

목 차

제1장 프로그래밍 시작하기 - 4

1. 프로그램과 프로그래밍	5
2. 프로그램의 5대 기본 구성 요소	7
3. C프로그램의 실행 흐름	8
4. C 프로그램 첫 예제 살펴보기	9
5. Escape Sequence (확장문자열-기능문자)의 이해	11
6. 상수와 변수의 이해	12
7. 프로그램의 전처리, 컴파일, 링크의 이해	15

제2장 기초 제어문 (if, while, break) - 15

1. 문장의 종류	17
2. 택일문(선택문)의 기본이 되는 if문의 이해	18
3. while문의 이해	21

제3장 데이터형 - 25

1. 기본 데이터형의 종류 및 데이터형의 범위	26
3. 정수형의 이해	29
4. 실수형의 이해	31
5. 문자형의 이해	34
6. 문자열형의 이해	35
7. 형변환의 이해	36

제4장 표준 입출력 함수 - 38

1. 데이터형과 형식변환문자	39
2. 표준출력함수 printf()	40
3. 표준입력함수 scanf()	44
4. 단일문자 입력 함수(버퍼화 입력의 의미)	47

제5장 제어문의 이해 - 49

1. 또 하나의 택일문 switch ~ case 문의 이해	50
2. 또 다른 반복문 for와 do~while의 이해	52
3. 분기문의 이해	55

제6장 연 산 자 - 57

1. 연산자의 기본개념	58
2. 연산자 우선순위표	58
3. 연산자 기능의 이해	59

제7장 함수(Function)의 기본 활용 - 66

1. 함수의 이해	67
2. 함수 전달인자와 매개변수	71
3. 함수 호출 기법	72
4. 재귀호출 함수	74

제8장 배열 - 77

1. 배열의 기본 개념	78
2. 1차원 배열의 선언 및 초기화	78
3. 배열에 file내의 데이터 저장하기	82
4. 2차원 배열의 선언 및 초기화	84
5. 배열과 증감연산자	85
6. 배열선언 및 초기화 시의 주의점	87

제9장 기억클래스 - 88

1. 기억클래스(Storage class)의 기본 개념	89
2. auto 기억 클래스의 이해	90
3. extern 기억클래스의 이해	91
4. static 기억클래스의 이해	93
5. register 기억클래스의 이해	95
6. 프로젝트	96

제10장 선행처리자 - 97

1. 선행처리자란?	98
2. #include	98
3. #define[매크로 정의]	99
4. 기타 다양한 선행처리자(preprocessor)	103

제11장 포인터 - 106

1. 포인터의 기본개념	107
2. 1차원 포인터	107
3. 1차원 배열과 포인터	109
4. 다차원 포인터	112
5. 포인터배열과 배열을 가리키는 포인터	114
6. 동적메모리 할당의 이해 및 활용	116
7. 함수 포인터	118
8. void형 포인터	120
9. const의 이해	122

제12장 구조체 (Structure) - 124

1. 구조체의 기본개념	125
2. 구조체 배열과 구조체 포인터	126
3. typedef를 이용한 다양한 타입 재정의 및 활용 기법	128
4. 구조체를 함수로 보내기	132
5. 구조체 Padding	135
6. Bit Field의 기본개념 및 특징	137
7. 공용체(union) 및 나열형(enum)	138

제13장 파일 입출력 - 141

1. 파일 입출력의 기본개념	142
2. 고수준 입출력 함수	144
3. 고수준화일 I/O와 저수준화일 I/O의 차이	150
4. 저수준 입출력 함수	151

제14장 기초 자료구조 & 알고리즘 - 153

1. 연결리스트(Linked List)	154
2. 스택(Stack)	159
3. 큐(Queue)	162

과제집 - 166**ASCII Code Table - 217**

제1장 프로그래밍 시작하기

학습 내용 소개

1장에서는 프로그램의 의미와 프로그래밍에 필요한 요소를 배우며, 프로그래밍의 기본 흐름인 로직(Logic) 개념을 이해하고 간단한 C프로그램 분석을 통해 C프로그램의 기본 요소와 용어를 익힌다.

학습 목차

1. 프로그램과 프로그래밍
2. 프로그램의 5대 기본구성요소
3. C 프로그램의 실행 흐름 이해
4. C 프로그램 첫 예제 살펴보기
5. Escape Sequence(기능 문자)의 이해
6. 상수와 변수의 이해
7. 프로그램의 전처리, 컴파일, 링크의 이해

1. 프로그램과 프로그래밍

1.1 프로그램 & 프로그래밍

- 프로그램 : 순서에 따라 한 번에 하나씩 실행되는 명령어로 구성되어 있어, 컴퓨터가 수행해야 하는 절차를 나타낸다.
- 프로그래밍 : 주어진 문제를 해결하기 위한 처리 방법과 순서를 정하는 작업
예) 하나의 정수를 입력 받아 그 정수의 2배 값을 출력하는 프로그램 작성하시오.

1.2 프로그래밍 언어란?

일의 처리방법과 순서를 컴퓨터가 알아듣고 실행할 수 있도록 명령을 내려줄 때 사용하는 언어. 단, 실제로 컴퓨터가 명령을 수행하기 위해서는 컴퓨터가 정확하게 알아들을 수 있는 기계어로 프로그램을 변환해주는 작업을 거쳐야 함.

예) Assembly, , C, C++, Java, Python, C#...

1.3 C언어의 탄생배경

1. ALGOL60 (ALGORITHMIC Language)
1960년 국제위원회에서 발표한 추상적이고 범용적인 언어, 모듈화 가능
2. CPL(Combind Programming Language)
1963년 영국 케임브리지 및 런던 대학교 팀이 발표한 언어로 ALGOL60의 추상적인 면을 보완
3. BCPL(Basic Combind Programming Language)
1967년 케임브리지의 Martin Richards 가 발표
4. B
1970년 Bell 연구소의 Ken Thompson 이 개발, BCPL을 간소화
5. C
1972년 Bell 연구소의 Dennis Ritch 가 개발

1.4 C언어의 특징

- ① C언어는 모든 언어의 기본이 되는 언어이다.
- ② 효율성 : 프로그램의 크기가 작고 실행 속도가 빠름, 메모리를 가장 효율적으로 사용 가능
- ③ 탁월한 이식성 : 특정 시스템에서 작성한 C코드를 다른 운영체제에서 사용하고자 할 때 최소한의 수정으로 사용이 가능하다.
- ④ 하드웨어 제어 및 로봇 제어 가능

1.5 프로그래밍의 7단계

- ① 프로그램 목적 정의 : 어떤 작업을 할 것인지 결정.
 - 프로그램이 요구하는 정보
 - 프로그램이 수행할 계산 및 조작
 - 프로그램이 사용자에게 제공할 정보를 정한다.
- ② 프로그램 설계 : 프로그램을 어떻게 만들 것인지 결정.
 - 사용자 인터페이스 결정(화면 설계)
 - 프로그램 내의 데이터 표현 방법 결정(데이터형 설계)
 - 보조 파일에서의 데이터 표현 방법 결정(파일 설계)
 - 데이터 처리 방법 결정 (함수 설계)

※ 복잡한 핵심 로직은 Flowchart나 NS-Chart로 표현 함
- ③ 코드 작성 : 설계한 프로그램의 실제 구현. 즉, 프로그램 설계를 C 언어로 번역하는 것.
- ④ 컴파일 : 소스코드(source code)를 실행 코드로 변환 시킴
Preprocessing -> compiling -> linking 의 과정을 걸쳐 실행 파일(executable file) 생성
- ⑤ 프로그램 실행 : 운영체제 상에서 파일 이름이나 아이콘을 더블 클릭하여 실행 또는
IDE(Integrated development environment : 통합개발환경)에서 메뉴를 통해서 실행
- ⑥ 테스트 및 디버깅 : 실질적인 데이터로 프로그램의 정확한 수행여부를 검사하고, 프로그램 내에 존재하는 오류(bug)를 찾아 수정(debugging)하는 과정
- ⑦ 유지보수(maintenance) : 사용중인 프로그램에 대해서 새로운 기능을 추가시키거나 다른 기종의 컴퓨터에서 실행시키기 위해서 기존의 프로그램을 수정하는 것.

※ C 프로그래밍 과정에서 발생하는 오류의 종류

- Warning : 코드 내의 문법이나 데이터 형의 사용 중 모호한 부분에 대한 경고
- Compile Error : C 문법을 잘못 사용하여 발생하는 오류
- Link Error : 전역변수 정의 및 함수 선언 및 정의가 잘못 되어있는 경우 발생하는 오류
- Run Time Error : 프로그램 실행 도중 발생하는 오류(주로 메모리를 관련 오류)
- Logic Error : 프로그램의 제어문이나 데이터 형의 잘못된 사용 등의 이유로 프로그램이 비정상적인 결과로 수행되는 오류

2. 프로그램의 5대 기본 구성 요소

- ① 변수(variable) : 기억공간
- ② 상수(constant) : 값 (3, 2.5, 'a', "sky" ...)
- ③ 연산자(operator) : 프로그램 내에서 연산을 할 때 사용하는 기호 (+, -, *, /, = ...)
- ④ 예약어(reserved word) : 프로그래밍 언어 중에서 특별한 의미로 고정되어 사용되는 단어로
사용자가 임의로 바꾸어 사용할 수 없는 명령어 (int, if, for ...)
- ⑤ 함수(function) : 변수, 상수, 예약어, 연산자 등을 이용하여 만든 한가지 이상의 기능을 가진 짧은 프로그램

예) 다음의 코드 중에서 5가지 기본구성요소를 분류해보시오.

```
int number;
number = 3;
printf("number=%d",number);
```

(함수의 2분류)

- Library Function :

- User Define Function :

(함수의 특징)

- 외형적인 측면의 특징 :

- 수행측면의 특징 : ① call
② return

(함수 사용을 위해 필요한 3요소)

- 함수 선언부
- 함수 호출부
- 함수 정의부

(Coding 시 유의점)

- C program은 프로그램의 기능을 함수 단위로 나누어 작성한다.
: 입력기능, 계산기능, 출력기능 등
- C program은 소문자로 작성한다. (대소문자 구분)
: 반드시 소문자로 작성해야 하는 것 : 예약어, main 함수명, 라이브러리 함수

3. C프로그램의 실행 흐름

3.1 C프로그램의 실행 흐름 파악하기

[예제 1-1] 다음 프로그램의 결과를 예측하시오.

```

1. preprocessor → #include<stdio.h> 선언부
2. (전처리기) void func1();
3.    void func2();
4.    void func3();
5.    void func4();
6.
7. start up → int main()
8.    {
9.        func2();
10.       func1();
11.       return 0;
12.    }
13.
14. void func1()
15. {
16. 실행    printf("func1() 함수 수행 중...\n");
17.       func3();
18.       return;
19. }
20. void func2()
21. {
22.       printf("func2() 함수 수행 중...\n");
23.       return;
24. }
25. void func3()
26. {
27.       printf("func3() 함수 수행 중...\n");
28.       return;
29. }
30. void func4()
31. {
32.       printf("func4() 함수 수행 중...\n");
33.       return;
34. }
```

call → *정의부*
return ←

4. C 프로그램 첫 예제 살펴보기

[예제 1-2] 첫 예제

```

1.      /* 작성자 : 정해경
2.         프로그램의 기능 : C프로그램의 기본 구조 소개 */
3.      #include<stdio.h>
4.      void printNumber(int);
5.      int main()
6.      {
7.          int num;
8.          num = 1;
9.          printNumber(num);
10.         num = 3;
11.         printNumber(num);
12.         return 0;
13.     }
14.     void printNumber(int n)
15.     {
16.         printf("정수값은 %d입니다.\n", n);
17.         return;
18.     }

```

1~2 line : /* 작성자 : 정해경 */

C 언어에서 주석문은 프로그램의 이해를 돕기 위한 해설 등에 사용되며 컴파일 대상에서 제외된다.

/* ... */ 여러줄 주석
// 한 줄 주석

① 저작권

② 복잡한 알고리즘

③ 특정한 방식만 사용하는 코드

- Comment out : 실행 가능한 code를 잠시 실행 안되도록 막아두는 것

3 line : #include <stdio.h>

#include는 선행처리기 명령(Preprocessor 지시자) 이다.

이는 Preprocessor에 의해 처리되며 "stdio.h" 라는 헤더파일을 현재 위치에 포함 시킨다.

4 line : void printNumber(int);

printNumber()함수의 선언부분

5~6 line : int main()

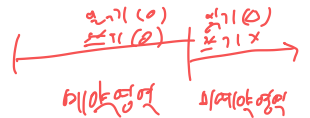
(C언어에서 사용되는 4가지 괄호)

사양내기 중대

- < > : 헤더파일 포함(include)시 사용
- () : 소괄호
 - ① 함수를 나타내기 위해 사용되며, ② 수식의 우선순위를 변경할 때도 사용된다
 - 예) int main() , 5*(3+2) ③ 조건식은 나타낼 때
- { } : 중괄호
 - Block의 시작과 끝 표시, 또는 복합 데이터형의 데이터 초기화 시에 사용된다
 - 예) int ary[5] = {1,2,3,4,5}
- [] : 대괄호
 - 배열선언 및 배열요소 지정에 사용된다
 - 예) int a[5]; a[0] = 1; a[1] = 2; a[2] = 3; a[3] = 4; a[4] = 5;

※ 배열의 특징

- ① 배열은 동일한 자료형의 값
- ② 연속된 메모리시 한 덩어리
- ③ 배열명은 곧 그 배열의 시작주소



7~8 line : int num; : 변수 선언

num 이라는 이름의 int형(정수형) 변수 생성.
 num = 1; : 치환 - 생성된 변수 num에 숫자상수 1을 대입한다
 int num = 1; 의 형태처럼 한 줄로 쓸 수 있다.

9, 11 line : printNumber(num); : printNumber()함수의 호출 부분

14~18 line : printNumber(int n) {...} : printNumber함수 정의부

16 line : printf("정수값은 %d입니다.\n", n);

printf() 는 표준 출력함수로서 괄호 안에 지정되어있는 출력형식대로
 화면(모니터)에 출력한다

- %d :
- "\n":

12,17 line : return 문

return 0; <=== 함수 수행을 모두 끝내고 리턴 할 때 0값을 갖고 제어를 되돌려 줌
 return; <=== 함수 수행을 모두 끝내고 리턴 할 때 리턴 값 없이 제어만 돌아 감

online 강의 2차 순회

5. Escape Sequence (확장문자열-기능문자)의 이해

'\ ' (역 슬래시) + 선행문자 : Escape Sequence로서 기능으로 출력된다.

표기	문자의 이름	의 미
\n	New line	커서를 다음 행의 첫 칸으로 이동 (개행 기능)
\t	Tab	다음 탭의 첫 칸으로 커서를 이동 시킨다
\b	Back space	커서를 한 칸 앞으로 이동시킨다
\r	Carrige return	커서를 현재 행의 첫 칸으로 이동시킨다
\a	Bell	beep음을 울린다
\\	역슬래시	역 슬래시 하나를 출력한다
\"	이중 인용부호	이중인용부호를 출력한다
\ooo	8진수로 표현	8진수로 ASCII 코드 값을 부여 W101
\xhh	16진수로 표현	16 진수로 ASCII 코드 값을 부여 Wx41

[예제 1-3] Escape Sequence 예제 프로그램

```

1.  /* Escape Sequence 의 기능을 살펴보는 예제 */
2.  #include <stdio.h>
3.  #include <conio.h>
4.  int main()
5.  {
6.      printf("Hello");
7.      printf("Hello\t");
8.      printf("Hello\n");      getch();
9.      printf("Korea\r");      getch();
10.     printf("C\n");           getch();
11.     printf("King\b\b\b");     getch();
12.     printf("ong\n");          getch();
13.     printf("12345678123456781234567812345678\n");  getch();
14.     printf("1\t123\t12345\t1234567\n");  getch();
15.     printf("\\, \", \n");  getch();
16.     return 0;
17. }

```

HelloHello Hello
 Korea
 kong
 12345678 ~ 78
 1 123 12345 1234567
 \ "

6. 상수와 변수의 이해

프로그램이란 각종 자료(data)를 사용자의 요구에 맞게 처리한 후 그 결과를 얻는 것을 말한다. 그렇다면 프로그램 내에서 처리해야 할 데이터는 어디에 저장되어있을까?

- 상수(constant)

프로그램 처리에 필요한 각종 자료(data)를 말하며 프로그램 시작부터 끝까지 변하지 않는 값을 의미한다

- 변수(variable)

프로그램 실행 중에 상수 값을 기억시키고 그 값을 변경하기 위해 Memory상의 임의의 위치에 필요크기만큼 확보된 기억공간을 의미한다. 프로그램 실행 중에 변경 가능하다. C언어에서 모든 변수는 반드시 블록의 선두에서 미리 선언되어야 한다

- 식별자 (identifier)

사용자가 정의하여 사용하는 이름 예) 변수명, 배열명, 사용자 정의함수명, 구조체 형명 등

(식별자 작성규칙)

- . 영문자('a'~'z', 'A'~'Z'), 밑줄('_'), 숫자('0'~'9')문자를 사용하며 첫 글자는 반드시 영문자나 밑줄이 와야 한다
- . 대소문자를 구별한다 (num, Num, NUM 은 모두 다르게 취급된다)
- . 예약어 및 main() 함수명은 식별자명으로 사용할 수 없다(void, int, if)
- . 특수문자와 공백은 사용할 수 없다
- . 한 영역내의 식별자명은 유일해야 된다
(즉, 한 블록 내에 동일한 이름의 변수명이나 함수명을 선언하여 사용할 수 없음)
- . 일반적으로 변수명, 사용자 정의 함수명은 소문자를 사용하며, 매크로명은 대문자를 사용한다.
- . 컴파일러와 리소스는 두 개의 밑줄 문자로 시작하는 변수명이나, 밑줄 문자와 대문자로 시작하는 변수명을 사용하므로 사용자는 이런 식별자명의 사용은 피하는 것이 좋다.

6.1 C언어의 5대 상수

① 숫자상수 : 정수형상수(10진수, 8진수, 16진수)와 실수형상수(10진수)가 있다.

- 정수상수 : 0, 3, 123, -53, 012, 0x2B (8진수, 10진수, 16진수 사용)
- 실수상수 : 12.5, 0.12325E2 ... (10진수만 사용)

2차원 배열

② 문자상수 : 단일인용부호를 이용해서 표현하며, 메모리에 한 Byte 크기로 할당된다.

'3', 'A', '\n', '' ..
(b 32)

③ 문자열상수 : 이중인용부호를 이용해서 표현하며, 문자열의 끝표시인 '\0'(NULL 문자)로 종료하는 데이터.

"3", "ABC", "A B", "Hello World", "" ...

