`와 `putc()`는 문자 하나를 파일로 출력하는 함수다



텍스트와 이진 파일 입력과 출력

함수 `fprintf()`와 `fscanf()`, `fscanf_s()`는 자료의 입출력을 텍스트 모드로 처리

함수 `fwrite()`와 `fread()` 텍스트 파일과는 다르게 이진파일은 C 언어의 자료형을 모두 유지하면서 바이트 단위로 저장되는 파일

이진모드로 블록다위 입출력을 처리하려면 함수 `fwrite()`와 `fread()`를 이용한다.



순차 접근과 임의 접근

파일 위치는 파일 내부를 바이트 단위로 파일 내부 위치를 나타내는 값이다. 파일지시자 또는 파일 표시자라고 한다

파일의 마지막에는 파일의 마지막임을 알리는 EOF 표시가 있다.

파일을 처음으로 열면 모드에 관계없이 파일 위치는 모두 0이다.



파일 순차적 접근과 임의 접근

파일 위치를 처음부터 하나씩 증가시키면서 파일을 참조하는 방식을 순차적 접근

순차적 접근과는 다르게 파일의 어느 위치든 바로 참조하는 방식을 임의 접근이라 함



파일의 임이 접근 함수

fseek() 파일의 임의 접근을 처리하기 위해서는 파일 위치를 자유 잣로 이동하는 함수 fseek()이 필요

- 함수 fseek() 에서 세 번째 인자는 오프셋을 계산하는 기준으로 정수형 기호 상수로 다음 세 가지 중의 하나를 이용할 수 있다.

ftell()은 인자인 파일의 파일 위치를 반환하며, 함수 rewind()는 파일 위치를 무조건 가장 앞으로 이동시킴



파일 열기 다양한 모드

fopen과 fopen_s()의 인자인 파일열기 종류에는 텍스트 파일인 경우 “r”, “w”, “a”, “r+”, “w+”, “a+” 등의 종류가 다.

- 읽기모드 r은 읽기가 가능한 모드이며 쓰기는 불가
- 쓰기모드 w는 파일 어디에든 쓰기가 가능한 모드이나 읽기는 불가
- 추가 모드 a는 파일 중간에 쓸수 없으며 파일 마지막에 추가적으로 쓰는 것만 가능, 읽기는 불가능하다.

파일모드에서 +의 삽입은 수정 모드 의미로 원래의 모드에서 읽기 또는 쓰기가 추가되는 모드이다.

모든 파일모드 전환 사이에는 fflush()와 fseek()또는 rewind()와 같은 함수 호출이 반드시 필요함.



이진 파일

함수 `getw()`와 `putw()`는 워드 크기의 `int` 형 정수를 파일에 이진모드로 입출력하는 함수이다.

파일처리 함수 `remove`는 지정된 특정파일을 삭제하고, `rename` 함수는 지정된 파일 또는 폴더의 이름을 바꾸는 역할을 수행