④ 심볼릭(symbolic) 상수 : 매크로상수(전처리기에 의해 처리되는 #define 명령문에 의해서 지정되는 상수로서 매크로상수는 치환할 문자열로 치환되어 컴파일 된다), 열거형 상수, const변수

```
#define PI 3.14
#define PC "personal computer"
enum {KOR, ENG, MAT};
```

⑤ 주소상수 : 변수의 시작주소, 배열의 시작주소, 배열원소의 시작주소, 함수의 시작주소

▶ C언어의 모든 상수와 변수에는 차원이 부여되어있으며 차원에 따라 상수와 변수는 아래와 같이 이 분류 된다. (C언어의 최저 차원은 0차원 임)

* 상수의 2분류

- 일반상수 : 숫자상수, 문자상수, 매크로상수 일부(#define PI 3.14) *0차원*
- 주소상수 : 문자열상수, 주소상수, 매크로상수 일부(#define S "smile") *적어도 1차원*

* 변수의 2분류

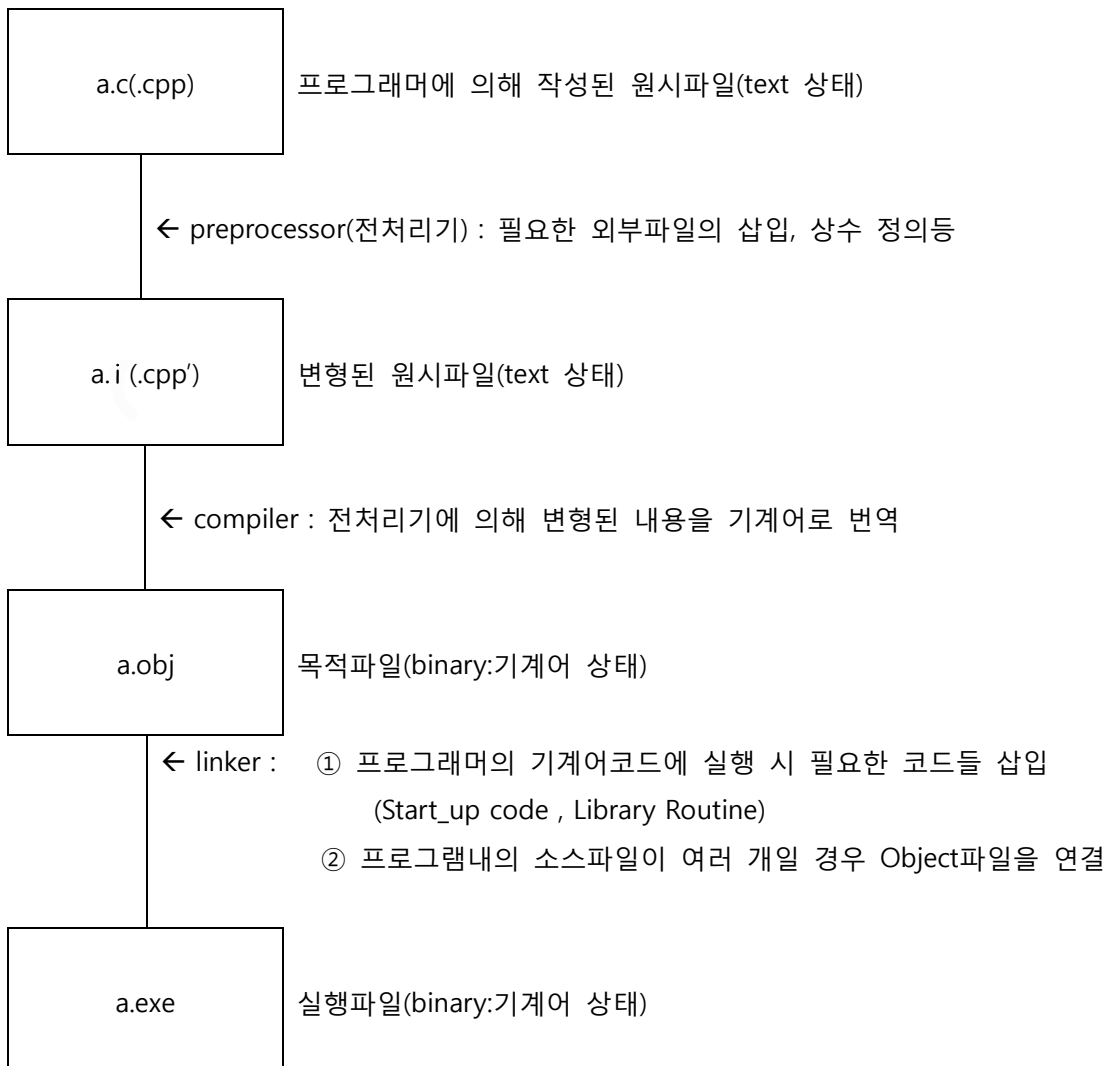
- 일반변수 : 일반상수 저장 *0차원*
- 포인터변수 : 주소상수 저장 *1차원*

[예제 1-4] 함수의 사용 더 보기

```
1.  #include<stdio.h>
2.  #define DAYS 365
3.  int input(void);
4.  void output(int, int);
5.  int calculate(int, int);
6.  int main()
7.  {
8.      int number;    // 입력 정수 저장변수
9.      int result;    // 계산 결과 저장변수
10.
11.      number = input();
12.      result = calculate(number, DAYS);
13.      output(number, result);
14.
15.      return 0;
16. }
17.
18. int input(void)
19. {
20.     int num;
21.     printf("* 나이를 입력하세요 : ");
22.     scanf("%d", &num);
23.     return num;
24. }
25.
26. int calculate(int num, int days)
27. {
28.     int totalDays;
28.     totalDays = num * days;
29.     return totalDays;
30. }
31.
32. void output(int num, int tot)
33. {
34.     printf("나이는 %d살이고, 살아온 총날 수는 약 %d일 입니다.\n", num, tot);
35.     return;
36. }
```

) 함수 선언부

7. 프로그램의 전처리, 컴파일, 링크의 이해



제2장 기초 제어문 (if, while, break)

학습 내용 소개

- ☞ C언어의 문장 종류에 대해 알아보자.
- ☞ 기본 제어문의 구조 및 사용법을 익혀보자.

학습 목차

1. 문장의 종류
2. 택일문(선택문)의 기본, if문
3. while문의 이해

1. 문장의 종류

Preprocessor 지시문		#include, #define, #pragma 등
선언문		char, int, long, float, double 등
제어문	택일문	if, if~else, if~ else if ~ else, switch~case
	반복문	for, while, do~while
	분기문	break, continue, goto, return
기타		공문 (NULL statement)

2. 택일문(선택문)의 기본이 되는 if문의 이해

1) if문이란?

- 조건식에 따라 실행문을 실행 하느냐 마느냐를 결정한다. 즉, 실행문이 여러 개일 경우 조건에 맞는 실행문 1가지만을 실행하기 때문에 택일문 또는 선택문이라 한다.

2) 1형식 if문

if(조건식)	/*조건식 : 결과가 참 또는 거짓이 나오는 식*/
{	
처리문장;	/*조건식의 결과가 참일 경우 수행되는 문장*/
}	
다음문장;	/*if문과 상관없는 별개의 문장*/

[예제 2-1] 정수를 입력받아 양수일 경우 출력하는 프로그램

① 논리흐름

1. 시작
2. 입력 받은 data를 저장할 변수 선언(정수형)
3. 변수에 data를 입력받음
4. 만약 입력받은 data가 양수면 출력
5. 종료

② Source code

```

1. #include<stdio.h>
2. int main()
3. {
4.     int indata;
5.     scanf("%d",&indata);
6.     if(indata>0)
7.     {
8.         printf("%d",indata);
9.     }
10.    return 0;
11. }
```

3) 2형식 if문

```

if(조건식)                /*조건식 : 결과가 참 또는 거짓이 나오는 식*/
{
    처리문장1;            /*조건식의 결과가 참일 경우 수행되는 문장*/
}
else
{
    처리문장2;            /*조건식의 결과가 거짓일 경우 수행되는 문장*/
}

```

[예제 2-2] 사과무게를 입력받아 300g이상이면 1000원, 300g미만이면 500원을 출력하는 프로그램

① (논리흐름)

1. 시작
2. 사과무게를 저장할 정수형 변수 weight 선언
3. 사과무게를 입력 받음
4. 사과무게를 비교
 - 4-1. 사과무게가 300g이상이면 사과가격 1000원 출력
 - 4-2. 사과무게가 300g미만이면 사과가격 500원 출력
5. 종료

② Source Code

```

1. #include<stdio.h>
2. int main()
3. {
4.     int weight;
5.     scanf("%d",&weight);
6.     if(weight>=300)
7.     {
8.         printf("사과가격 1000원");
9.     }
10.    else
11.    {
12.        printf("사과가격 500원");
13.    }
14.    return 0;
15. }

```

4) 3형식 if문

```

if(조건식1)
{
    처리문장1;      /*조건식 1이 참일 때 수행하는 문장*/
}
else if(조건식2)
{
    처리문장2;      /*조건식 2가 참일 때 수행하는 문장*/
}
else
{
    처리문장3;      /*조건식1,2 모두가 거짓일 때 수행하는 문장*/
}

```

[예제 2-3] 사과의 무게를 입력 받아 200g 이하는 300원, 400g이하이면 500원, 400g초과면 700원으로 출력하시오.

① 논리흐름

1. 시작
2. 사과무게 저장할 정수형 변수 weight ,가격 저장할 정수형 변수 price선언
3. 사과무게를 입력 받음
4. 사과무게를 비교
 - 4-1. 사과무게가 200g이하이면 사과가격 300원 출력
 - 4-2. 사과무게가 400g이하이면 사과가격 500원 출력
 - 4-3. 400g 초과이면 사과 가격을 700원으로 출력
5. 종료

② Source Code

```

1. #include<stdio.h>
2. int main()
3. {
4.     int weight, price;
5.     scanf("%d",&weight);
6.     if(weight<=200)
7.     {
8.         price=300;
9.     }
10.    else if(weight<=400)
11.    {
12.        price=500;
13.    }
14.    else
15.    {
16.        price=700;
17.    }
18.    printf("%d",price);
19.    return 0;
20. }

```

2. while문의 이해

1) while문이란?

- while문은 조건이 참인 동안 반복시키는 선조건비교 순환(Loop)문이다.

2) while문의 형식

```
while(조건식)
{
    반복할문장;
}
```

3) 사용 예

[예제 2-4] 정수를 반복적으로 입력 받아 누적하다가 0이 입력되면 입력을 중단하고 총 누적 값을 출력.
(폴이-1)

① 논리흐름

1. 시작
2. 입력 받은 정수를 저장하는 변수 선언(정수형)
3. 덧셈의 누적변수 선언 후 0으로 초기화(정수형)
4. 정수 값 입력받기
5. 입력 값이 0이 아닐 동안 반복
 - 5-1. 덧셈의 누적 변수에 입력값 누적하기
 - 5-2. 정수 값 입력받기
6. 총 누적 값 출력하기
7. 종료

② Source code

```
1. #include<stdio.h>
2. int main()
3. {
4.     int indata, sum=0;
5.     scanf("%d",&indata);
6.     while(indata!=0)
7.     {
8.         sum= sum+indata;
9.         scanf("%d",&indata);
10.    }
11.    printf("%d",sum);
12.    return 0;
13. }
```

(풀이-2)

① 논리흐름

1. 시작
2. 입력 받은 정수를 저장하는 변수 선언(정수형)
3. 덧셈의 누적변수 선언 후 0으로 초기화(정수형)
4. 무한 Loop 돌리기
 - 4-1. 정수값 입력 받기
 - 4-2. 입력 data가 0이면 무한 Loop 탈출
 - 4-3. 덧셈의 누적변수에 입력값 누적
5. 총 누적 값 출력하기
6. 종료

② Source code

```
1. #include<stdio.h>
2. int main()
3. {
4.     int indata,sum=0;
5.     while(1)
6.     {
7.         scanf("%d",&indata);
8.         if(indata==0){break;}
9.         sum+=indata;
10.    }
11.    printf("%d",sum);
12.    return 0;
13. }
```

[예제 2-5] 임의의 수를 입력 받아 계속적인 합 출력하기 - 상황에 따른 반복

(숫자 0 입력시 종료)

```
1. #include<stdio.h>
2. int main()
3. {
4.     int k, s = 0;
5.     printf("정수를 입력하세요. 0(zero)을 입력하면 종료됩니다 : ");
6.     scanf("%d", &k);
7.     while (k != 0)
8.     {
9.         printf("지금까지 합은 : %d 입니다 \n", s += k);
10.        printf("수를 입력하세요 0(zero)을 입력하면 종료됩니다 : ");
11.        scanf("%d", &k);
12.    }
13.    return 0;
14. }
```

(문자 입력시 종료)

```
1. #include<stdio.h>
2. int main()
3. {
4.     int k, s = 0;
5.     printf("정수를 입력하세요.문자를 입력하면 종료됩니다 : ");
6.     while (scanf("%d", &k) == 1)
7.     {
8.         printf("지금까지 합은 : %d 입니다 \n", s += k);
9.         printf("정수를 입력하세요.문자를 입력하면 종료됩니다 : ");
10.    }
11.    return 0;
12. }
```

[예제 2-6] 구구단 2단을 출력 - 정해진 횟수의 반복

```
1. #include<stdio.h>
2. int main()
3. {
4.     int i = 1;
5.     while (i <10)
6.     {
7.         printf("2 * %d = %d\n", i, 2 * i);
8.         i++;
9.     }
10.    return 0;
11. }
```

[예제 2-7] 10에서 1까지 1씩 감소하면서 출력 - 변수값을 이용한 종료조건

```
1. #include<stdio.h>
2. int main()
3. {
4.     int num = 10;
5.     while (num)
6.     {
7.         printf(" %5d", num);
8.         num--;
9.     }
10.    return 0;
11. }
```

제3장 데이터형

학습 내용 소개

3장에서는 데이터처리에 필요한 데이터형에 대해서 체계적으로 배운다.

기본 데이터형과 복합 데이터형의 차이점을 알아보고 기본 데이터형의 이해와 사용시 발생할 수 있는 문제점들은 이해하고 해결하는 방법을 익힌다.

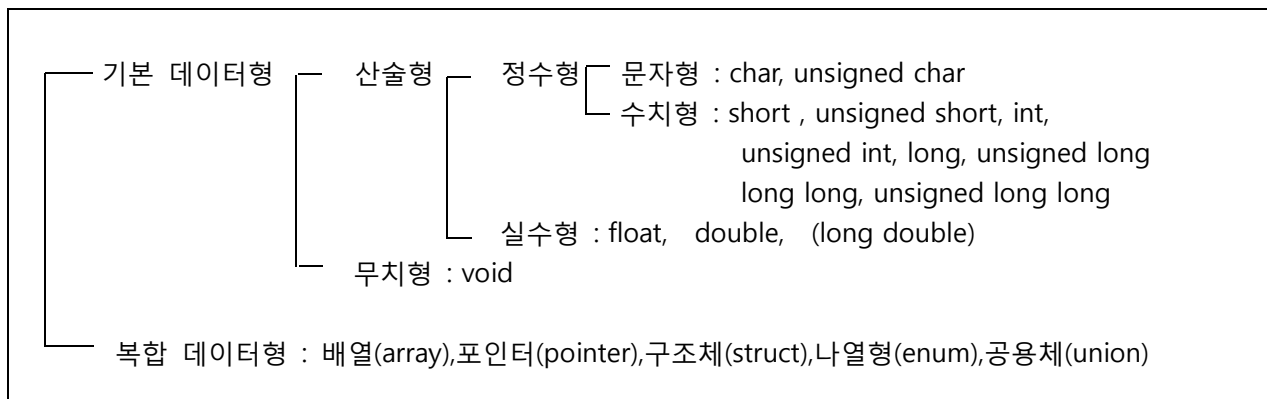
학습 목차

1. 상수와 변수의 이해
2. 기본 데이터형의 종류와 데이터형의 범위
3. 정수형의 이해
4. 실수형의 이해
5. 문자형의 이해
6. 문자열형의 이해
7. 형변환의 이해

1. 기본 데이터형의 종류 및 데이터형의 범위

- 기본 데이터형(primitive data type) : 프로그램 수행에 필요한 Memory를 할당 받기 위해 기본적으로 제공 되는 Data Type
- 복합 데이터형(compound data type) : 기본 데이터형을 응용해서 다양한 형태의 복합 데이터형을 만들어 사용한다.

1.1 데이터형의 구분



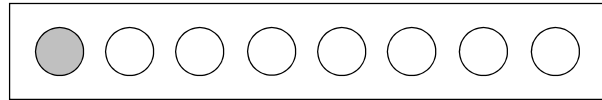
1.2 기본 데이터형의 범위

구 분	기본 데이터형	범 위	Byte
문자형	char unsigned char	-128 ~ 127($-2^7 \sim 2^7-1$) 0 ~ 255	1
수치형	int unsigned int	-2,147,483,648 ~ 2,147,483,647 ($-2^{31} \sim 2^{31}-1$) 0 ~ 4294967295	4
	short (int) unsigned short (int)	-32,768 ~ 32,767 ($-2^{15} \sim 2^{15}-1$) 0 ~ 65,535	2
	long (int) unsigned long (int)	-2,147,483,648 ~ 2,147,483,647 ($-2^{31} \sim 2^{31}-1$) 0 ~ 4294967295	4
	long long (int) unsigned long long (int)	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 0 ~ 18,446,744,073,709,551,615	8
	float double	$\pm 3.4 * 10^{-38} \sim \pm 3.4 * 10^{38}$ $\pm 1.7 * 10^{-308} \sim \pm 1.7 * 10^{308}$	4 8

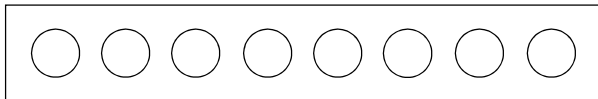
※ 일부 시스템에서는 int type이 2바이트인 경우도 있음.

1.3 signed와 unsigned의 이해

[signed char 형 기억공간의 형태]



[unsigned char 형 기억공간의 형태]



※ signed와 unsigned의 차이점은 Sign Bit가 있느냐 없느냐의 차이이다.

signed 데이터형은 음수, 0, 양수 모두를 나타낼 수 있으며, unsigned 데이터형은 0과 양수만을 나타내는 용도로 사용된다. Sign bit가 **0** 이면 양수, **1**이면 음수를 나타낸다.

1.4 데이터 표현에 사용되는 3가지 진법의 이해

- ① 10진법 : 0이 아닌 수로 시작되는 수 예) 1, 30, 123, -5, -128
- ② 8진법 : 0을 접두어로 갖고, 0~7까지의 수로 표현되는 수 예) 01, 030, 0123, 0245
- ③ 16진법 : 0x를 접두어로 갖고, 0~9까지의 수와 10~15까지의 수에 대응되는 a ~ f의 문자로 표현되는 수 예) 0x123, 0xfffe, 0x423c

* 컴퓨터 내부에 저장되는 2진수 표기는 프로그래밍 시에는 허용되지 않음

* 2,8,10,16진법의 변환은 자유롭게 표현할 줄 알아야 함

▶ suffix : 숫자에 붙이는 접미사

- 정수형 상수의 기본형은 signed int 형이다.
- 정수형 suffix : u, l(L), ul(UL) 예) 100L, 0123L, 0xffL
- 실수형 상수의 기본형은 double형이다.
- 실수형 suffix : f(F), L *long double*

(%#)원형표기

[예제 3-1] 세가지 진법의 사용

```

1 :    #include <stdio.h>
2 :    int main()
3 :    {
4 :        int x = 10, y = 010, z = 0x10;
5 :        printf("x의 값 : %d %o %x\n", x, x, x);
6 :        printf("y의 값 : %d %o %x\n", y, y, y);
7 :        printf("z의 값 : %d %o %x\n", z, z, z);
8 :        return 0;
9 :    }
```

2. 정수형의 이해

- ▶ 정수형 데이터의 크기는 limits.h 헤더파일에서 확인 가능
 - ▶ 정수형의 종류 : char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, long long, unsigned long long(대표 type : int)
 - ▶ 저장방식
 - 양수 : 부호화 절대치 표현 / - 음수 : 2의 보수 표현
 - ▶ 2의 보수(2's complement) 만들기
 - ① 바꾸고자 하는 대상 숫자(절대값)를 2진수로 변환 (저장 자릿수 고려)
 - ② 위 결과를 다시 1의 보수로 변환
 - ③ 위의 ②번의 결과에 1을 더한다.
- ** 2의 보수를 만드는 간단한 방법 : 위에서 제시한 ①번의 방법으로 바꾼 후 bit열의 우측에서 좌측으로 처음 1이 나올 때까지는 그대로 bit열을 내려쓰고, 나머지는 1의 보수를 취한다.**

[예제 3-2] 최대값을 넘어가는 숫자의 표현

```

1:  #include <stdio.h>
2:  int main()
3:  {
4:      short x = 32767; // short 형의 최대값
5:      short y = x + 1; // short 형의 최대값보다 1큰 값을 y에 저장
6:      printf(" x = %hd\t y = %hd\n", x, y); // h는 short형 변경자
7:      return 0;
8:  }

```

* %hd 형식변환문자의 h는 short형 출력 시 사용되는 변경자

[예제 3-3] unsigned형 기억공간에 음수값을 저장한다?

```

1:  #include <stdio.h>
2:  int main()
3:  {
4:      short  x = -32767;
5:      unsigned short y = x ; // unsigned형 기억공간에 음수값을 저장하면?
6:      printf(" x = %hd \t y = %hu\n", x, y);
7:      printf(" x = %hu \t y = %hd\n", x, y);
8:      return 0;
9:  }

```

* 저장되어 있는 값과 상관없이 출력되는 값은 형식변환 문자열의 해석에 따른다.

그렇다면 굳이 signed와 unsigned형 기억공간을 구분하여 변수를 선언하는 것이 의미 있는 것 일까?

[예제 3-4] signed와 unsigned형으로 구분하는 것의 의미

```

1:  #include <stdio.h>
2:  int main()
3:  {
4:      short  x;
5:      unsigned short y;
6:      int  res1,res2;
7:      x = y = -1; // 65535y=65535, x=-1
8:      res1 = x * 3; // intres1=-3
9:      res2 = u.sy * int3; // intres2=195...
10:     printf("res1 = %d\n", res1);
11:     printf("res2 = %d\n", res2);
12:     return 0;
13: }

```

* 위의 예제에서는 x, y 변수가 각각 선언된 형태대로 x는 signed 형의 데이터로 인식되어 계산되며, y는 unsigned형의 데이터로 인식되어 계산된다.

3. 실수형의 이해

- ▶ 컴퓨터내의 실수저장방식에서는 오차가 발생하므로 "real"이 아니라 "float"이라 함.
- ▶ 실수형의 종류 : float, double (대표 type : double)
- ▶ 저장방식 : IEEE754 방식으로 저장
 - ① 단정도 부동소수(32bit - float)
 - sign bit : 1bit / 지수부 : 8bit / 가수부(유효숫자부) : 23bit
 - ② 배정도 부동소수(64bit - double)
 - sign bit : 1bit / 지수부 : 11bit / 가수부(유효숫자부) : 52bit
- signed bit : 가수부(유효숫자부)의 sign (양수 : 0, 음수 : 1)
- 지수부 저장방식 : 지수승 + Bias 값
(Bias 값 : 지수부 최대값 / 2 → float : 127, double : 1023)
- 가수부 저장 방식 : 2진수로 표현된 유효숫자를 2진수 정규화(유효숫자 첫 자리를 2의 0승 자리에 고정)한 후 소수점 이하 유효숫자만 가수부에 저장 함

(변환해 보기) 아래 변수들의 저장 형태를 비트 열로 표현해보시오.

```
double da = 0.5;
```

```
double db = 1.0;
```

```
double dc = -1.0;
```

```
double dd = 0.0;
```

▶ 표기방식

- 소수형 표기의 예 : 12.3, 0.256 (일반적인 부동소수점수 표기 방식)
- 지수형 표기의 예 : 1.824E+4 (매우 작거나 큰 수의 표기 방식)

▶ 유효정밀도

실수 데이터형에만 있는 개념으로 유효숫자를 몇 자리까지 저장할 수 있는가를 10진수 기준으로 나타낸다.

float 형 : 7 ~ 8 자리 / double 형 : 14 ~ 16 자리

▶ 실수형의 오차

- ① 유효정밀도에 의한 오차(round off 오류 or epsilon 오차)
- ② 2진수 변환 시 순환소수로 변환되어 발생하는 오차

[예제 3-5] 실수형 데이터의 출력

```

1 :    #include <stdio.h>
2 :    int  main()
3 :    {
4 :        double  d_num1 = 32.145e-1;
5 :        double  d_num2 = d_num1 + 10;
6 :        printf("d_num1 = %lf(%le)\n", d_num1, d_num1);
7 :        printf("d_num2 = %lf(%le)\n", d_num2, d_num2);
8 :        return 0;
9 :    }

```

(실행결과)

d_num1 = 3.214500(3.214500e+000)

d_num2 = 13.214500(1.321450e+001)

※ float형 출력 시 %f 사용하며, double형 출력 시에는 %lf 사용

[예제 3-6] 실수형이 비교

```

1 :    #include<stdio.h>
2 :    #include<math.h>    // fabs() 함수 헤더 선언부
3 :    #include<float.h>    // DBL_EPSILON 매크로 선언
4 :    int main()
5 :    {
6 :        float a,b;
7 :        printf("a=");
8 :        scanf("%f", &a);
9 :
10 :       printf("b=");
11 :       scanf("%f", &b);
12 :
13 :       printf("DBL_EPSILON = %.30lf\n", DBL_EPSILON);
14 :       // DBL_EPSILON = 0.0000000000000000222044604925031
15 :
16 :       if(fabs(a-b) < DBL_EPSILON)
17 :       //if(a==b)
18 :       {
19 :           printf(" 두 수는 같은 수!!\n");
20 :       }
21 :       else
22 :       {
23 :           printf("두 수는 다른 수!!\n");
24 :       }
25 :       return 0;
26 :   }

```

** 다음 코드의 수행결과는?

```

float x;
int count = 1;
for (x=100000001.0f ; x<=1000000010.0f ; x+=1.0f)
{
    printf("%d. x = %.30f\n", count++, x);
}

```

```

float x;
int count = 1;
for (x=0.1f ; x<=1.0f ; x+=0.1f)
{
    printf("%d. x = %.10f\n", count++, x);
}

```

4. 문자형의 이해

- ▶ char형은 1바이트 크기의 정수형으로 정수형 저장 방식을 그대로 따른다.
- ▶ 문자상수는 작은 따옴표(')로 묶으며 값은 그 문자에 대한 ASCII code값으로 저장된다.

char형의 변수는 하나의 문자를 표현하는데 사용되며 각 문자는 한 바이트에 저장된다.

문자상수	'0'	'1'	'2'	'3'	'9'
문자상수의 ASCII Code값	48	49	50	51	57
문자상수	'A'	'B'	'C'	'D'	'Z'
문자상수의 ASCII Code값	65	66	67	68	90
문자상수	'a'	'b'	'c'	'd'	'z'
문자상수의 ASCII Code값	97	98	99	100	122

프로그램내에서 문자에 대한 처리는 각 상수의 ASCII code 값으로 처리되므로 문자에 대한 연산('A' + 1 -> 'B', 'a' < 'b' ...) 이 가능하다.

[예제 3-7] 문자형 데이터의 처리

```

1 :    #include <stdio.h>
2 :    int  main()
3 :    {
4 :        char a = 'A', b = 'a', c = 'a' - 'A' ;
5 :        printf("ASCII code %c is %d\n", a, a);
6 :        printf("'a' + 1 = %c\n", 'a'+1);
7 :        printf("%d = '%c'\n", b - a, c);
8 :        printf("%c -> %d\n", b-32, 'a'-32);
9 :        return 0;
10 :    }

```

Handwritten annotations in red:

- Line 4: 65 above 'A', 97 above 'a', $97 - 65$ above 'a' - 'A'
- Line 6: 65 above 'a' in the format string, 66 below 'a'+1
- Line 7: 32 above 'a' in the format string, 97 above 'b', 65 above 'a'
- Line 8: 32 above 'b' in the format string, 65 above 'a'

5. 문자열형의 이해

- ▶ 문자열은 문자형 자료의 집합으로 char형 1차원 배열로 표현된다.
- ▶ 문자열 상수는 가변길이 상수이므로 반드시 데이터의 끝 위치를 나타내기 위해 NULL(' 0')문자로 종료되며 이중인용부호("")로 묶는다.

문자열 상수 "hello"는 'h' 'e' 'l' 'l' 'o' ' 0'의 집합이다

char str[6] = "Hello";

H	e	l	l	o	�0
str[0]	str[1]	str[2]	str[3]	str[4]	str[5]

[예제 3-8] 문자열형 데이터의 처리

```

1 :      #include <stdio.h>
2 :      int main()
3 :      {
4 :          char str[6] = "Apple";
5 :
6 :          printf("%s\n", str);
7 :          printf("%c\n", str[0]);
8 :          printf("%c\n", str[3]);
9 :          printf("%x\n", str);
10 :         return 0;
11 :     }
```

6. 형변환의 이해

▶ 형변환의 종류

- 자동형변환(묵시적 형변환) : 이항연산 시 두 피연산자의 type이 다른 경우 type을 맞추기 위해 또는 연산 시에 존재하지 않는 type을 연산에 적합한 type으로 바꾸기 위해 자동으로 일어나는 형변환 (자동형변환은 0차원의 기본 데이터형에서만 발생)
- 강제형변환(명시적 형변환) : 프로그래머가 프로그래밍 시 필요에 의해서 cast(형변환)연산자를 이용하여 실시하는 형변환

▶ 자동형변환의 원칙

- 작은형이 큰형으로 변환된다(최대값이 크기로 형의 크기를 구분함)
- 대입연산자 Rvalue는 Lvalue의 type으로 변환된다.
- 주소는 자동형변환 안됨(C언어에서는 경고 발생 후 실행됨, C++에서는 에러발생)

[32bit 응용프로그램 기준]

int(long) → unsigned int (unsigned long) → long long → unsigned long long → float → double



char, unsigned char

short, unsigned short

enum(나열형)

[예제 3-9] 형변환 예제

```

1 :    #include <stdio.h>
2 :    int main()
3 :    {
4 :        printf("1 / 2 = %lf\n", 1 / 2 ); // 무엇이 출력될까?
5 :        printf("1.0 / 2 = %lf\n", 1.0 / 2);
6 :        printf(" 'A' + 32 = %c(%d)\n", 'A' + 32, 'A' + 32);
7 :        printf(" 'a' - 32 = %c(%d)\n", 'a' - 32, 'a' - 32);
8 :        return 0;
9 :    }

```

[예제 3-10] 서로 다른 형의 대소 비교

```

1 :    #include<stdio.h>
2 :    int main()
3 :    {
4 :        int    x = -1;
5 :        unsigned int y = 1;
6 :        if ( x >= y ) {
7 :            printf ( " True \n" );
8 :        }
9 :        else {
10:            printf ( " False \n" );
11:        }
12:        return 0;
13:    }

```

▶ 다음 프로그램의 결과를 예측하시오.

(형변환 예제1)

```

int add(int, int);
int main()
{
    int a=-1;
    unsigned int b = 0xffffffff;

    printf("%d\n", add(a,b));
    return 0;
}
int add(int a, int b)
{
    return (a+b);
}

```

(형변환 예제2)

```

int main()
{
    double da=3.5;
    if(-1<sizeof(da))
    {
        printf("True\n");
    }
    else
    {
        printf("False\n");
    }

    return 0;
}

```

(형변환 예제3)

```

int main()
{
    float fa=3.5;
    int a=3;
    if(&fa > &a)
    {
        printf("True\n");
    }
    else
    {
        printf("False\n");
    }
    return 0;
}

```

(형변환 예제4)

```

void sub(int *);
int main()
{
    double a=10.5;

    sub(&a);

    return 0;
}
void sub(int *p)
{
    printf("p가 가리키는 것:%d\n",*p);
}

```

제4장 표준 입출력 함수

학습 내용 소개

4장에서는 표준입출력 함수에 대하여 공부하는 장입니다.
printf()함수와 scanf()함수의 동작원리를 정확히 이해하고 다양한 형태로 활용하는
기법을 배웁니다.

학습 목차

1. 데이터형과 형식변환문자
2. 표준출력함수 printf()
3. 표준입력함수 scanf()
4. 단일문자입력 함수(버퍼화입력의 의미)

1. 데이터형과 형식변환문자

변환 문자	변환 내용
%d(%i)	인수를 부호 있는 10진수로 변환
%o	인수를 부호 없는 8진수로 변환
%x	인수를 부호 없는 16진수로 변환(0x)
%X	인수를 부호 없는 16진수 대문자로 변환(0X)
%u	인수를 부호 없는 10진수로 변환
%s	인수를 문자열로 출력
%f	인수를 실수형태로 변환(0.000000)
%e	인수를 지수형태로 변환(0.000000e±00)
%E	인수를 지수형태로 변환(0.000000E±00)
%g	%f 또는 %e의 변환 중 출력되는 문자수가 적은 쪽으로 변환
%G	%f 또는 %E의 변환 중 출력되는 문자수가 적은 쪽으로 변환
%c	인수를 한 문자로 변환
%p	인수를 16진수 8자리로 주소값 출력
%%	% 기호 출력

2. 표준출력함수 printf()

▶ printf()함수의 형식 : `int printf(const char *, ...);`

`printf("출력양식", 출력대상); /* 출력대상은 생략 가능함 */`

예) `printf("a = %d\n", a);`

① 출력양식의 3요소

- 일반출력문자 (a =) :
- Escape Sequence (\n) :
- 형식변환문자 (%d) :

② 출력대상의 4요소

- 변수 : `printf("%d", num); /* 변수내의 값을 출력 */`
- 상수 : `printf("%d", 100); /* 상수값을 그대로 출력 */`
- 수식 : `printf("%d", num+100); /* 수식의 연산결과를 출력 */`
- 함수리턴값 : `printf("%d", printf("Hello")); /* 함수의 리턴값 출력 */`

▶ printf()함수의 동작원리

printf()함수는 출력대상을 stack구조의 출력버퍼에 저장한 후 형식변환문자의 형태대로 버퍼에서 꺼내어 출력한다.

(printf buffer에 데이터 입출력 규칙)

- ① printf() 호출구문에 표기된 출력대상의 우측→좌측 방향으로 저장 됨
- ② int size로 입출력 됨 (char, short형은 int로 자동형변환되어 저장 됨)
- ③ 실수형(float, double)형은 double형으로 자동형변환되어 4Byte씩 나누어 두번에 걸쳐 입출력 됨

▶ printf() 함수의 변경자

변경자	의 미
10진수	최소의 필드너비 지정. 예) %5d 출력될 데이터가 지정된 크기에 맞지 않으면 크기 무시
.10진수	정밀도 예) %.2f %f,%e,%g에서는 소수이하 자리수를 의미 %s에서는 프린트될 최대 문자의 개수 %d에서는 출력될 수의 최소개수를 의미, 앞의 여유분에 '0'을 채움
h	short 형임을 표시 예) %hu
l	long형 또는 double형 임을 표시 예) %ld, %lf
ll	long long 형 임을 표시 예) %lld, %llu
L	long double 형임을 표시 예) %Lf %10.4Le
*	필드의 너비나 정밀도등을 임의로 지정하고자 할 때 사용 예) %*d %*.f

▶ printf() 함수의 플래그(flag)

플래그	의 미
-	항목이 필드의 왼쪽부터 시작하여 프린트된다 예) %-20s
+	수치 앞에 부호를 붙여 출력한다 예) %+6.2f
공백	수치가 양의 값이면 공백을 음의 값이면 -를 붙인다 예) % 6.2f
#	%o 에 대해서는 8진 접두어 0을 %x에 대해서는 16진 접두어 0x를 붙여 출력한다 예) %#o %#x
0 (zero)	수치값 출력 시 여분 공간을 0으로 채운다 예) %08d %08.3f

[예제 4-1] int형 변수값의 서식화 출력

```

1 :    #include <stdio.h>
2 :    #include <stdio.h>
3 :    int main()
4 :    {
5 :        int x = 12345;
6 :        printf("1234567890\n");
7 :        printf("=%d=\n",x);
8 :        printf("=%8d=\n",x);
9 :        printf("=%-8d=\n",x);
10:        printf("=%3d=\n",x);
11:        return 0;
12:    }
```

[예제 4-2] float형 변수값의 서식화 출력

```
1 :      #include <stdio.h>
2 :      int  main()
3 :      {
4 :          float  x = 22.17;
5 :          printf("1234567890123456\n");
6 :          printf("=%f=\n", x);
7 :          printf("=%10.4f=\n", x);
8 :          printf("=%-10.4f=\n", x);
9 :          printf("=%.2f=\n", x);
10:         return 0;
11:     }
```

[예제 4-3] 문자열의 서식화 출력

```
1 :      #include <stdio.h>
2 :      int  main()
3 :      {
4 :          printf("1234567890123456\n");
5 :          printf("=%10.5s=\n", "Happiness");
6 :          printf("=%-10.5s=\n", "Happiness");
7 :          printf("=%*.*s=\n", 15, 7, "Happiness");
8 :          return 0;
9 :      }
```

[예제 4-4] # 플래그 예제

```
1 :      #include <stdio.h>
2 :      int  main()
3 :      {
4 :          int  x = 10, y = 010, z = 0x10;
5 :          printf("10은 %#o(8) 와 같고 %#x(16)와 같다\n", x, x);
6 :          printf("010(8)은 %d와 같고 %#x(16)와 같다\n", y, y);
7 :          printf("0x10는 %#o(8) 와 같고 %d와 같다\n", z, z);
8 :          return 0;
9 :      }
```

▶ sprintf() 함수의 활용

printf()함수는 모니터에 출력하는 반면 sprintf()함수는 문자열에 출력하는 함수이다.

[예제 4-5] sprintf()함수의 활용 예

```
1 :   #include <stdio.h>
2 :   int  main()
3 :   {
4 :       char str[100];
5 :       double pi = 3.141592;
6 :       sprintf(str, "원주율 값은 %.21f 입니다.", pi);
7 :       printf("%s\n", str);
8 :       return 0;
9 :   }
```

3. 표준입력함수 scanf()

▶ scanf() 함수의 형식 : `int scanf(const char *, ...);`

`scanf("입력양식", 입력대상처의 시작주소);`

예) `int a;`
`scanf("%d", &a);`

① 입력양식의 구성요소

- 형식변환문자 (%d)
- 구분자 : 여러 개의 데이터 입력 시 데이터 사이를 구분하기 위한 문자
 (단, 마지막 형식변환문자 뒤에는 구분자를 쓰지 않는다)
- ** 기본 구분자 : space, tab, enter

② 입력 대상처의 시작주소

- 반드시 주소를 지정(주소 추출 연산자 & 사용)
 (단, 배열명의 경우는 배열명 자신이 그 배열의 시작주소를 의미하므로 &를 쓰지 않는다)

[예제 4-6] 10진수 3개를 입력 받아 그 합 출력하기

```

1 :  #pragma warning(disable : 4996)
2 :  #include <stdio.h>
3 :  int main()
4 :  {
5 :      int a=0, b=0, c=0, result, total;
6 :      printf("정수 세개를 입력하십시오 : ");
7 :      result = scanf("%d %d %d", &a, &b, &c);
8 :      if(result == 3)
9 :      {
10 :          total = a + b + c;
11 :          printf("세값의 합은 %d입니다.\n", total);
12 :      }
13 :      else
14 :      {
15 :          printf("입력데이터 오류입니다.\n");
16 :      }
17 :      printf("scanf() 함수의 리턴값 : %d\n", result);
18 :      return 0;
19 :  }
```

** 위의 예제를 다양한 방법으로 수행해보자.

- ①

10	20	30✓
----	----	-----

 ②

10	a	20✓
----	---	-----

 ③

a	10	20✓
---	----	-----

[예제 4-7] 문자열의 입력

```

1 : #include <stdio.h>
2 : int main()
3 : {
4 :     char oneStr[20], twoStr[20];
5 :     printf("input string : ");
6 :     scanf("%s", oneStr);
7 :     printf("input string : ");
8 :     scanf("%s", twoStr);
9 :     printf("oneStr : %s\n", oneStr);
10 :    printf("twoStr : %s\n", twoStr);
11 :    return 0;
12 : }
```

* fgets(char 배열명, 최대입력문자수, stdin) : 여백 있는 문자열 입력 시 사용

[예제 4-8] 숫자, 문자, 문자열의 입력

```

1 : #include<stdio.h>
2 : int main()
3 : {
4 :     int num, res;
5 :     char ch, str[20];
6 :     printf("input number, character, string : " );
7 :     res = scanf("%d %c %s", &num, &ch, str);
8 :     printf("number : %d\t character : %c\t string : %s\n", num, ch, str);
9 :     printf("res = %d\n", res);
10 :    return 0;
11 : }
```

▶ scanf()의 변경자

변경자	의 미
*	값을 입력은 받으나 그 값을 변수에 저장하지 않는다 예) %*d
10진수	최대필드 폭, 문자열 입력 시 첫 번째 여백 문자를 만나거나 최대 필드 폭에 도달하면 입력을 중지한다 예) %5s, %3d
h, l, ll 또는 L	short, long, long long, long double형으로 입력 받을 때 사용한다 예) %hd %ld %lld %lf %Lf

※ 정밀도(.10진수) 변경자는 scanf()함수에서는 절대 사용 불가

[예제 4-9] 사용자 지정 구분자의 사용

```

1 :    #pragma warning(disable:4996) //Visual C++ 2005 이상에서 사용
2 :    #include <stdio.h>
3 :    int  main()
4 :    {
5 :        int a, b, c;
6 :        int res;
7 :        res = scanf("%d:%d:%d", &a, &b, &c);
8 :        printf("res=%d, a=%d, b=%d, c=%d\n", res, a, b, c);
9 :        return 0;
10:    }
```

4. 단일문자 입력 함수(버퍼화 입력의 의미)

▶ 문자전용 입력 함수의 사용

- 전달인자 : 없음
- return 값 : 성공적으로 입력 받은 문자의 ASCII code 값

함수명	입력버퍼 사용여부	입력된 문자 사용시점	Enter 키 인식	입력시 수정 가능 여부	입력문자 화면 echo
getchar()	○	Enter키 입력 후	'\n'	Enter키 입력 전 수정 가능	○
getche()	X	입력 즉시	'\r'	수정 불가능	○
getch()	X	입력 즉시	'\r'	수정 불가능	X

** 다음 코드의 수행 결과는?

```
#include<stdio.h> // getchar(), putchar()함수 사용을 위해 include함
#include<conio.h> // getche(), getch()함수 사용을 위해 include함
int main()
{
    char ch;
    while( (ch=getchar()) != '\n')
    {
        putchar(ch);
    }
    getch();
    return 0;
}
```

* 위의 코드를 getche(), getch() 함수로도 테스트 해보자

[예제 4-10] 문자를 입력 받아 소문자와 대문자를 서로 변환하는 프로그램

```
1 :    #include<stdio.h>
2 :    int main()
3 :    {
4 :        char ch ;
5 :        while( ( ch = getchar() ) != '#' )
6 :        {
7 :            if(ch >= 'a' && ch <='z' ) { ch -= 'a'-'A'; }
8 :            else if(ch >= 'A' && ch <='Z' ) { ch += 'a'-'A' ;}
9 :            putchar (ch) ;
10:        }
11:        return 0;
12:    }
```

[예제 4-11] 파일의 끝(Ctrl+Z, unix에서 Ctrl+D)까지 대소문자 변환

```
1 :    #include<stdio.h>
2 :    int main()
3 :    {
4 :        char ch ;
5 :        while( ( ch = getchar() ) != EOF )
6 :        {
7 :            if(ch >='a' && ch <='z' ) { ch -= 'a'-'A';}
8 :            else if(ch >='A' && ch <='Z' ) { ch += 'a'-'A' ;}
9 :            putchar (ch) ;
10:        }
11:        return 0;
12:    }
```

제5장 제어문의 이해

학습 내용 소개

5장에서는 이전 2장에서 배웠던 기본 제어문을 바탕으로 좀 더 발전된 여러가지 제어문을 배움으로써 좀 더 복잡하고 다양한 알고리즘을 만드는 연습을 한다.

학습 목차

1. 또 하나의 택일문 switch ~ case문의 이해
2. 또 다른 반복문 for와 do~while의 이해
3. 분기문의 이해

1. 또 하나의 택일문 switch ~ case 문의 이해

switch ~ case문은 조건수식의 연산결과에 따라 각기 다른 별개의 문장을 수행한다.

[일반형식]

```
switch (조건수식)
{
    case 상수1 : 문장1 ; break ;
    case 상수2 : 문장2 ; break ;
    :
    case 상수n : 문장n ; break ;
    default   : 문장 ;
}
```

- 조건수식 : 반드시 정수식(정수변수, 문자변수 포함)이 와야 한다
- 상수1 ~ 상수n : 반드시 정수형 상수 또는 문자형 상수가 와야 한다
- break : switch ~ case 의 블록{} 을 빠져 나온다
- default : 주어진 경우(case) 이외 모든 경우의 처리를 맡는다
- * break 와 default 문은 생략 가능하다

【예】 수도요금 계산하기

사용자 code(1:가정용, 2:상업용, 3:공업용)와 사용량(ton단위)을 입력 받아 수도요금을 계산하여 출력하기

(논리흐름)

1. 시작
2. code저장 변수, 사용량 저장변수, 수도요금저장 변수 선언
3. 사용자 code, 사용량 입력 받기
4. 사용자 code에 따라 선택적으로 수행하시오.
 - 4-1. 가정용(code : 1)이면 : 사용량 * 1000원으로 수도사용요금 계산
 - 4-2. 상업용(code : 2)이면 : 사용량 * 700원으로 수도사용요금 계산
 - 4-3. 공업용(code : 3)이면 : 사용량 * 500원으로 수도사용요금 계산
5. 수도요금 출력하기
6. 종료

(source code)

```
int code, amount, fee;
scanf("%d", &code);
scanf("%d", &amount);
switch(code){
    case 1 : fee = amount*1000; break;
    case 2 : fee = amount*700; break;
    case 3 : fee = amount*500; break;
}
printf("수도요금은 %d입니다.\n", fee);
```

[예제 5-1] switch~case문의 예

```
1 :    #include<stdio.h>
2 :    int main()
3 :    {
4 :        int  score ;
5 :        char  remark ;
6 :        printf("점수를 입력하세요 : ") ;
7 :        scanf("%d", & score) ;
8 :        switch(score / 10)
9 :        {
10:            case 10 :
11:                case 9 : remark = 'A' ; break ;
12:                case 8 : remark = 'B' ; break ;
13:                case 7 : remark = 'C' ; break ;
14:                case 6 : remark = 'D' ; break ;
15:                default : remark = 'F' ;
16:        }
17:        printf("점수는 %d 학점은 %c 입니다 \n", score, remark) ;
18:        return 0;
19:    }
```

2. 또 다른 반복문 for와 do~while의 이해

2.1 for 문

[일반형식]

```
for ( 초기식 ; 조건식 ; 증감식 )
{
    반복할 문장 ;
    :
}
```

[각 식의 일반적 역할]

- 초기식 : Loop 제어변수를 초기화 한다 (처음 한번만 수행)
- 조건식 : Loop 제어변수의 범위를 검사하여 반복여부를 결정한다
(조건식이 참인동안 반복)
- 증감식 : Loop 제어변수를 증가 또는 감소 시킨다

초기식, 증감식에 순차연산자(,) 를 이용하여 Loop 제어변수 이외의 변수를 초기화 하거나 증감 할 수 있다

```
for ( i = 0, j = 0 ; 조건식 ; i++, j++ )
{
    반복할 문장 ;
}
```

조건식에 논리연산자(&& , ||)를 이용하여 중복 조건을 처리 할 수 있다

```
for ( 초기식 ; i >= 1 && i <= 10 ; 증감식 )
{
    반복할 문장 ;
}
```

초기식, 조건식, 증감식을 모두 널(NULL)문으로 처리 하면 for문의 무한 loop이 됨.

【 예 】단어를 입력 받아 길이를 출력하는 작업을 5회 반복하세요.

(논리흐름)

1. 시작
2. 단어를 저장할 기억공간(문자배열) 선언
3. 단어의 길이를 저장할 변수(int형) 선언
4. 반복제어변수 i=0,1,2,3,4일 동안 반복
 - 4-1. 단어 입력 받기
 - 4-2. 단어 길이를 구하여 변수에 저장
 - 4-3. 단어길이 출력
5. 종료

(source code)

```
char word[80];
int length, i;
for(i=0; i<5; i++)
{
    scanf("%s", word);
    length = strlen(word);
    printf("%s의 길이는 %d Byte 입니다.\n", word, length);
}
```

[예제 5-2] 단일 for문의 활용

```
1 :   #include<stdio.h>
2 :   int main()
3 :   {
4 :       int i, s = 0 ;
5 :       for( i = 1 ; i <= 10 ; i++ )
6 :       {
7 :           printf("i = %d\n", i) ;
8 :           s += i ;
9 :       }
10:      printf("1 → %d = %d \n", i, s) ;
11:      return 0;
12:  }
```

[예제 5-3] 단일for문을 이용한 홀수 합 구하기

```
1 :   #include<stdio.h>
2 :
3 :   int main()
4 :   {
5 :       int i, s ;
6 :       for( i = 1, s = 0 ; i <= 100 ; i += 2 )
7 :       {
8 :           s += i ;
9 :       }
10:      printf("1 + 3 + 5 + .... + 99 = %d \n", s) ;
11:      return 0;
12:  }
```

▶ 다중 for문

```

for(i=0; i<10; i++)
{
    statement1;
    for(j=5; j>0; j--)
    {
        statement2;
    }

    statement3;
}

```

- 다중 반복문을 구현할 때에는 서로 다른 loop제어 변수를 사용해야 함
- 반복문 중첩 시에 loop제어 변수명은 i,j,k,l,m,n... 순서로 사용하는 것이 묵시적인 약속임

2.2 do ~ while 문

반복할 문장을 우선 한번 실행 후 조건이 참일 동안 반복하는 후 조건 비교 반복문

[일반형식]

```

do
{
    반복할 문장 ;
    :
} while ( 조건식 );

```

[예제 5-4] do~while문의 예

```

1 :   #include<stdio.h>
2 :   int main()
3 :   {
4 :       char  ch  =  'A' ;
5 :
6 :       do
7 :       {
8 :           putchar ( ch++ ) ;
9 :       } while ( ch <= 'Z' ) ;
10:      return 0;
11:  }

```

3. 분기문의 이해

▶ 분기문의 종류

- **break** : for문, while문, do~while문과 같은 반복문이나
switch ~ case문을 탈출하는데 사용
- **continue** : for문, while문, do~while문의 조건부로 제어를 옮기는데 사용
- **goto** : 같은 함수 내의 특정 레이블로 제어를 옮길 때 사용
- **return** : 자신을 호출했던 함수로 제어를 되돌릴 때 사용

[예제 5-5] break 와 continue 예제

```
1 :    #include<stdio.h>
2 :    int  main()
3 :    {
4 :        int  num ;
5 :        for( num = 1 ; num < 10 ; num++ )
6 :        {
7 :            if ( num == 5 ) break ;
8 :            printf("%3d", num ) ;
9 :        }
10:
11:        putchar( '\n' ) ;
12:
13:        for( num = 1 ; num < 10 ; num++ )
14:        {
15:            if ( num == 5 ) continue ;
16:            printf("%3d", num ) ;
17:        }
18:        return 0;
19:    }
```

▶ goto문

제어를 특정 레이블이 있는 곳으로 옮김

다중 Loop를 탈출하는 경우 외에는 거의 사용하지 않는다.

goto문과 레이블명은 한 함수 block내에 존재해야만 한다.

[일반형식]

```
int num ;
scanf("%d",&num) ;
if (num < 0) goto LB1 ;
else goto LB2 ;

LB1 :
    printf(. . . ) ;
    :
LB2 :
    printf( . . . ) ;
    :
```


제6장 연 산 자

학습 내용 소개

6장에서는 C언어에서 제공되는 40여 개의 다양한 연산자의 기능 및 역할을 살펴보고 연산자 사용 시 꼭 알고 넘어가야 할 여러 가지 상황을 짚어본다.

학습 목차

1. 연산자의 기본개념
2. 연산자 우선순위표
3. 연산자 기능의 이해

1. 연산자의 기본개념

- 연산자(operator) : 프로그램상의 상수, 변수 등에 대하여 여러 가지 조작을 하기 위한 기호이다

- 피연산자(operand) : 연산자에 대해서 연산이 적용되는 대상

예) $C = A + B$

+ 연산자로 피연산자 A, B의 두 값을 더한 후

더해진 값을 = 연산자를 이용 피연산자 C로 치환

2. 연산자 우선순위표

순위	명칭	연산자	결합방향
1	1차 연산자	() [] . ->	->
2	단항 연산자	+ - ! ~ (type) sizeof ++ -- & *	<-
3	이 항 연 산 자	승법 연산자 * / %	->
4		가법연산자 + -	
5		Shift 연산자 << >>	
6		관계 연산자 < > <= >=	
7		등가 연산자 == !=	
8		bit 곱 연산자 &	
9		bit 차 연산자 ^	
10		bit 합 연산자 	
11		논리곱 연산자 &&	
12		논리합 연산자 	
13	조건 연산자	?:	<-
14	대입 연산자	= += -= *= /= %= <<= >>= &= = ^=	<-
15	순차 연산자	,	->

3. 연산자 기능의 이해

3.1 1순위 연산자의 이해

1. `()` : 수식의 우선순위 변경
2. `[]` : 배열선언, 배열 요소 지정
3. `.` : 구조체 및 공용체의 멤버 연산자(membership operator)
4. `→` : 구조체 및 공용체의 간접멤버 참조 연산자

3.2 단항연산자의 이해

1. `-` : unary minus (부호반전)
`+` : unary plus (부호유지)
2. `!` : 논리부정 (0 - False, 0 이외의 것 - True)
 예) `a = 10 > 5 ;` (True 이므로 a안에는 1이 대입됨)
`b = !(10 > 5);` (True 를 논리부정 하였으므로 False 가 됨.
 b에는 0이 대입됨)
3. `++ /--` : 1 증가 / 1 감소 연산자 (Increment / Decrement)
 변수의 값을 1씩 증가하거나 감소 시키는 연산자

[다른 연산자와 함께 쓰일 때 전위형과 후위형으로 처리됨]

- 전위형(prefix) : 먼저 현재 변수의 값을 증감시킨 후 증감된 값으로 다른 연산 수행
- 후위형(postfix) : 먼저 현재 변수의 값으로 다른 연산을 수행한 후 변수값을 증감

[예제 6-1] 증가/감소 연산자 예제

```

1 :   #include<stdio.h>
2 :   int main()
3 :   {
4 :       int a = 0, b = 0;
5 :       a++; b++;
6 :       printf ( " a = %d b = %d \n " , a , b );
7 :       b = a++;
8 :       printf ( " a = %d b = %d \n " , a , b );
9 :       b = ++a;
10:      printf ( " a = %d b = %d \n " , a , b );
11:      printf ( " a = %d b = %d \n " , a ++, ++b );
12:      printf ( " a = %d b = %d \n " , a , b );
13:      return 0;
14:  }

```

4. ~ : One's complement (1의 보수 연산자)

비트값을 0은 1로, 1은 0으로 변환하여주는 연산자

[예제 6-2] 1의 보수 연산자 예제

```

1 :   #include<stdio.h>
2 :   int main()
3 :   {
4 :       short num = 0x0f0f , res ;
5 :       res = ~ num ;
6 :       printf ( " num = %hx      res = %hx \n " , num , res ) ;
7 :       return 0;
8 :   }

```

* 위의 예제를 %x로 출력하면 어떤 현상이 일어날까요?

5. (type) : Cast 연산자 (형변환 연산자)

```

int a = 10;
double res;
res = (double)a/3;
printf("a=%d, res=%lf\n", a, res);

```

[자동형변환의 기본방식]

- 작은형이 큰형으로 변환된다.
- 대입문에서는 오른쪽의 자료가 왼쪽의 변수의 형으로 변환되어 대입된다.
- 0차원의 기본 데이터형에서만 발생한다.

6. & : 주소 연산자 (주소추출 연산자)

예)) `int x = 5;`
`printf (" x = %d , &x = %p\\n", x, &x);` => x값 5 와 x의 주소가 출력됨

7. * : 포인터 연산자

- 선언문 : pointer변수(주소상수 저장 변수) 선언 시 사용
- 실행문 : pointer변수가 가리키는 대상체를 의미

[예제 6-3] & 와 * 연산자 예제

```

1 :    #include<stdio.h>
2 :    int  main()
3 :    {
4 :        int  x = 5, *ptr ;
5 :        ptr = &x ;
6 :        printf ( "%p %p %p %d %d \\n" , &ptr , ptr , &x , x ,*ptr);
7 :        return 0;
8 :    }

```

8. sizeof : 변수, 상수 또는 데이터형의 크기를 byte 수로 계산해 준다

[예제 6-4] sizeof 연산자 예제

```

1 :    #include<stdio.h>
2 :    int  main()
3 :    {
4 :        int a=5, b[10];
5 :        double c[3];
6 :        printf(" 'A'      size : %d \\n" , sizeof('A') ) ;
7 :        printf(" \"A\"      size : %d \\n" , sizeof("A") ) ;
8 :        printf(" \"AB\"      size : %d \\n" , sizeof("AB") ) ;
9 :        printf(" 5        size : %d \\n" , sizeof(5) ) ;
10:        printf(" 5.0      size : %d \\n" , sizeof(5.0) ) ;
11:        printf(" 5.0f     size : %d \\n" , sizeof(5.0f) ) ;
12:        printf(" char     size : %d \\n" , sizeof(char) ) ;
13:        printf(" int       size : %d \\n" , sizeof(int) ) ;
14:        printf(" float     size : %d \\n" , sizeof(float) ) ;
15:        printf(" double    size : %d \\n" , sizeof(double) ) ;
16:        printf(" a         size : %d \\n" , sizeof(++a) ) ;
17:        printf(" a = %d\\n", a);
18:        printf(" b         size : %d \\n" , sizeof(b) ) ;
19:        printf(" c         size : %d \\n" , sizeof(c) ) ;
20:        return 0;
21:    }

```

3.3 이항연산자의 이해

1. `<<`, `>>` : **shift 연산자** : bit 단위 좌/우 이동 연산자

[shift시 공백메우는 법칙]

Shift 방향 및 대상체 부호		공백 메우기
<< (left shift)	signed	0으로 채워짐
	unsigned	
>> (right shift)	signed	sign bit의 값으로 채워짐
	unsigned	0으로 채워짐

[예제 6-5] shift 연산자 예제

```

1 :  #include<stdio.h>
2 :  int main()
3 :  {
4 :      int i = 0, k = 1;
5 :      while(i++ < 8)
6 :      {
7 :          printf( " k << %d = %d \n " , i , k = k << 1);
8 :      }
9 :      return 0;
10: }
```

** shift 연산자를 이용해서 특정정수(n)의 값의 10배 값을 구하고자 하면
n의 8배수+n의 2배수의 형태로 결과를 얻으면 된다.

즉, $(n \ll 3) + (n \ll 1)$ 로 계산함

** 위의 예제에서 $k \ll 33$ 을 수행하면?

2. `%` : **나머지 연산자 (modulus 연산자)** : 몫이 아니라 나머지를 구함

예) $13 \% 4 \Rightarrow 1$ (나머지) , $13 / 4 \Rightarrow 3$ (몫)

[예제 6-6] 나머지 연산자 예제

```

1 :  #include<stdio.h>
2 :  int main()
3 :  {
4 :      short res;
5 :      res = 9 - 2 % 7 + 6 ;
6 :      printf( " 9 - 2 %% 7 + 6 = %hd \n", res ) ;
7 :      return 0;
8 : }
```

3. <, >, <=, >= : 관계연산자

4. == (Equal) , != (Not equal) : 등가 연산자

** 관계연산자와 등가연산자의 결과는 true 또는 false 로 결과가 나온다

5. & : bitwise AND

^ : bitwise Exclusive OR

| : bitwise OR

[진리표 : 대응되는 x,y 각 비트에 대한 비트 연산 결과]

X	y	x & y	x ^ y	x y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

6. && : Logical AND, || : Logical OR

[진리표 : T : true, F : false]

x	y	x && y	x y
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

[수학적 표기와의 차이점]

$a \leq b \leq c \rightarrow a \leq b \ \&\& \ b \leq c$

$a \geq b, b \leq c \rightarrow a \geq b \ || \ b \leq c$

[short curcuit 기능 : 처리 속도의 효율성을 고려한 기능]

Logical AND : 왼편이 거짓이면 오른쪽은 계산되지 않는다(결과는 false)

Logical OR : 왼편이 참이면 오른쪽은 계산되지 않는다(결과는 true)

[예제 6-7] 논리 연산자 예제

```

1 :   #include<stdio.h>
2 :   int main()
3 :   {
4 :       char ch;
5 :       int i = 0;
6 :       printf("입력한 문자에 대하여 대/소 판별 해 드립니다\n");
7 :       while ( i++ < 3 ) {
8 :           printf("\n문자를 하나 입력하고 엔터 치세요 : ");
9 :           scanf("%c", &ch);
10:            if(ch >= 'A' && ch <= 'Z') {
11:                printf("입력한 문자는 대문자 입니다\n");
12:            }
13:            else if (ch >= 'a' && ch <= 'z') {
14:                printf("입력한 문자는 소문자 입니다\n");
15:            }
16:        } /* end of while */
17:    }

```

3.4 삼항연산자와 대입연산자, 단축형연산자

1. ?: : 조건 연산자 (3 항 연산자)

[형식 1]

(조건식) ? (T-식) : (F-식);

조건식이 참이면 T-식을 거짓이면 F-식을 수행한다

예) (p == 1) ? (p++) : (p--);

[형식 2]

변수 = (조건식) ? (T-값) : (F-값);

조건식이 참이면 T-값을 거짓이면 F-값을 좌측의 변수로 넘겨준다

예) int max = (a > b) ? (a) : (b);

[예제 6-8] 조건 연산자 예제

```

1 :   #include<stdio.h>
2 :   int  main()
3 :   {
4 :       int  a,b;
5 :       printf("두 수를 입력하세요 :");
6 :       scanf("%d %d", &a, &b);
7 :       (a>b) ? printf("%d > %d \n", a, b) : (a < b) ? printf("%d < %d \n", a, b)
8 :           : printf("%d == %d \n", a, b) ;
9 :       return 0;
10:  }
```

2. =, +=, *=, /= ... : 대입연산자 및 단축형 연산자

좌측의 피연산자를 우측의 피연산자 값(단위) 만큼 연산을 행하여 갱신시킨다

예) a += b; → a = a + b;

a를 b만큼 더하여 갱신시킨다

a <<= 2 → a = a << 2;

a를 2비트 좌측(<<)으로 이동(shift)하여 갱신시킨다

제7장 함수(Function)의 기본 활용

학습 내용 소개

7장에서는 C프로그램의 기본 단위인 함수에 대해 공부합니다.

사용자 정의 함수 만들기부터 함수의 활용 기법을 체계적으로 배워 실제 프로그래밍에 활용해보자.

학습 목차

1. 함수의 이해
2. 함수 매개변수
3. 함수 호출 기법
4. 가변인수 함수 만들기
5. 재귀호출 함수

1. 함수의 이해

▶ 함수란?

특정한 작업을 수행하도록 독립적으로 작성된 명령 코드 블록({})을 말한다.

▶ 함수의 사용 목적

- 프로그램 내에서 반복적으로 수행되는 부분을 함수로 작성하여 필요 시 마다 호출하여 사용한다.
- 특정 기능별로 함수로 나누어 작성하여 구조적인 프로그래밍이 가능하게 한다.
- 프로그램 코드의 분할과 재사용이라는 의미에서 매우 유용한 개념이다.

▶ 함수의 종류

- library function : Maker에서 제공하는 함수
- user define function : 사용자가 직접 작성하여 사용하는 함수

▶ 함수의 기본형식

● 함수선언부

Type 함수명 ([parameter list]); ← 함수선언부에서 함수의 type은 생략불가

● 함수정의부

[Type] 함수명 ([parameter list])

{

 [내부 변수 선언 ;]

 [문장 ;]

 :

 [return return값;]

}

[] : 생략 가능 표시

- 함수명은 식별자이므로 식별자 만드는 규칙에 의거하여 지어준다.
- 함수명은 전체 프로그램 내에서 유일해야 한다. (단, static함수는 제외)
- Type은 함수의 형을 의미하며, 그 함수의 return 값의 type 을 의미한다
(생략하면 compiler는 무조건 int형으로 인식함)
- parameter list는 그 함수가 호출될 때 외부로부터 전달받는 데이터들의 받아주는 변수들을

기술하는 부분이다. (전달받을 데이터가 없을 때 생략하거나 "void"로 지정 함)

- 함수는 중괄호({})로 함수 body(block의 시작과 끝)를 표시하며, block 내에 함수의 기능을 기술한다

※ 함수 시그니처(signature) : 함수명+매개변수(parameter) list(매개변수의 type, 개수, 순서)

1.1 함수의 선언, 호출, 정의

▶ 함수의 선언 : 함수원형(function prototype : 리턴값 정보, 함수명, 전달인자 정보)을 compiler에게 알려주는 역할을 한다.

▶ 함수의 호출 : 작성된 함수를 사용하기 위해 함수 이름을 불러준다. 반복 호출이 가능하다.

▶ 함수의 정의 : 함수가 수행 해야 할 실제 기능을 기술한 부분

* library function의 선언부는 헤더파일에 들어 있고, 정의는 compile 되어 라이브러리 파일에 들어있다.

* user define function의 선언부와 정의부는 모두 source code 내에 들어있다.

(sub()함수를 호출하는 형태는 적절한가?)

void sub(int); // 함수선언부	sub(); ←
int main()	sub(da); ←
{	}
int a=3, result;	// 함수정의부
double da = 7.8;	void sub(int n)
sub(a); ←	{
result = sub(a); ←	printf("n의 2배수 : %d", n*2);
sub(3); ←	}

[예제 7-1] 함수의 선언, 호출, 정의의 예

```

1 :    #include<stdio.h>
2 :    double sumPrn ( int , double ) ;    /* 함수선언 */
3 :    int main()
4 :    {
5 :        int a = 10 ;
6 :        double b = 5.4 ;
7 :        double res ;
8 :        res = sumPrn ( a, b ) ;    /* 함수호출 */
9 :        printf(" res = %lf\n ", res ) ;
10:        return 0;
11:    }
12:    double sumPrn ( int x, double y) /* 함수정의 */
13:    {
14:        double num ;
15:        printf(" int : %d\t double : %lf\n" , x, y ) ;
16:        num = x + y ;
17:        return num ;
18:    }

```

▶ return 문의 기능

- 자신을 호출했던 함수(호출함수)로 제어를 되돌린다
- 피호출함수의 수행결과를 호출함수로 전달한다 (return 리턴값;)
(return 값은 오로지 1개의 값을 전달 가능)
- return값 없이 제어만 호출 함수로 되돌릴 수도 있다(return;)
- 함수의 끝부분에 return문이 없는 경우 함수의 닫는 괄호가 return문의 기능을 대신함

▶ 함수에서의 void의 의미

- 함수형이 "void"인 것은 그 함수의 반환값(return 값)이 없음을 의미한다
- 전달인자가 "void"인 것은 전달인자가 없음을 의미한다

[예제 7-2] 전달인자와 리턴값이 있는 함수 예제

```
1 :    #include<stdio.h>
2 :    int  abs ( int ) ;
3 :    int main()
4 :    {
5 :        int num ;
6 :        printf("하나의 수를 입력하세요 : ") ;
7 :        scanf("%d" , &num) ;
8 :        printf("입력된 수의 절대값은 : %d 입니다 \n" , abs(num) ) ;
9 :        return 0;
10:    }
11:    int abs (int x)
12:    {
13:        if ( x > 0 ) { return (x) ; }
14:        else { return (-x) ; }
15:    }
```

[예제 7-3] 전달인자와 리턴값이 없는 함수예제

```
1 :    #include<stdio.h>
2 :    void hello ( void ) ;
3 :    int main()
4 :    {
5 :        hello () ;
6 :        return 0;
7 :    }
8 :    void hello ( void )
9 :    {
10:        printf(" Hello!  I am glad to meet you \n");
11:    }
```

2. 함수 전달인자와 매개변수

▶ 인수(argument)와 매개변수(parameter)의 차이

- 인수 또는 전달인자(argument) : 호출 함수 쪽에서 사용되는 변수
- 매개변수(parameter) : 피 호출함수의 정의부에서 사용되는 변수

▶ 전달인자와 매개변수의 특징

- argument의 개수와 parameter의 개수는 일치해야 한다
- argument와 parameter type은 같게 사용하는 것이 기본이나 프로그래머의 특별한 의도에 의해서 다르게 사용할 경우 parameter의 type 대로 argument가 자동형변환되어 전달 됨
(자동형변환은 0차원의 기본자료형에서만 일어난다)
- argument명과 parameter명은 같아도 되고 달라도 된다.
(같은 이름을 쓰더라도 서로 다른 기억 장소에 할당 됨)

※ 변수선언 위치에 따른 변수들의 사용범위(scope)

<pre>int <u>a</u>; void sub1(int); void sub2(int); int main() { int <u>a</u>=10; : sub1(a); sub2(a); return 0; }</pre>	<pre>void sub1(int <u>a</u>) { : } void sub2(int x) { : }</pre>
--	--

※ 다음 code 는 옳을까?

<pre>void myFunc(int); int main() { int <u>a</u>=10; : myFunc (a); return 0; }</pre>	<pre>void myFunc(int <u>num</u>) { int num = 5; printf("num = %d\n", num); }</pre>
--	--

3. 함수 호출 기법

▶ Call by value 기법 (값 전달 방식)

- 호출함수에서 피호출 함수로 argument의 상수값을 복사하여 전달하는 방식.
- 피호출 함수의 조작으로부터 argument의 값은 항상 보호됨 (원본 데이터 변경 불가능)
- parameter는 argument와 동일한 type의 변수로 받아 주는 것이 일반적임

▶ Call by pointer 기법 (주소 전달 방식)

- 호출함수에서 피호출 함수로 argument의 시작주소를 전달하는 방식
- 피호출 함수의 조작으로 argument의 값을 변경할 수도 있음 (원본 데이터 변경 가능)

[예제 7-4] call by value 기법의 예

```

1 :    #include<stdio.h>
2 :    void exchange (int , int) ;
3 :    int main()
4 :    {
5 :        int x = 5, y = 10 ;
6 :        exchange(x, y) ;
7 :        printf("x = %d \t y = %d\n", x, y) ;
8 :        return 0;
9 :    }
10:    void exchange (int a, int b)
11:    {
12:        int temp;
13:        temp = a ; a = b ; b = temp ;
14:        printf("a = %d \t b = %d\n", a, b) ;
15:    }

```

[예제 7-5] call by pointer 기법의 예

```

1 :    #include<stdio.h>
2 :    void exchange (int * , int *) ;
3 :    int main()
4 :    {
5 :        int x = 5, y = 10 ;
6 :        exchange(&x, &y) ;
7 :        printf("x = %d \t y = %d\n", x, y) ;
8 :        return 0;
9 :    }
10:    void exchange (int *a, int *b)
11:    {
12:        int temp;
13:        temp = *a ; *a = *b ; *b = temp ;
14:        printf("**a = %d \t *b = %d\n", *a, *b) ;
15:    }

```


※ 생각해 보기!!

[예제 7-6] 다음 코드에서 sub()함수는 call by pointer일까 call by value 일까?

```
#include<stdio.h>
void sub (int *);
int main()
{
    int a=7;
    int *p;
    p=&a;

    printf("p = %p  a = %d\n", p, a);
    sub(p);
    printf("p = %p  a = %d\n", p, a);

    return 0;
}
void sub (int *ap)
{
    *ap = 10;
    return;
}
```

4. 재귀호출 함수

▶ 재귀적 호출(Recursive call) 함수란 ?

- 작업을 작은 단위로 쪼개서 재귀 호출된 함수 내에서 이 조각중에 하나를 해결하고, 나머지 조각들은 자기 자신을 또 호출해서 해결하는 방식으로 처리된다.
- 함수 내부에서 직접 혹은 간접적으로 자기 자신을 호출하는 함수
- 재귀적 정의를 이용해서 재귀 함수를 구현한다.
 - * 재귀적 정의 : 기본 부분(Basis part)와 유도 부분(Inductive part)로 구성된 알고리즘
 - . 기본 부분(Basis part) : 재귀 알고리즘을 끝내기 위한 부분
 - . 유도 부분(Inductive part) : 새로운 집합의 원소를 생성하기 위해 재귀하는 부분
- 재귀적 프로그래밍은 반복 구조에 비해 간결한다.
(단, 재귀에 익숙치 않은 경우 어렵다고 느껴짐)
- 재귀 함수 호출은 반복적으로 스택을 사용하므로 메모리 및 속도에서 성능저하가 발생한다.
- 특정 시점에서 재귀적호출을 종료하지 못하면 무한 재귀호출에 빠질 수 있다
(기본 부분(Basis part)가 제대로 구현되지 않은 경우)

[예제 7-7] 무한 재귀 호출에 빠지는 경우의 예

```

1 :   #include<stdio.h>
2 :   int  i = 0 ;
3 :   void  sub ();
4 :
5 :   int  main()
6 :   {
7 :       sub();
8 :       return 0;
9 :   }
10:  void  sub ()
11:  {
12:      printf("i = %d \n", ++i ) ;
13:      sub () ;    // 유도 부분(inductive part)
14:  }

```

[예제 7-8] 한 방향 재귀호출의 예

```

1 :   #include<stdio.h>
2 :   int  fact (int) ;
3 :   int  main()
4 :   {
5 :       int  num = 5 ;
6 :       printf("5! = %d \n" , fact(num) ) ;
7 :       return 0;
8 :   }
9 :
10:  int  fact (int  x)
11:  {
12:      printf("x = %2d\n", x);
13:      if(x==1) return 1;          // 기본 부분(Basis part)
14:      else return x*fact(x-1);    // 유도 부분(Inductive part)
15:  }

```

예) 위의 factorial 예제에서 return 문의 조건식에서 아래와 같을 경우는 문제가 발생할 수 있다.

- 기본 파트(Basis part) 오류

```

int  fact (int  x)
{
    return ( x==2 ) ? 2 : x*fact(x-1);
}

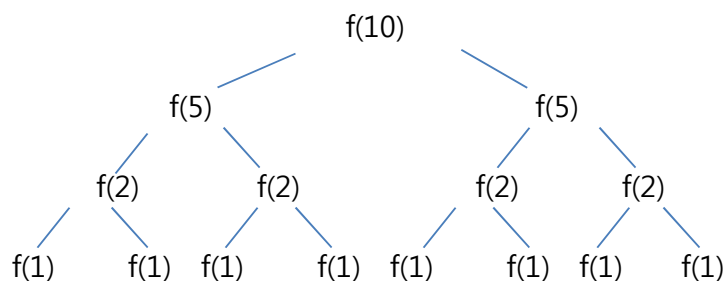
```

[예제 7-9] 여러방향 재귀호출의 예

```

1 :   #include<stdio.h>
2 :
3 :   int cnt;
4 :   void f(int n);
5 :
6 :   int main()
7 :   {
8 :       int num = 10 ;
9 :       f(num);
10:      printf("%d\n" , cnt) ;
11:      return 0;
12:  }
13:
14:  void f(int n) //정수 1개를 전달받아 n에 저장
15:  {
16:      if(n<=1) cnt++; //n<=1 이면 c에 저장된 값을 1만큼 증가
17:      else
18:      {
19:          f(n/2); //n을 2로 나눈 몫으로 재귀호출
20:          f(n/2); //한 번 더 f(n/2) 실행
21:      }
22:      return;
23:  }

```



▶ 재귀와 반복의 비교

	재귀	반복
종료	재귀함수 호출이 종료되는 기본 부분(Base part)필요	반복문이 종료 조건
수행 시간	반복에 비해 느림	빠름
메모리 공간	반복에 비해 많이 사용	적게 사용
소스 코드 길이	짧고 간결	길다
소스 코드 형태	선택 구조 (if~else)	반복 구조 (for, while, do while)
무한 반복 시	Stack overflow 발생	CPU 반복적 점유

제8장 배 열

학습 내용 소개

8장에서는 복합 데이터형 중 가장 대표적으로 많이 사용되는 배열에 대해서 상세히 배운다.
배열의 기본개념과 배열을 이용한 다양한 프로그램을 작성해보고, 포인터와의 연관성을 쉽게 이해하기 위해서 차원개념을 파악하자~

학습 목차

1. 배열의 기본 개념
2. 1차원 배열의 선언 및 초기화
3. 배열에 file내의 데이터 저장하기
4. 2차원 배열의 선언 및 초기화
5. 배열과 증감 연산자
6. 배열선언 및 초기화 시의 주의점

1. 배열의 기본 개념

▶ 배열이란 ?

동일한 데이터형 변수의 집합

`int a[5];` // int형 변수 5개가 할당 됨

▶ 배열 선언 규칙

배열명 뒤의 대괄호([])안에 정수형 첨자(subscript, index)를 지정하면 첨자수 만큼 배열의 요소(원소, 방)가 할당됨

▶ 배열의 특징

1. 메모리 상의 연속적인 공간에 할당된다.
2. 배열명은 곧 그 배열의 시작주소상수 이다.
3. 배열 요소는 첨자로 구분하며 첨자는 0(zero)부터 시작된다. (선언문에서의 첨자는 배열 요소의 개수를 의미하나 일반 실행문에서의 첨자는 배열 요소의 번호를 의미함)

2. 1차원 배열의 선언 및 초기화

2.1 1차원 숫자배열의 선언

`int ary [10];` // int 방 10개로 구성된 1차원 배열 선언

▶ 차원 및 차원 조절

- C코드 내에서 사용되는 모든 상수나 변수는 차원이 부여되어있다
- 상수나 변수의 차원은 차원조절 연산자에 의해서 차원이 변경될 수 있다
- Lvalue와 Rvalue는 차원이 같아야 대입연산이 가능하다.
(주소상수는 자동형변환이 불가능 하다)

▶ 차원 조절 연산자

연산자	선언문	일반 실행문	비 고
<code>*</code> , <code>[]</code>	차원 올림	차원 내림	
<code>&</code>	사용불가	차원 올림	
<code>(type)</code>	사용불가	여러차원 올리고 내리기 모두 가능	
<code>-></code>	사용불가	차원 내림	

2.2 1차원 배열의 초기화

1. 배열 전체의 초기화는 선언문에서만 가능하다
2. 배열을 초기화 할 때는 초기화 시작과 끝 괄호로 중괄호({})를 이용하여 초기화할 데이터를 묶어서 표현한다.
(단, 1차원 char배열을 문자열로 초기화 할 때에는 중괄호를 생략할 수 있다)
3. 배열요소의 개수보다 초기화 데이터가 부족하면 남은 요소는 0(zero)으로 채워진다
4. 배열 요소의 개수보다 초기화 데이터가 많으면 compile error가 발생한다
5. 배열선언 시 첨자는 생략할 수 없으나 초기화 데이터가 있는 경우에 한하여 첨자를 생략할 수 있다
6. 문자열을 char 배열에 저장할 때에는 배열이 첨자 크기를 문자열의 길이+1(null문자 저장 공간)만큼으로 정한다.

[예제 8-1] 초기화된 배열의 합 출력

```
1 :    #include<stdio.h>
2 :    int main()
3 :    {
4 :        int ary[5] = { 1, 3, 5, 7, 9 } ;
5 :        int i, sum ;
6 :        for ( i = sum = 0 ; i < 5 ; i++ )
7 :        {
8 :            printf("ary[%d] = %d\n", i, ary[i];
9 :            sum += ary[i] ;
10:        }
11:        printf("sum = %d \n" , sum ) ;
12:        return 0;
13:    }
```

[예제 8-2] 배열을 이용한 합과 평균 출력

```
1 :      #pragma warning (disable : 4996) // Visual Studio 2005 이상 버전에서
2 :          // 표준입출력함수와 관련된 경고를 없애는 전처리 명령어
3 :      #include<stdio.h>
4 :      void array_input(int *, int);
5 :      int array_sum(int *, int);
6 :      int main()
7 :      {
8 :          int sum , ary[10] ;
9 :          double ave ;
10:
11:          array_input(ary, 10);
12:          sum = array_sum(ary, 10);
13:          ave = (double)sum / 10 ;
14:          printf("Sum = %d Ave = %.2lf \n", sum, ave) ;
15:          return 0;
16:      }
17:      void array_input(int *ip, int count)
18:      {
19:          int i;
20:          for(i=0 ; i<count ; i++)
21:          {
22:              printf("%d번방의값= " , i ) ;
23:              scanf("%d", &ip[i] ) ;
24:          }
25:          return;
26:      }
27:      int array_sum(int *ip, int count)
28:      {
29:          int i;
30:          int sum;
31:          for(i=0, sum=0; i<count; i++)
32:          {
33:              sum += ip[i];
34:          }
35:          return sum;
36:      }
```

[예제 8-3] 1차원 문자 배열 선언 및 초기화

```

1 :      # include <stdio.h>
2 :      int  main()
3 :      {
4 :          char  str[8];
5 :          char  str1[8] = {'P','r','o','g','r','a','m'};
6 :          char  str2[] = "  Good !";
7 :          str[0] = 'T' ;
8 :          str[1] = 'u' ;
9 :          str[2] = 'r' ;
10:         str[3] = 'b' ;
11:         str[4] = 'o' ;
12:         str[5] = '-' ;
13:         str[6] = 'C' ;
14:         str[7] = '\0' ;  // <== 이 줄은 꼭 필요한 것일까?
15:         printf("str : %s \n" , str ) ;
16:         printf("str1: %s \n" , str1 ) ;
17:         printf("str2 : %s \n" , str2 ) ;
18:         return 0;
19:     }

```

[예제 8-4] 문자 배열 출력

```

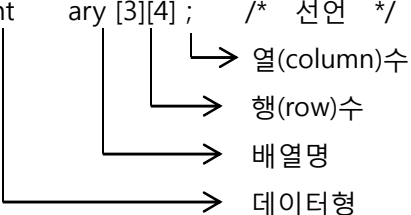
1 :      #include <stdio.h>
2 :      void string_print(char *, int);
3 :      int  main()
4 :      {
5 :          char  str[ ] = "Dream";
6 :          int  size = sizeof(str) / sizeof(str[0]);
7 :          string_print(str, size);
8 :          return 0;
9 :      }
10:     void string_print(char *cp, int size)
11:     {
12:         int i;
13:         for( i = 0 ; i < size ; i++ )
14:         {
15:             printf("str[%d]:%p:%c:%s\n", i, &cp[i], cp[i], &cp[i]);
16:         }
17:         return;
18:     }

```

3. 2차원 배열의 선언 및 초기화

▶ 2차원 배열의 선언

```
int ary[3][4]; /* 선언 */
```



	0열	1열	2열	3열
0행	ary[0][0]	ary[0][1]	ary[0][2]	ary[0][3]
1행	ary[1][0]	ary[1][1]	ary[1][2]	ary[1][3]
2행	ary[2][0]	ary[2][1]	ary[2][2]	ary[2][3]

▶ 2차원 배열의 초기화

1. 2차원 숫자 배열을 초기화 할 때는 초기화 시작과 끝괄호({}) 외에도

각 행의 초기화 시작과 끝 괄호를 넣어준다({})

이때 행초기화 시작과 끝괄호는 생략 가능하다

```
int ary[3][4] = {{1,1,1,1}, {2,2,2,2}, {3,3,3,3}};
```

```
int ary[3][4] = {1,1,1,1,2,2,2,2,3,3,3,3};
```

```
int ary[3][4] = {{1,1}, {2,2,2}, {3,3,3,3}};
```

```
int ary[3][4] = {1,1,0,0,2,2,2,0,3,3,3,3};
```

int ary[3][4] = {1,1,2,2,2,3,3,3,3}; 는 어떤 모양으로 초기화 될까요?

2. 2차원 문자 배열을 초기화 할 때에는 문자열의 개수 만큼 행 첨자를 지정하고, 열 첨자는 초기화 문자열들 중 가장 긴 문자열의 길이에 NULL문자를 위하여 +1 한 크기로 정한다.

이차원 문자배열 초기화 시에는 배열 초기화 시작, 끝 괄호를 생략할 수 없다.

[예제 8-5] 이차원 숫자 배열 예제

```
1 :    #include <stdio.h>
2 :    void array_input(int (*)[4], int);
3 :    void array_output(int (*)[4], int);
4 :    int main()
5 :    {
6 :        int num_ary[3][4];
7 :        int row, col;
8 :
9 :        // 행의개수구하기
10:        row = sizeof(num_ary) / sizeof(num_ary[0]);
11:        // 열의개수구하기
12:        //col = sizeof(num_ary[0]) / sizeof(num_ary[0][0]);
13:        array_input(num_ary, row);
14:        array_output(num_ary, row);
15:        return 0;
16:    }
17: void array_input(int (*p)[4], int row)
18: {
19:     int i, j;
20:     for( i = 0 ; i < row ; i++ )
21:     {
22:         for( j = 0 ; j < 4 ; j ++ )
23:         {
24:             printf("%d행%d열 방의 값입력: " , i, j);
25:             scanf("%d", &p[i][j] ) ;
26:         }
27:     }
28:     return;
29: }
30: void array_output(int (*p)[4], int row)
31: {
32:
33:
34:
35:
36:
37:
38:
39:
40:
41:
42:
43:
44:
45:
46:
47:
48:
49:
50:
51:
52:
53:
54:
55:
56:
57:
58:
59:
60:
61:
62:
63:
64:
65:
66:
67:
68:
69:
70:
71:
72:
73:
74:
75:
76:
77:
78:
79:
80:
81:
82:
83:
84:
85:
86:
87:
88:
89:
90:
91:
92:
93:
94:
95:
96:
97:
98:
99:
100:
101:
102:
103:
104:
105:
106:
107:
108:
109:
110:
111:
112:
113:
114:
115:
116:
117:
118:
119:
120:
121:
122:
123:
124:
125:
126:
127:
128:
129:
130:
131:
132:
133:
134:
135:
136:
137:
138:
139:
140:
141:
142:
143:
144:
145:
146:
147:
148:
149:
150:
151:
152:
153:
154:
155:
156:
157:
158:
159:
160:
161:
162:
163:
164:
165:
166:
167:
168:
169:
170:
171:
172:
173:
174:
175:
176:
177:
178:
179:
180:
181:
182:
183:
184:
185:
186:
187:
188:
189:
190:
191:
192:
193:
194:
195:
196:
197:
198:
199:
200:
201:
202:
203:
204:
205:
206:
207:
208:
209:
210:
211:
212:
213:
214:
215:
216:
217:
218:
219:
220:
221:
222:
223:
224:
225:
226:
227:
228:
229:
230:
231:
232:
233:
234:
235:
236:
237:
238:
239:
240:
241:
242:
243:
244:
245:
246:
247:
248:
249:
250:
251:
252:
253:
254:
255:
256:
257:
258:
259:
260:
261:
262:
263:
264:
265:
266:
267:
268:
269:
270:
271:
272:
273:
274:
275:
276:
277:
278:
279:
280:
281:
282:
283:
284:
285:
286:
287:
288:
289:
290:
291:
292:
293:
294:
295:
296:
297:
298:
299:
300:
301:
302:
303:
304:
305:
306:
307:
308:
309:
310:
311:
312:
313:
314:
315:
316:
317:
318:
319:
320:
321:
322:
323:
324:
325:
326:
327:
328:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
340:
341:
342:
343:
344:
345:
346:
347:
348:
349:
350:
351:
352:
353:
354:
355:
356:
357:
358:
359:
360:
361:
362:
363:
364:
365:
366:
367:
368:
369:
370:
371:
372:
373:
374:
375:
376:
377:
378:
379:
380:
381:
382:
383:
384:
385:
386:
387:
388:
389:
390:
391:
392:
393:
394:
395:
396:
397:
398:
399:
400:
401:
402:
403:
404:
405:
406:
407:
408:
409:
410:
411:
412:
413:
414:
415:
416:
417:
418:
419:
420:
421:
422:
423:
424:
425:
426:
427:
428:
429:
430:
431:
432:
433:
434:
435:
436:
437:
438:
439:
440:
441:
442:
443:
444:
445:
446:
447:
448:
449:
450:
451:
452:
453:
454:
455:
456:
457:
458:
459:
460:
461:
462:
463:
464:
465:
466:
467:
468:
469:
470:
471:
472:
473:
474:
475:
476:
477:
478:
479:
480:
481:
482:
483:
484:
485:
486:
487:
488:
489:
490:
491:
492:
493:
494:
495:
496:
497:
498:
499:
500:
501:
502:
503:
504:
505:
506:
507:
508:
509:
510:
511:
512:
513:
514:
515:
516:
517:
518:
519:
520:
521:
522:
523:
524:
525:
526:
527:
528:
529:
530:
531:
532:
533:
534:
535:
536:
537:
538:
539:
540:
541:
542:
543:
544:
545:
546:
547:
548:
549:
550:
551:
552:
553:
554:
555:
556:
557:
558:
559:
560:
561:
562:
563:
564:
565:
566:
567:
568:
569:
570:
571:
572:
573:
574:
575:
576:
577:
578:
579:
580:
581:
582:
583:
584:
585:
586:
587:
588:
589:
590:
591:
592:
593:
594:
595:
596:
597:
598:
599:
600:
601:
602:
603:
604:
605:
606:
607:
608:
609:
610:
611:
612:
613:
614:
615:
616:
617:
618:
619:
620:
621:
622:
623:
624:
625:
626:
627:
628:
629:
630:
631:
632:
633:
634:
635:
636:
637:
638:
639:
640:
641:
642:
643:
644:
645:
646:
647:
648:
649:
650:
651:
652:
653:
654:
655:
656:
657:
658:
659:
660:
661:
662:
663:
664:
665:
666:
667:
668:
669:
670:
671:
672:
673:
674:
675:
676:
677:
678:
679:
680:
681:
682:
683:
684:
685:
686:
687:
688:
689:
690:
691:
692:
693:
694:
695:
696:
697:
698:
699:
700:
701:
702:
703:
704:
705:
706:
707:
708:
709:
710:
711:
712:
713:
714:
715:
716:
717:
718:
719:
720:
721:
722:
723:
724:
725:
726:
727:
728:
729:
730:
731:
732:
733:
734:
735:
736:
737:
738:
739:
740:
741:
742:
743:
744:
745:
746:
747:
748:
749:
750:
751:
752:
753:
754:
755:
756:
757:
758:
759:
760:
761:
762:
763:
764:
765:
766:
767:
768:
769:
770:
771:
772:
773:
774:
775:
776:
777:
778:
779:
780:
781:
782:
783:
784:
785:
786:
787:
788:
789:
790:
791:
792:
793:
794:
795:
796:
797:
798:
799:
800:
801:
802:
803:
804:
805:
806:
807:
808:
809:
810:
811:
812:
813:
814:
815:
816:
817:
818:
819:
820:
821:
822:
823:
824:
825:
826:
827:
828:
829:
830:
831:
832:
833:
834:
835:
836:
837:
838:
839:
840:
841:
842:
843:
844:
845:
846:
847:
848:
849:
850:
851:
852:
853:
854:
855:
856:
857:
858:
859:
860:
861:
862:
863:
864:
865:
866:
867:
868:
869:
870:
871:
872:
873:
874:
875:
876:
877:
878:
879:
880:
881:
882:
883:
884:
885:
886:
887:
888:
889:
890:
891:
892:
893:
894:
895:
896:
897:
898:
899:
900:
901:
902:
903:
904:
905:
906:
907:
908:
909:
910:
911:
912:
913:
914:
915:
916:
917:
918:
919:
920:
921:
922:
923:
924:
925:
926:
927:
928:
929:
930:
931:
932:
933:
934:
935:
936:
937:
938:
939:
940:
941:
942:
943:
944:
945:
946:
947:
948:
949:
950:
951:
952:
953:
954:
955:
956:
957:
958:
959:
960:
961:
962:
963:
964:
965:
966:
967:
968:
969:
970:
971:
972:
973:
974:
975:
976:
977:
978:
979:
980:
981:
982:
983:
984:
985:
986:
987:
988:
989:
990:
991:
992:
993:
994:
995:
996:
997:
998:
999:
1000:
1001:
1002:
1003:
1004:
1005:
1006:
1007:
1008:
1009:
1010:
1011:
1012:
1013:
1014:
1015:
1016:
1017:
1018:
1019:
1020:
1021:
1022:
1023:
1024:
1025:
1026:
1027:
1028:
1029:
1030:
1031:
1032:
1033:
1034:
1035:
1036:
1037:
1038:
1039:
1040:
1041:
1042:
1043:
1044:
1045:
1046:
1047:
1048:
1049:
1050:
1051:
1052:
1053:
1054:
1055:
1056:
1057:
1058:
1059:
1060:
1061:
1062:
1063:
1064:
1065:
1066:
1067:
1068:
1069:
1070:
1071:
1072:
1073:
1074:
1075:
1076:
1077:
1078:
1079:
1080:
1081:
1082:
1083:
1084:
1085:
1086:
1087:
1088:
1089:
1090:
1091:
1092:
1093:
1094:
1095:
1096:
1097:
1098:
1099:
1100:
1101:
1102:
1103:
1104:
1105:
1106:
1107:
1108:
1109:
1110:
1111:
1112:
1113:
1114:
1115:
1116:
1117:
1118:
1119:
1120:
1121:
1122:
1123:
1124:
1125:
1126:
1127:
1128:
1129:
1130:
1131:
1132:
1133:
1134:
1135:
1136:
1137:
1138:
1139:
1140:
1141:
1142:
1143:
1144:
1145:
1146:
1147:
1148:
1149:
1150:
1151:
1152:
1153:
1154:
1155:
1156:
1157:
1158:
1159:
1160:
1161:
1162:
1163:
1164:
1165:
1166:
1167:
1168:
1169:
1170:
1171:
1172:
1173:
1174:
1175:
1176:
1177:
1178:
1179:
1180:
1181:
1182:
1183:
1184:
1185:
1186:
1187:
1188:
1189:
1190:
1191:
1192:
1193:
1194:
1195:
1196:
1197:
1198:
1199:
1200:
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:
1209:
1210:
1211:
1212:
1213:
1214:
1215:
1216:
1217:
1218:
1219:
1220:
1221:
1222:
1223:
1224:
1225:
1226:
1227:
1228:
1229:
1230:
1231:
1232:
1233:
1234:
1235:
1236:
1237:
1238:
1239:
1240:
1241:
1242:
1243:
1244:
1245:
1246:
1247:
1248:
1249:
1250:
1251:
1252:
1253:
1254:
1255:
1256:
1257:
1258:
1259:
1260:
1261:
1262:
1263:
1264:
1265:
1266:
1267:
1268:
1269:
1270:
1271:
1272:
1273:
1274:
1275:
1276:
1277:
1278:
1279:
1280:
1281:
1282:
1283:
1284:
1285:
1286:
1287:
1288:
1289:
1290:
1291:
1292:
1293:
1294:
1295:
1296:
1297:
1298:
1299:
1300:
1301:
1302:
1303:
1304:
1305:
1306:
1307:
1308:
1309:
1310:
1311:
1312:
1313:
1314:
1315:
1316:
1317:
1318:
1319:
1320:
1321:
1322:
1323:
1324:
1325:
1326:
1327:
1328:
1329:
1330:
1331:
1332:
1333:
1334:
1335:
1336:
1337:
1338:
1339:
1340:
1341:
1342:
1343:
1344:
1345:
1346:
1347:
1348:
1349:
1350:
1351:
1352:
1353:
1354:
1355:
1356:
1357:
1358:
1359:
1360:
1361:
1362:
1363:
1364:
1365:
1366:
1367:
1368:
1369:
1370:
1371:
1372:
1373:
1374:
1375:
1376:
1377:
1378:
1379:
1380:
1381:
1382:
1383:
1384:
1385:
1386:
1387:
1388:
1389:
1390:
1391:
1392:
1393:
1394:
1395:
1396:
1397:
1398:
1399:
1400:
1401:
1402:
1403:
1404:
1405:
1406:
1407:
1408:
1409:
1410:
1411:
1412:
1413:
1414:
1415:
1416:
1417:
1418:
1419:
1420:
1421:
1422:
1423:
1424:
1425:
1426:
1427:
1428:
1429:
1430:
1431:
1432:
1433:
1434:
1435:
1436:
1437:
1438:
1439:
1440:
1441:
1442:
1443:
1444:
1445:
1446:
1447:
1448:
1449:
1450:
1451:
1452:
1453:
1454:
1455:
1456:
1457:
1458:
1459:
1460:
1461:
1462:
1463:
1464:
1465:
1466:
1467:
1468:
1469:
1470:
1471:
1472:
1473:
1474:
1475:
1476:
1477:
1478:
1479:
1480:
1481:
1482:
1483:
1484:
1485:
1486:
1487:
1488:
1489:
1490:
1491:
1492:
1493:
1494:
1495:
1496:
1497:
1498:
1499:
1500:
1501:
1502:
1503:
1504:
1505:
1506:
1507:
1508:
1509:
1510:
1511:
1512:
1513:
1514:
1515:
1516:
1517:
1518:
1519:
1520:
1521:
1522:
1523:
1524:
1525:
1526:
1527:
1528:
1529:
1530:
1531:
1532:
1533:
1534:
1535:
1536:
1537:
1538:
1539:
1540:
1541:
1542:
1543:
1544:
1545:
1546:
1547:
1548:
1549:
1550:
1551:
1552:
1553:
1554:
1555:
1556:
1557:
1558:
1559:
1560:
1561:
1562:
1563:
1564:
1565:
1566:
1567:
1568:
1569:
1570:
1571:
1572:
1573:
1574:
1575:
1576:
1577:
1578:
1579:
1580:
1581:
1582:
1583:
1584:
1585:
1586:
1587:
1588:
1589:
1590:
1591:
1592:
1593:
1594:
1595:
1596:
1597:
1598:
1599:
1600:
1601:
1602:
1603:
1604:
1605:
1606:
1607:
1608:
1609:
1610:
1611:
1612:
1613:
1614:
1615:
1616:
1617:
1618:
1619:
1620:
1621:
1622:
1623:
1624:
1625:
1626:
1627:
1628:
1629:
1630:
1631:
1632:
1633:
1634:
1635:
1636:
1637:
1638:
1639:
1640:
1641:
1642:
1643:
1644:
1645:
1646:
1647:
1648:
1649:
1650:
1651:
1652:
1653:
1654:
1655:
1656:
1657:
1658:
1659:
1660:
1661:
1662:
1663:
1664:
1665:
1666:
1667:
1668:
1669:
1670:
1671:
1672:
1673:
1674:
1675:
1676:
1677:
1678:
1679:
1680:
1681:
1682:
1683:
1684:
1685:
1686:
1687:
1688:
1689:
1690:
1691:
1692:
1693:
1694:
1695:
1696:
1697:
1698:
1699:
1700:
1701:
1702:
1703:
1704:
1705:
1706:
1707:
1708:
1709:
1710:
1711:
1712:
1713:
1714:
1715:
1716:
1717:
1718:
1719:
1720:
1721:
1722:
1723:
1724:
1725:
1726:
1727:
1728:
1729:
1730:
1731:
1732:
1733:
1734:
1735:
1736:
1737:
1738:
1739:
1740:
1741:
1742:
1743:
1744:
1745:
1746:
1747:
1748:
1749:
1750:
1751:
1752:
1753:
1754:
1755:
1756:
1757:
1758:
1759:
1760:
1761:
1762:
1763:
1764:
1765:
1766:
1767:
1768:
1769:
1770:
1771:
1772:
1773:
1774:
1775:
1776:
1777:
1778:
1779:
1780:
1781:
1782:
1783:
1784:
1785:
1786:
1787:
1788:
1789:
1790:
1791:
1792:
1793:
1794:
1795:
1796:
1797:
1798:
1799:
1800:
1801:
1802:
1803:
1804:
1805:
1806:
1807:
1808:
1809:
1810:
1811:
1812:
1813:
1814:
1815:
1816:
1817:
1818:
1819:
1820:
1821:
1822:
1823:
1824:
1825:
1826:
1827:
1828:
1829:
1830:
1831:
1832:
1833:
1834:
1835:
1836:
1837:
1838:
1839:
1840:
1841:
1842:
1843:
1844:
1845:
1846:
1847:
1848:
1849:
1850:
1851:
1852:
1853:
1854:
1855:
1856:
1857:
1858:
1859:
1860:
1861:
1862:
1863:
1864:
1865:
1866:
1867:
1868:
1869:
1870:
1871:
1872:
1873:
1874:
1875:
1876:
1877:
1878:
1879:
1880:
1881:
1882:
1883:
1884:
1885:
1886:
1887:
1888:
1889:
1890:
1891:
1892:
1893:
1894:
1895:
1896:
1897:
1898:
1899:
1900:
1901:
1902:
1903:
1904:
1905:
1906:
1907:
1908:
1909:
1910:
1911:
1912:
1913:
1914:
1915:
1916:
1917:
1918:
1919:
1920:
1921:
1922:
1923:
1924:
1925:
1926:
1927:
1928:
1929:
1930:
1931:
1932:
1933:
1934:
1935:
1936:
1937:
1938:
1939:
1940:
1941:
1942:
1943:
1944:
1945:
1946:
1947:
1948:
1949:
1950:
1951:
1952:
1953:
1954:
1955:
1956:
1957:
1958:
1959:
1960:
1961:
1962:
1963:
1964:
1965:
1966:
1967:
1968:
1969:
1970:
1971:
1972:
1973:
1974:
1975:
1976:
1977:
1978:
1979:
1980:
1981:
1982:
1983:
1984:
1985:
1986:
1987:
1988:
1989:
1990:
1991:
1992:
1993:
1994:
1995:
1996:
1997:
1998:
1999:
2000:
2001:
2002:
2003:
2004:
2005:
2006:
2007:
2008:
2009:
2010:
2011:
2012:
2013:
2014:
2015:
2016:
2017:
2018:
2019:
2020:
2021:
2022:
2023:
2024:
2025:
2026:
2027:
2028:
2029:
2030:
2031:
2032:
2033:
2034:
2035:
2036:
2037:
2038:
2039:
2040:
2041:
2042:
2043:
2044:
2045:
2046:
2047:
2048:
2049:
2050:
2051:
2052:
2053:
2054:
2055:
2056:
2057:
2058:
2059:
2060:
2061:
2062:
2063:
2064:
2065:
2066:
2067:
2068:
2069:
2070:
2071:
2072:
2073:
2074:
2075:
2076:
2077:
2078:
2079:
2080:
2081:
2082:
2083:
2084:
2085:
2086:
2087:
2088:
2089:
2090:
2091:
2092:
2093:
2094:
2095:
2096:
2097:
2098:
2099:
2100:
2101:
2102:
2103:
2104:
2105:
2106:
2107:
2108:
2109:
2110:
2111:
2112:
2113:
2114:
2115:
2116:
2117:
2118:
2119:
2120:
2121:
2122:
2123:
2124:
2125:
2126:
2127:
2128:
2129:
2130:
2131:
2132:
2133:
2134:
2135:
2136:
2137:
2138:
2139:
2140:
2141:
2142:
2143:
2144:
2145:
2146:
2147:
2148:
2149:
2150:
2151:
2152:
2153:
2154:
2155:
2156:
2157:
2158:
2159:
2160:
2161:
2162:
2163:
2164:
2165:
2166:
2167:
2168:
2169:
2170:
2171:
2172:
2173:
2174:
2175:
2176:
2177:
2178:
2179:
2180:
2181:
2182:
2183:
2184:
2185:
218
```

[예제 8-6] 이차원 문자 배열 예제

```

1 : #include <stdio.h>
2 : void string_print(char (*)[6], int);
3 : int main()
4 : {
5 :     char str[3][6] = {"Year", "Month", "Day" };
6 :     int row;
7 :     row = sizeof(str) / sizeof(str[0]);
8 :
9 :     string_print(str, row);
10:
11:    return 0;
12: }
13: void string_print(char (*cp)[6], int row)
14: {
15:     int i;
16:     for( i = 0 ; i < row ; i++ )
17:     {
18:         printf("%d행의문자열은%s이고, 첫글자는%c입니다\n",
19:             i , &cp[i][0], cp[i][0] ) ;
20:     }
21:     return;
22: }

```

4. 배열과 증감 연산자

ary[i++] : ary[i] 값을 사용후 i를 1 증가 시킨다
 ary [++i] : i의 값을 우선 1 증가 시킨후 ary[i] 값을 사용한다
 -- ary [i] : ary[i] 값을 1감소 시킨후 ary[i]를 사용한다
 ary [i]-- : ary[i] 값을 사용후 ary[i]를 1 감소 시킨다

```

int ary[4] = {10,20,30,40};
int i = 0, n;
n = ary[i++];
n = ary[++i];
n = --ary[i];
n = ary[i--];

```

[예제 8-7] 배열과 증감연산자 예제

```

1 :   #include <stdio.h>
2 :   int  main()
3 :   {
4 :       char  str[8] = "Dream" ;
5 :       int  i = 2;
6 :       printf("문자열은 %s 입니다 \n" ,  str ) ;
7 :       printf("%c \n" , str[i++] ) ;
8 :       printf("%c \n" , str[++i] ) ;
9 :       printf("%c \n" , --str[i] ) ;
10:      printf("%c \n" , str[i]-- ) ;
11:      printf("문자열은 %s 입니다 \n" ,  str ) ;
12:      return 0;
13:  }
```

5. 배열선언 및 초기화 시의 주의점

▶ 배열 선언 시 첨자 생략 규칙

1. 배열 선언 시 첨자를 생략하기 위해서는 꼭 선언문에서 배열을 초기화 해야 한다.
2. 배열의 첨자 중 가장 큰 첨자 1개만 생략이 가능하다. (면 > 행 > 열)

```

int  a[4] = {1,2,3,4} ;           ( )
int  a[4] = {1,2,3} ;            ( )
int  a[4] = {1,2,3,4,5} ;        ( )
int  a[] = {1,2,3} ;             ( )
int  a[4] ;                      ( )
int  a[] ;                       ( )
int  a[][4] = {{1,1,1,1} , {2,2,2,2} ,{3,3,3,0}} ;   ( )
int  a[][4] = {1,1,1,1,2,2,2,2,3,3,3,0} ;            ( )
int  a[][4] = {1,1,1,1,2,2,2,2,3,3,3} ;              ( )
int  a[][] = {{1,1,1,1} , {2,2,2,2} ,{3,3,3,0}} ;    ( )
char a[][7] = {"apple" , "pear" , "banana"} ;       ( )
char a[][] = {"apple" , "pear" , "banana"} ;        ( )
```

[예제 8-8] 입력된 문자열중 각 영문자의 발생 빈도수 출력 예제

```
1 :    #include <stdio.h>
2 :    int  main()
3 :    {
4 :        int  c, i ,large[26] = {0,} , small[26] = {0,};
5 :        int size = sizeof(large) / sizeof(large[0]);
6 :        while((c = getchar()) != EOF)
7 :        {
8 :            if(c >= 'A' && c <= 'Z') ++ large[c - 'A'] ;
9 :            else if(c >= 'a' && c <= 'z') ++ small[c - 'a'] ;
10:        }
11:        for( i = 0 ; i < size ; i++ )
12:        {
13:            if( i % 6 == 0) printf("\n") ;
14:            printf("%4c : %3d", 'A' + i , large[i]) ;
15:        }
16:        for( i = 0 ; i < size ; i++ )
17:        {
18:            if( i % 6 == 0) printf("\n") ;
19:            printf("%4c : %3d", 'a' + i , small[i]) ;
20:        }
21:        return 0;
22:    }
```

6. 배열에 file내의 데이터 저장하기

파일에 저장되어있는 데이터를 읽어 배열에 저장하는방법 이해하고 활용해보자. (fileTest1.txt)

```

3  ← 테스트 건수 (총 3set의 데이터를 읽어들임)
1 4 3 2 9 7 18 22 0  ← 첫번 째 테스트 데이터 그룹 (0은 종료표시로 실제 데이터는 아님)
2 4 8 10 0           ← 두번 째 테스트 데이터 그룹
7 5 11 13 1 3 0      ← 세번 째 테스트 데이터 그룹

```

[예제 8-9] 배열에 file내의 데이터 저장하기

```

1 :    #include <stdio.h>
2 :    FILE * fileOpen(char *filename, char *mode);
3 :    int fileRead(FILE *fp, int *dataArray);
4 :    int main(int argc, char *argv[])
5 :    {
6 :        int testCnt;
7 :        int dataCnt;
8 :        int dataArray[100];
9 :        int i,j;
10:        FILE *fp;
11:        fp = fileOpen("d:\\data\\fileTest1.txt", "rt");
12:        if(fp == NULL) return 1;
13:
14:        fscanf(fp, "%d", &testCnt); // 총 테스트 건수를 읽어들임
15:        for(i=0; i<testCnt; i++){ // 테스트 건수만큼 반복작업
16:            dataCnt = fileRead(fp, dataArray);
17:            for(j=0; j<dataCnt; j++)
18:                printf("%5d", dataArray[j]);
19:            printf("\n");
20:        }
21:        fclose(fp);
22:        return 0;
23:    }
24:    FILE * fileOpen(char *filename, char *mode)
25:    {
26:        FILE *fp;
27:        if(( fp = fopen(filename, mode))==NULL){
28:            printf("File open error!\n");
29:        }
30:        return fp;
31:    }
32:    int fileRead(FILE *fp, int *dataArray)
33:    {
34:        int dataCnt = 0;
35:        while(1){
36:            fscanf(fp, "%d", &dataArray[dataCnt]);
37:            if(dataArray[dataCnt]==0) break;
38:            dataCnt++;
39:        }
39:        return dataCnt;
40:    }

```

제9장 기억클래스

학습 내용 소개

9장에서는 프로그램 내에서 사용되는 각종 변수들의 특성을 기억클래스로 나누어 그 특징을 파악하고 프로그래밍 시에 적절한 변수를 선택하여 사용할 수 있는 기법을 익힌다.

학습 목차

1. 기억클래스의 기본개념
2. auto 기억클래스의 이해
3. extern 기억클래스의 이해
4. static 기억클래스의 이해
5. register 기억클래스의 이해
6. 프로젝트 기법

1. 기억클래스(Storage class)의 기본 개념

▶ 기억클래스란?

- 기억클래스(Storage class)란 변수의 유효범위(Scope), 생존기간(Life time), Memory내에서의 위치, 자동초기화 여부 등 프로그램 내에서 사용되는 변수들의 성격을 결정하는 프로그램 요소이다.
- 기억클래스의 특징을 파악하여 프로그래밍 시에 적절한 변수를 선택하여 사용할 수 있는 기법을 익히는 것이 중요하다.

▶ 기억클래스 지정하기

[Storage class] Type 변수명 ;

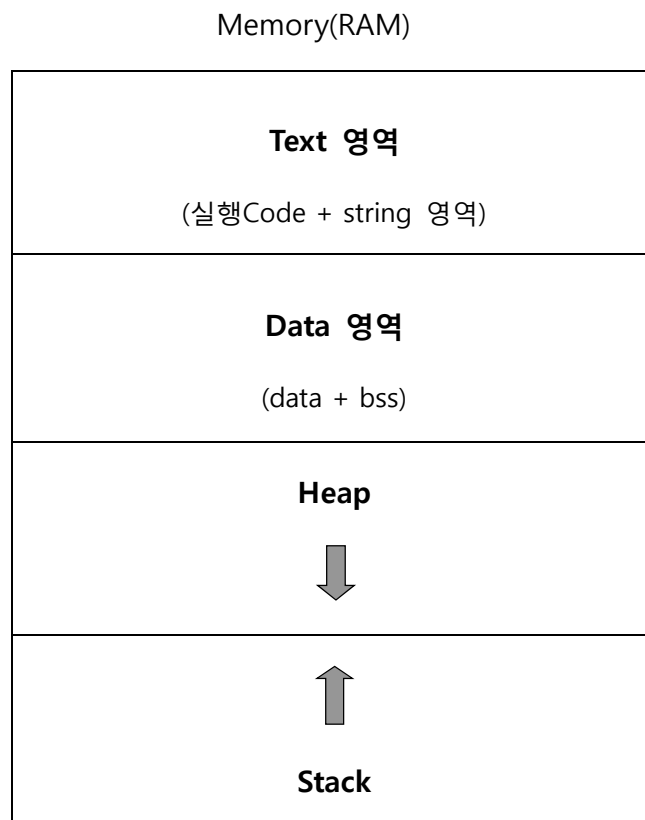
예) auto int num;

[Storage class] 생략 시 block 내에서는 auto로 block밖에서는 extern으로 인식

▶ 실행파일과 메모리 영역

C언어로 작성한 실행파일이 Memory(RAM)내에서 실행될 때에는 Code영역에 실행파일이 올라가고

Data영역에 프로그램 내에서 선언된 변수나 동적메모리 할당된 기억공간이 잡히게 된다.



▶ 기억클래스표

기억부류 지정자		유효범위(scope)	생존기간 (life time)	Memory에서의 위치	자동 초기화
auto(자동변수)		선언된 block{}내	block 종료 시 까지	Stack	X
extern (외부변수)		한 module 전체 program	program 종료 시 까지	Data 영역	O
static (정적 변수)	내부static	선언된 block{}내	program 종료 시 까지	Data 영역	O
	외부static	선언된 module			
register (레지스터변수)		선언된 block{}내	block 종료 시 까지	CPU내 register	X

2. auto 기억 클래스의 이해

[예제 9-1] auto 변수값의 출력

```

1 :   #include<stdio.h>
2 :   int  main()
3 :   {
4 :       auto int  a = 1; // C++ 컴파일러를 사용 시 auto를 표기하면 에러 발생
5 :       printf("a = %d \n" , a) ;
6 :       {
7 :           int  a = 2 ;
8 :           printf("a = %d \n" , a) ;
9 :           {
10:              a += 1 ;
11:              printf("a = %d \n" , a) ;
12:           }
13:           printf("a = %d \n" , a) ;
14:       }
15:       printf("a = %d \n" , a) ;
16:       return 0;
17:   }
```

[예제 9-2] parameter의 기억클래스는?

```

1 :   #include<stdio.h>
2 :   void workover( int ) ;
3 :   int reset( int ) ;
4 :   int main()
5 :   {
6 :       int a = 5 ;
7 :       reset(a / 2) ;      printf(“%d\n”, a) ;
8 :       reset(a /= 2) ;     printf(“%d\n”, a) ;
9 :       a = reset(a / 2) ;  printf(“%d\n”, a) ;
10:      workover(a) ;        printf(“%d\n”, a) ;
11:      return 0;
12:  }
13:  void workover( int a )
14:  {
15:      a = ( (a * a) / (2 * a) + 4 ) * (a % a) ;
16:      printf(“%d\n”, a) ;
17:  }
18:  int reset ( int a )
19:  {
20:      a = (a < 2) ? 5 : 0 ;
21:      return (a) ;
22:  }

```

** parameter의 기억클래스는 무조건 auto 이다.

3. extern 기억클래스의 이해

3.1 auto와 extern기억클래스 변수로 구성된 프로그램의 실행 흐름 보기

```

#include<stdio.h>
int ex;
int main()
{
    int a=1;
    :
    sub(a);
    :
}
void sub(int n)
{
    double da;
    :
}

```

[예제 9-3] 외부변수(extern)와 자동변수(auto)의 차이

```

1 :    #include <stdio.h>
2 :    void sum() ;
3 :    int temp ;
4 :    int main()
5 :    {
6 :        int a = 10 ;
7 :        sum () ;
8 :        printf("temp = %d \n" , temp) ;
9 :        printf("a = %d \n" , a) ;
10:        return 0;
11:    }
12:    void sum ()
13:    {
14:        temp += 100 ;
15:        /* printf("a = %d \n" , a) ; */
16:    }

```

3.2 extern 변수의 선언과 정의의 차이

하나의 프로그램이 여러 개의 소스파일로 이루어져있을 때 각 소스파일에서 extern변수를 사용하려면 하나의 파일에서 정의하고 나머지 파일에서는 선언해주어야 한다.

- extern 변수의 정의 : 실제 메모리 block이 할당됨.
 - 초기화 가능
 - 전체 프로그램 내에서 딱 1회만 정의함
 - block 밖에서 int a;와 같은 형태로 정의
- extern 변수의 선언 : compiler에게 extern변수에 대한 정보를 줌
 - 초기화 불가능
 - 여러 번 선언 가능
 - 프로그램의 어떤 위치에서든 선언가능하며
 - extern int a; 와 같은 형태로 선언

** extern 배열은 정의부의 첨자 생략이 불가능하나 선언부는 첨자 생략이 가능함

4. static 기억클래스의 이해

4.1 auto, extern, static 변수로 구성된 프로그램의 실행 흐름 보기

```
#include<stdio.h>
int ex;
static long st;
int main()
{
    int a=2;
    :
    sub(a);
    :
}
void sub(int n)
{
    static char ch;
    double da;
    :
}
```

[예제 9-4] auto 변수와 내부static 변수의 차이

```
1 :      #include <stdio.h>
2 :      void sub() ;
3 :      int main()
4 :      {
5 :          int i ;
6 :          for ( i = 0 ; i < 3 ; i++ )
7 :          {
8 :              sub () ;
9 :              printf("main i = %d \n" , i) ;
10:         }
11:         return 0;
12:     }
13:     void sub ()
14:     {
15:         static int i = 1;
16:         auto int k = 3;
17:         printf("sub i = %d \t k = %d \t" , i++ , k++) ;
18:     }
```

[예제 9-5] 틀리기 쉬운 기억클래스 예제

```

1 :      #include <stdio.h>
2 :      int  reset(), next(int) , last(int) , now( int ) ;
3 :      int  i = 1 ;
4 :      int main()
5 :      {
6 :          int  i, j ;
7 :          i = reset() ;
8 :          for ( j = 0 ; j < 3 ; j++ )
9 :          {
10:             printf(" i = %d \t j = %d \n" , i, j ) ;
11:             printf("next(i) = %d\n", next(i) ) ;
12:             printf("last(i) = %d\n", last(i) ) ;
13:             printf("now( i + j ) = %d\n", now( i + j ) ) ;
14:          }
15:          return 0;
16:      }
17:      int  reset() { return (i) ; }
18:
19:      int  next(int j){ return (j = i ++ ) ; }
20:
21:      int  last(int j){
22:          static int  i = 10 ;
23:          return (j = i --) ;
24:      }
25:      int  now(int i)
26:      {
27:          int  j = 10 ;
28:          return (i = j += i) ;
29:      }

```

▶ static 함수

static 키워드를 이용해서 선언 및 정의해주는 함수가 static 함수이다.

static 함수는 사용 영역이 자신이 속한 하나의 file내로 제한된다. (file scope)

(main.c)	(sub.c)
<pre> #include<stdio.h> static void abc(); void sub(); int main() { abc(); sub(); return 0; } static void abc() { printf("main ()함수 호출 abc 함수\n"); } </pre>	<pre> #include<stdio.h> static void abc(); void sub() { abc(); return; } static void abc() { printf("sub()함수 호출 abc 함수\n"); } </pre>

5. register 기억클래스의 이해

- CPU 내의 register를 변수로 사용한다.
- 연산작업을 빠르게 하기 위하여 사용된다.
- register 기억클래스는 사용자의 요청일 뿐 실제 잡히는 영역은 시스템 결정한다.

[예제 9-6] 레지스터(register)변수

```
1 :    #include<stdio.h>
2 :    int  main()
3 :    {
4 :        register int  i;
5 :        long  sum = 0 ;
6 :        for ( i = 1 ; i <= 20000 ; i++ )
7 :        {
8 :            sum += i ;
9 :        }
10:        printf("1 부터 20000 까지의 합은 %ld 입니다\n" ,sum) ;
11:        return 0;
12:    }
```

6. 프로젝트

여러 개의 file로 구성된 프로그램을 compile, link 하여 하나의 실행파일을 생성해내는 것을 프로젝트 기법이라고 한다.

▶ 프로젝트 실습해 보기

[예제 9-7-1] 실행파일 abc.exe 를 생성하기 위한 첫번째 모듈 (a.c)

```

1 :   #include<stdio.h>
2 :   int  reset() , next() , last() ;
3 :   int  now( int ) ;
4 :   void main()
5 :   {
6 :       auto int i, j ;
7 :       i = reset() ;
8 :       for ( j = 0 ; j <= 3 ; j++ ) {
9 :           printf("i = %d \t j = %d \n" , i, j ) ;
10:          printf("next() = %d\n", next() ) ;
11:          printf("last() = %d\n", last() ) ;
12:          printf("next( i + j ) = %d\n", now( i + j ) ) ;
13:      }
14:  }
```

[예제 9-7-2] 실행파일 abc.exe 를 생성하기 위한 두번째 모듈 (b.c)

```

1 :   extern int i ;
2 :   int  reset()
3 :   {
4 :       return (i) ;
5 :   }
```

[예제 9-7-3] 실행파일 abc.exe 를 생성하기 위한 세번째 모듈 (c.c)

```

1 :   int  i = 10 ;
2 :   int  next()
3 :   {
4 :       return (i += 1) ;
5 :   }
6 :   int  last()
7 :   {
8 :       return (i -= 1) ;
9 :   }
10:  int  now(int i)
11:  {
12:      static int j = 5 ;
13:      return (i = j += i) ;
14:  }
```

제10장 선행처리자

학습 내용 소개

10장에서는 전처리가 처리하는 명령어인 선행처리자에 대해 이해하고 대표적인 선행처리자에 대해서 공부한다.

include를 이용한 헤더파일의 포함과 define을 이용한 매크로 정의는 C프로그램에서는 없어서 안될 매우 중요한 요소이며, 기타 유용한 선행처리자에 대한 활용 기법을 익히자.

학습 목차

1. 선행처리자란?
2. #include
3. #define [macro의 정의]
4. 기타 다양한 선행처리자(preprocessor)

1. 선행처리자란?

C source code 내에는 Preprocessor가 처리하는 명령을 포함시킬 수 있는데 이러한 명령들을 선행처리자(Preprocessor 지시자)라 하며, C 프로그램 환경의 통용범위를 확장해 주는 기능을 갖는다.

선행처리기(preprocessor)는 compile전에 동작하며 프로그래머가 프로그래밍하기 좋은 환경을 제공한다

선행처리자(preprocessor 지시자)는 Preprocessor에 의해 처리되며 각 명령에 따라 각기 다른 기능이 수행된다

2. #include

compiler가 compile을 시작하기 전에 Preprocessor는 #include 지시자가 지시하는 header file을 현재 위치에 포함시킨다.

[일반형식]

```
#include <header_file>
#include "header_file"
```

[설명]

< >

Compiler를 제공하는 Maker에서 제공하는 Header file을 포함할 때 사용
(Option에서 미리 설정된 폴더에서 header file을 찾음)

“ ”

사용자가 작성한 Header file을 포함할 때 사용

예)

```
#include <stdio.h>
#include "user.h"
#include <c:\w\include\sys\time.h>
#include "d:\sample\abc.h"
```

※ 사용자 헤더파일에 저장되는 내용

- 사용자 정의 함수 선언부
- extern 변수 선언부
- Macro 상수 및 Macro 함수 선언부
- 다른 header file 포함부(include)
- 구조체, 공용체, 나열형과 같은 복합 데이터형의 형틀 선언부(template definition)
- 데이터형 정의(type definition) – typedef 명령어에 의한 형명재지정 등...

3. #define[매크로 정의]

프로그램내에서 Compiler가 매크로를 만났을 때 대체할 문자열을 정의한다

3.1 매크로 상수와 매크로 함수

compiler가 compile을 시작하기 전 Preprocessor는 #define 으로 정의된 기호상수 등을 원래의 확장 문자열로 확장한다

[일반형식]

#define	매크로상수명	매크로 확장 문자열
#define	매크로함수명(전달인자)	매크로 확장 문자열

- Macro 상수 : 치환 문자열로 단순치환(단순확장)
- Macro 함수 : 전달인자의 형태를 고려해서 변환치환(변환확장)

[설명]

- 매크로명은 보통 대문자를 사용하며 매크로명과 치환 문자열은 하나 이상의 공백으로 구분된다

예) #define PI 3.141592
 #define PC "Personal Computer"
 #define PRN printf("%d\\n",x)

- 매크로는 프로그램을 읽기 쉽게하고, 유지보수를 용이하게 해주며, 변수 사용 시 보다 처리속도를 높여준다.

- 상수식은 반드시 괄호로 묶는다

예) #define P (3+5) ==> P * 2 ==> (3+5)*2 ==> 16
 #define P 3+5 ==> P * 2 ==> 3+5*2 ==> 13

[예제 10-1] 매크로 상수와 매크로 함수

```
1 : #include <stdio.h>
2 : #define PI 3.14
3 : #define MAX(x,y) ( x > y ) ? x : y
4 :
5 : int main()
6 : {
7 :     int a = 123, b = 456;
8 :     printf("PI = %lf\n",PI) ;
9 :     printf("The Max is %d\n" , MAX(a,b)) ;
10:    return 0;
11: }
```

[예제 10-2] 매크로 함수의 사용

```
1 : #include <stdio.h>
2 : #define SWAP(x,y) {int t ; t=x ; x=y ; y=t ; }
3 :
4 : int main()
5 : {
6 :     int a = 5, b = 10;
7 :     printf("a = %d\t b = %d\n" , a, b ) ;
8 :     SWAP(a,b) ;
9 :     printf("a = %d\t b = %d\n" , a, b ) ;
10:    return 0;
11: }
```

[예제 10-3] 매크로 함수의 주의사항

```
1 : #include <stdio.h>
2 : #define SQUARE(x) x * x
3 : #define PRT(x) printf("%d\n",x)
4 : int main()
5 : {
6 :     int x = 4 ;
7 :
8 :     PRT ( SQUARE(x+2) ) ;
9 :     PRT ( 100 / SQUARE(2) ) ;
10:    PRT ( SQUARE(++x) ) ;
11:    return 0;
12: }
```

3.2 매크로의 다양한 활용

▶ 매크로 전달인자로 문자열 만들기 : # 연산자

매크로 함수 작성 시 #연산자를 사용하면 매크로 확장 문자열 안에 전달인자를 문자열화 시켜 포함시킬 수 있다.

[예제 10-4] 매크로 인수의 출력

```

1 : #include <stdio.h>
2 : #define PRINT(x) printf(#x " = %d\n", x)
3 :
4 : int main()
5 : {
6 :     int a = 5, b = 10;
7 :     PRINT(a + b) ;
8 :     PRINT(a * 3) ;
9 :     PRINT(b / 5) ;
10:    PRINT( (100 + a) * 2 ) ;
11:    return 0;
12: }
```

[문제] 다음의 결과 나오도록 매크로 함수를 완성해보시오.

<pre> #include<stdio.h> #define PRN(x,y) _____ int main() { int iVal = 10; double dVal = 3.5; char strVal[20] = "strawberry"; PRN(iVal, d); PRN(dVal, .2lf); PRN(strVal, s); return 0; }</pre>	<p>(실행 결과)</p> <p>iVal= 10</p> <p>dVal= 3.50</p> <p>strVal= strawberry</p>
---	--

▶ 토큰 결합 : ## 연산자

매크로 작성시 ## 연산자를 사용하면 매크로의 대체 리스트를 만들어 낼 수 있다.

즉, ## 연산자는 두 개의 토큰을 하나의 토큰으로 결합한다.

[예제 10-5] ## 연산자를 이용한 매크로 상수 및 함수 생성

```

1 :  #include<stdio.h>
2 :  #define MAKENAME(variable, number)  variable##number
3 :  #define PRN(var, num) printf("#var #num"="%d\n", MAKENAME(var, num))
4 :  int main()
5 :  {
6 :      int MAKENAME(a,1) = 10;
7 :      int MAKENAME(x,2) = 20;
8 :      PRN(a,1);
9 :      PRN(x,2);
10:     //PRN(y,3);
11:     return 0;
12: }
```

3.3 매크로함수와 실제 함수의 차이

- 실제 함수 : argument의 데이터형을 정밀하게 사용할 수 있다.

함수정의부가 1회만 기계어 코드로 변환되므로 실행파일의 크기가 커지는 단점을 해결할 수 있다. 함수 호출의 모든 프로세스를 수행 해야 하므로 처리 속도가 느려짐.

- 매크로 함수 : argument의 데이터형을 자유롭게 사용할 수 있으며 실제 호출이 아니라 매크로 확장 문자열로 치환되어 수행된다.(처리 속도 빠름)

[예제 10-6] 매크로함수와 일반함수의 차이

```

1 :  #include<stdio.h>
2 :  #define SQUARE(x) x*x  // (x)*(x) 로 수정하여 수행해보자
3 :  int square(int);
4 :  int main()
5 :  {
6 :      int res;
7 :      res=square(3+4);
8 :      printf("일반 함수를 이용한 3+4의 거듭제곱 : %d\n", res);
9 :      res=SQUARE(3+4);
10:     printf("매크로 함수를 이용한 3+4의 거듭제곱 : %d\n", res);
11:     return 0;
12: }
13: int square(int x)
14: {
15:     return x*x;
16: }
```

4. 기타 다양한 선행처리자(preprocessor)

▶ #undef 지시자 : #define 정의를 무효화 함

#define은 자신이 정의된 위치에서 부터 파일의 끝 또는 #undef지시자에 의해 취소되는 위치까지 유효하다.

<pre>#include<stdio.h> #define LIMIT 100 void sub(); int main() { printf("%d\n", LIMIT); sub(); return 0; } void sub() { #undef LIMIT #define LIMIT 150 printf("%d\n", LIMIT); }</pre>	<p>(다음과 같은 사용시 문제가 없을까?)</p> <pre>#include<stdio.h> void sub(); int main() { sub(); return 0; } void sub() { #undef LIMIT; #define LIMIT 150 printf("%d\n", LIMIT); }</pre>
--	---

▶ #ifdef, #else, #endif 지시자 : 조건부 컴파일을 하기 위해 사용

<pre>(my.h) #define FINISH //#define CONTINUE</pre>	
<pre>#include<stdio.h> #include "my.h" #ifdef FINISH int sleepingHours=7; #define MSG "고생했어요~" #else #ifdef CONTINUE int sleepingHours=4; #define MSG "열프~ 즐프~" #else int sleepingHours=0; #define MSG "날새 보아요~" #endif #endif int main() { printf("%s\n", MSG); printf("오늘의 수면시간은 %d시간 입니다.\n", sleepingHours); return 0; }</pre>	

▶ #if, #elif, #else, #endif 지시자 : C의 if 명령문 처럼 논리연산자(==, !=, >, >=, <, <=)를 사용 하여 조건부 컴파일을 처리 할 수 있다.

(예1) #if를 이용한 조건부 컴파일1

<pre>(my.h) #define CLASS 2 //#define CLASS 1 //#define CLASS 3 (one.h) #define STUDENT_CNT 10 (two.h) #define STUDENT_CNT 20 (three.h) #define STUDENT_CNT 30</pre>	<pre>#include <stdio.h> #include "my.h" #if CLASS == 1 #include "one.h" #elif CLASS == 2 #include "two.h" #else #include "three.h" #endif int main() { printf("%d반의 학생수는 %d명 입니다.\n", CLASS, STUDENT_CNT); return 0; }</pre>
---	---

(예2) #if를 이용한 조건부 컴파일2

<pre>(my.h) //#define IBMPCL //#define VAX //#define MAC (ibmpc.h) #define MSG "IBMPCL 시스템 환경으로 세팅되었습니다." (vax.h) #define MSG "VAX 시스템 환경으로 세팅되었습니다." (mac.h) #define MSG "MAC 시스템 환경으로 세팅되었습니다." (general.h) #define MSG "일반 시스템 환경으로 세팅되었습니다."</pre>	<pre>#include<stdio.h> #include "my.h" #if defined (IBMPCL) #include "ibmpc.h" #elif defined (VAX) #include "vax.h" #elif defined (MAC) #include "mac.h" #else #include "general.h" #endif int main() { printf("%s\n",MSG); return 0; }</pre>
--	---

- ▶ 미리 정의된 매크로 : C언어에서는 미리 정의되어 제공되는 매크로들은 프로그램의 수행상황을 사용자에게 알려주는 역할을 한다.

[예제 10-7] 미리 정의된 매크로

```

1 :    #include<stdio.h>
2 :    void sub();
3 :    int main()
4 :    {
5 :        printf("파일명 : %s\n", __FILE__);
6 :        printf("수행날짜 : %s\n", __DATE__);
7 :        printf("수행시간 : %s\n", __TIME__);
8 :        printf("현재 라인 번호 : %d\n", __LINE__);
9 :        sub();
10:        return 0;
11:    }
12:    void sub()
13:    {
14:        printf("현재 라인 번호 : %d\n", __LINE__);
15:    }

```

- ▶ #pragma 지시자 : 소스코드 안에서 컴파일러 지시 사항을 설정할 때 사용

(주요 사용 예)

#pragma once - 헤더 파일의 중복 include를 예방

(person.h) // pragma 이용 #pragma once 헤더파일 내용 :	(person.h) // ifndef 이용 #ifndef _PERSON_H_ #define _PERSON_H_ 헤더파일 내용 : #endif
---	---

#pragma warning(disable : 4996) - 해당번호(4996)의 경고를 비활성화 시킴

#pragma pack (1) - 프로그램 실행 시 메모리 할당은 1바이트 단위로 할 것(구조체 패딩 없음)

제11장 포 인 터

학습 내용 소개

11장에서는 고급프로그래밍의 초석이 되는 포인터에 대해서 배운다.
차원의 개념과 주소개념을 정확히 이해하고 배열과 포인터와의 관계에 대해서 살펴보고, 포인터를 활용한 유용한 프로그램을 만들어 본다.

학습 목차

1. 포인터의 기본 개념
2. 1차원 포인터
3. 1차원 배열과 포인터
4. 다차원 포인터
5. 포인터배열과 배열을 가리키는 포인터
6. 동적메모리 할당의 이해
7. 함수 포인터
8. void형 포인터
9. const의 이해
10. main()함수의 parameter

1. 포인터의 기본개념

▶ 포인터 변수란?

주소값을 저장하는 변수로서, 포인터 변수는 자신이 저장하고 있는 주소값에 해당하는 영역을 가리키면서 간접적으로 값을 읽어오거나 변경시킬 수도 있다

▶ 상수의 2 분류

- ┌ 일반상수 : 숫자상수, 문자상수, 매크로상수 일부
- └ 주소상수: 문자열상수, 주소상수, 매크로상수 일부

▶ 변수의 2 분류

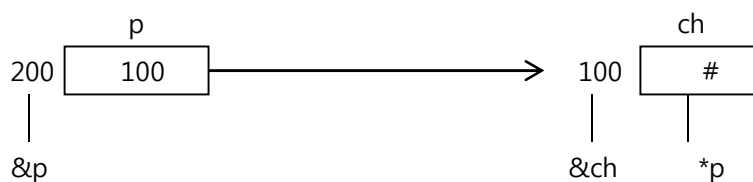
- ┌ 일반변수 : 일반상수 저장
- └ 포인터변수 : 주소상수 저장

▶ 차원조절 연산자 : *, [], &, (type), ->

2. 1차원 포인터

포인터 변수를 선언 할 때는 변수명 앞에 '*'를 붙여 선언한다.

```
예) char *p;    /* 변수 p를 문자형 포인터 변수로 선언한다 */
    char ch = '#';
    p = &ch;    /* 이제 변수 p는 문자변수 ch의 주소를 갖는다 */
```



- p : 포인터 변수 p 자체, 또는 포인터 변수에 들어있는 주소 값 (100번지)
- &p : 포인터 변수 p 자체의 주소 값 (200번지)
- *p : 포인터 변수가 가리키는 대상체 자체(ch), 또는 그 대상체 내의 값('#')
- ch : 문자를 저장하는 문자변수 자체, 또는 변수에 들어있는 값 ('#')
- &ch : 일반변수 ch 자체의 시작주소값 (100번지)

**** 모든 포인터 변수는 동일한 크기(size)를 갖는다**

[예제 11-1] 1차원 포인터 예제

```

1 : #include <stdio.h>
2 : int main()
3 : {
4 :     int *ptr, num = 10 ;
5 :     ptr = &num ;
6 :     *ptr = *ptr+4;
7 :     printf("num = %d\t *ptr = %d \n", num, *ptr) ;
8 :     return 0;
9 : }
```

[예제 11-2] 포인터 변수의 크기 확인하기

```

1 : #include <stdio.h>
2 : int main()
3 : {
4 :     char *cp, ch = 'a' ;
5 :     int *ip, num = 5 ;
6 :     double *dp, dnum = 3.14 ;
7 :     cp = &ch ;
8 :     ip = &num ;
9 :     dp = &dnum ;
10:    printf("cp : %d, *cp : %d Byte\n", sizeof(cp), sizeof(*cp));
11:    printf("ip : %d, *ip : %d Byte\n", sizeof(ip), sizeof(*ip));
12:    printf("dp : %d, *dp : %d Byte\n", sizeof(dp), sizeof(*dp));
13:    return 0;
14: }
```

**** 주요 주소연산 규칙**

1. 주소상수와 정수형 상수와의 연산시에서는 덧셈, 뺄셈이 가능하다.
 이때 (주소상수+n)가 의미하는 값은
 (주소상수 + n * 주소상수가 의미하는 기억공간의 크기) 번지 이다.
2. 주소상수끼리의 연산 시 뺄셈이 가능하며
 이때 두 주소상수는 동일한 차원의 동일한 데이터형 이어야 한다.
 주소끼리의 연산 시에는 그 주소에 해당하는 기억공간 간의 첨자차이가 계산 됨

3. 1차원 배열과 포인터

3.1 1차원 배열과 포인터와의 관계

1차원 배열이 다음과 같이 선언되어 있을 때

```
int *p, *pp;
int ary[5] = {10,20,30,40,50}, num = 10;
p = ary;
pp = &num;    일때
```

※ p가 포인터 변수일 때 p+i가 의미하는 값은 p + i*sizeof(*p)번지 이다.
이 규칙은 p가 배열일때에도 적용된다.

[예제 11-3] 1차원 정수배열과 포인터 예제

```
1 : #include <stdio.h>
2 : int main()
3 : {
4 :     int ary[5] = {10,20,30,40,50} ;
5 :     int *ptr, i ;
6 :     ptr = ary ;
7 :     for(i=0 ; i<5 ; i++) printf("%6d", ary[i]) ;
8 :     printf("\n\n");
9 :     for(i=0 ; i<5 ; i++) printf("%6d", *(ptr+i)) ;
10:    printf("\n\n");
11:    for(i=0 ; i<5 ; i++) printf("%6d", *(ary+i)) ;
12:    printf("\n\n");
13:    for(i=0 ; i<5 ; i++) printf("%6d", ptr[i]) ;
14:    printf("\n\n");
15:    return 0;
16: }
```

[예제 11-4] 포인터가 가리키는 곳은 어디인가?

```
1 : #include <stdio.h>
2 : int main()
3 : {
4 :     int ary[5] = {10,20,30,40,50};
5 :     int *ptr, i;
6 :     ptr = &ary[2];
7 :     for(i=0; i<5; i++) printf("%6d", ary[i]);
8 :     printf("\n\n");
9 :     for(i=-2; i<3; i++) printf("%6d", *(ptr+i));
10:    printf("\n\n");
11:    return 0;
12: }
```

[예제 11-5] char 배열과 char 포인터

```
1 : #include <stdio.h>
2 : #include <string.h>
3 : int main()
4 : {
5 :     char *p = "C program" ;
6 :     char str[20] = "C program";
7 :
8 :     // 문자열 변경 가능 여부를 판단해보세요.
9 :     p[0] = 'B';
10:    str[0] = 'B';
11:
12:    printf("p : %s\n", p);
13:    printf("str : %s\n", str);
14:
15:    strcpy(p, "Java");
16:    strcpy(str,"Java");
17:
18:    printf("p : %s\n", p);
19:    printf("str : %s\n", str);
20:
21:    p = "Python programming";
22:    str = "Python programming";
23:
24:    printf("p : %s\n", p);
25:    printf("str : %s\n", str);
26:
27:    return 0;
28: }
```

3.2 포인터와 연산자

***ptr++**

*ptr의 값을 구하고 ptr을 1 증가시킨다

(*ptr)++

먼저 *ptr의 값을 구한 후, *ptr의 값을 1 증가시킨다

***++ptr**

ptr을 1증가시킨 후, *ptr을 구한다

++*ptr

*ptr의 값을 1증가시킨 후, 그 값을 결과로 취한다

포인터변수는 증감 시 1씩 증감하는 것이 아니라, 그 포인터변수가 가리키는 대상체의 크기만큼 증감한다

ptr ++ => ptr = ptr + 1 = ptr + 1 * sizeof(대상체)

ptr+2 => ptr + 2 * sizeof(대상체)

[예제 11-6] 포인터와 연산자

```

1 :  #include <stdio.h>
2 :  int main()
3 :  {
4 :      int ary[5] = {10,20,30,40,50};
5 :      int *ptr, i;
6 :      ptr = ary;
7 :      printf("1. ptr      %x\n", ptr);
8 :      printf("2. *ptr++   %d\n", *ptr++);
9 :      printf("3. ptr      %x\n", ptr);
10:      printf("4. *++ptr   %d\n", *++ptr);
11:      printf("5. ptr      %x\n", ptr);
12:      printf("6. (*ptr)++ %d\n", (*ptr)++);
13:      printf("7. *ptr     %d\n", *ptr);
14:      printf("8. ++*ptr    %d\n", ++*ptr);
15:
16:      for(i=0; i<sizeof(ary)/sizeof(ary[0]); ++i)
17:      {
18:          printf("ary[%d] = %d\n", i, ary[i]);
19:      }
20:      return 0;
21:  }
```

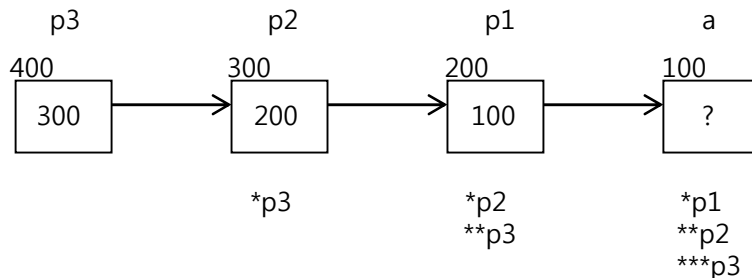
4. 다차원 포인터

4.1 다차원 포인터란?

다차원 포인터란 포인터 기호 '*'를 2개 이상 사용하여 선언한 포인터변수를 의미한다

```
int  a, b ;
int  *p1 ;      => 1차원 int 포인터변수 선언
int  **p2 ;     => 1차원 int 포인터변수를 가리키는 2차원 포인터변수 선언
int  ***p3 ;    => 2차원 int 포인터변수를 가리키는 3차원 포인터변수 선언
p1 = &a ;
p2 = &p1 ;
p3 = &p2 ;
```

위에서 각 변수들의 관계는 다음과 같다



이때 다음의 식들은 동일한 효과의 식이다.

```

a = 7 ;           b = a ;
*p1 = 7 ;         b = *p1 ;
**p2 = 7 ;        b = **p2 ;
***p3 = 7 ;       b = ***p3 ;
```

※ 대입식에서는 항상 Lvalue와 Rvalue의 type과 차원이 맞아야 한다.

일반 변수와 상수값은 0차원, 포인터변수는 선언시 표시된 '*'연산자의 개수가 차원이 된다

4.2 다차원 배열과 포인터의 관계

```
int ary[3][4] = {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}};
```

ary[0] : 2차원 배열 ary 의 0번 부분배열명(0행의 이름, 0행의 주소) 이다

ary[1] : 2차원 배열 ary 의 1번 부분배열명(1행의 이름, 1행의 주소) 이다

ary[2] : 2차원 배열 ary 의 2번 부분배열명(2행의 이름, 2행의 주소) 이다

[다차원 배열의 포인터 표현]

멤버(값) 표현

a[i]	==	*(a + i)
a[i][j]	==	*(*(a + i)+j)
a[i][j][k]	==	*(*(*(a + i)+j)+k)

주소 표현

&a[i]	==	(a + i)
&a[i][j]	==	(*(a + i)+j)
&a[i][j][k]	==	(*(*(a + i)+j)+k)

[예제 11-7] 숫자배열의 포인터 표현

```
1 :  #include <stdio.h>
2 :  int main()
3 :  {
4 :      int a[3][2] = {{1,2}, {3,4}, {5,6}};
5 :      int i, j;
6 :      for(i=0 ; i<3 ; i++)
7 :      {
8 :          printf("\n *(a+%d) : %p\t", i, *(a+i)) ;
9 :          for(j=0 ; j<2 ; j++)
10:         {
11:             printf("%5d", (*(a+i)+j) ) ;
12:         }
13:     }
14:     return 0;
15: }
```

[예제 11-8] 문자배열의 포인터 표현

```
1 : #include <stdio.h>
2 : int main()
3 : {
4 :     char str[3][6] = {"Year" , "Month" , "Day"} ;
5 :     int i ;
6 :
7 :     for(i=0 ; i<3 ; i++)
8 :     {
9 :         printf("%d행의 시작주소 %p이고,%d행의 시작문자는 %c이다\n",
10:                i, *(str+i), i,*(*(str+i)+0));
11:     }
12:     return 0;
13: }
```

5. 포인터배열과 배열을 가리키는 포인터

int **p1; p1은 1차원 int형 포인터를 가리키는 포인터

int *p2[5]; p2는 요소가 5개인 배열로서 각 배열의 요소는 int형 포인터이다.
(포인터 배열)

int (*p3)[5]; p3은 포인터로서 열이 5개인 2차원 배열을 가리키는 포인터 변수
그 배열의 각 요소는 모두 int형이다. (배열 포인터)

[예제 11- 9] 포인터 배열과 배열 포인터

```

1 : #include<stdio.h>
2 : void menuDisplay1(char **menuPtr, size_t menuCnt);
3 : void menuDisplay2(char (*menuPtr)[10], size_t menuCnt);
4 : int main()
5 : {
6 :     char *ptrAryMenu[] = {"insert", "output", "search", "quit"};
7 :     char aryPtrMenu[][10] = {"INSERT", "OUTPUT", "SEARCH", "QUIT"};
8 :
9 :     menuDisplay1(ptrAryMenu, sizeof(ptrAryMenu)/sizeof(ptrAryMenu[0]));
10:    printf("\n\n");
11:    menuDisplay2(aryPtrMenu, sizeof(aryPtrMenu)/sizeof(aryPtrMenu[0]));
12:
13:    return 0;
14: }
15: void menuDisplay1(char **menuPtr, size_t menuCnt)
16: {
17:     for(size_t i=0; i<menuCnt; i++)
18:     {
19:         printf("%d. %s\n", i+1, menuPtr[i]);
20:     }
21:     return;
22: }
23: void menuDisplay2(char (*menuPtr)[10], size_t menuCnt)
24: {
25:     for(size_t i=0; i<menuCnt; i++)
26:     {
27:         printf("%d. %s\n", i+1, menuPtr[i]);
28:     }
29:     return;
30: }

```

[예제 11-10] 포인터 배열과 배열을 가리키는 포인터 예제

```

1 : #include <stdio.h>
2 : int a[] = {0, 1, 2, 3, 4} ;
3 : int *p[] = {a, a+1, a+2, a+3, a+4} ;
4 : int **pp = p ;
5 : int main()
6 : {
7 :     pp++;    printf("%d %d %d\n", pp - p, *pp - a, **pp);
8 :     *pp++;   printf("%d %d %d\n", pp - p, *pp - a, **pp);
9 :     *++pp;   printf("%d %d %d\n", pp - p, *pp - a, **pp);
10:    pp = p ;
11:    **pp++;   printf("%d %d %d\n", pp - p, *pp - a, **pp);
12:    *++*pp ;  printf("%d %d %d\n", pp - p, *pp - a, **pp);
13:    ++**pp ;  printf("%d %d %d\n", pp - p, *pp - a, **pp);
14:    return 0;
15: }

```

6. 동적메모리 할당의 이해 및 활용

▶ 동적메모리 할당이란?

프로그래밍 시에 정해진 크기의 기억공간을 미리 선언하여 놓는 것을 정적메모리 할당이라하며, 프로그램 수행도중에 필요한 메모리를 할당 받아 사용하는 것을 동적메모리 할당이라 한다.

- malloc()함수로 메모리를 할당 받고 사용이 끝난 메모리는 free()함수로 해제 해준다.

(두 함수는 malloc.h를 include하여 사용 함)

▶ 변수 1개의 동적메모리 할당

```
double *dp;
dp=(double *)malloc( sizeof(double)); // 동적메모리 할당
*dp = 3.5; // 동적메모리 할당한 기억공간에 3.5 저장
printf("dp = %.2lf\n", *dp);
scanf("%lf", dp); // 동적메모리 할당한 기억공간에 실수값 입력 받음
printf("dp = %.2lf\n", *dp);
free(dp); // 동적메모리 할당한 기억공간을 반납한다.(미예약 영역으로 되돌림)
```

▶ 1차원 배열의 동적메모리 할당

[예제 11-11] 1차원 배열의 동적 메모리 할당

```
#include<stdio.h>
#include<malloc.h>
int * myAlloc(int);
void dataInput(int *, int);
void dataOutput(int *, int);
void myDelete(int *);
int main()
{
    int *ip;
    int col;
    printf("column 수 입력 : ");
    scanf("%d", &col);
    ip = myAlloc(col);
    dataInput(ip, col);
    dataOutput(ip, col);
    myDelete(ip);

    return 0;
}

// 4개의 사용자 정의 함수 구현하기
```

▶ 2차원 배열의 동적메모리 할당

[예제 11-12] 2차원 배열의 동적 메모리 할당

```
#include<stdio.h>
#include<malloc.h>
int ** myAlloc(int, int);
void dataInput(int **, int, int);
void dataOutput(int **, int, int);
void myDelete(int **, int);

int main()
{
    int **ip;
    int col, row;

    printf("row 수 입력 : ");
    scanf("%d", &row);
    printf("column 수 입력 : ");
    scanf("%d", &col);

    ip = myAlloc(row, col);
    dataInput(ip, row, col);
    dataOutput(ip, row, col);
    myDelete(ip, row);
    return 0;
}
```

// 4개의 사용자 정의 함수 구현하기

7. 함수 포인터

▶ 함수포인터의 의미와 선언 형식

함수포인터란 함수를 가리키는 포인터를 말한다

(선언형식)

함수의 type (*함수포인터명) (함수전달인자);

[예제 11-13] 함수 포인터 예제

```

1 :   #include <stdio.h>
2 :   int  plus(int, int) ;
3 :   int main()
4 :   {
5 :       int  a =10, b = 20 ;
6 :       int  res ;
7 :       int  (*fptr) (int,int); // plus() 함수를 가리키는 포인터변수 선언
8 :       fptr = plus ;           // 함수포인터로 함수 가리키기
9 :       res = fptr(a, b) ;      // 함수호출부 : res = (* fptr)(a, b) ;
10:      printf("%d + %d = %d\n", a, b, res) ;
11:      return 0;
12:  }
13:  int  plus(int x, int y)
14:  {
15:      return (x + y) ;
16:  }

```

[예제 11-14] 함수 포인터 배열 예제

```

1 : #include<stdio.h>
2 : int plus(int, int);
3 : int minus(int, int);
4 : int multi(int, int);
5 :
6 : int main()
7 : {
8 :     int i,j, res;
9 :     int (*fpAry[4])(int, int) = {plus, minus, multi, plus};
10:
11:     for(i = 0; i<sizeof(fpAry)/sizeof(fpAry[0]); i++)
12:     {
13:         res=fpAry[i](10,20);
14:         printf("res = %d\n", res);
15:     }
16:     return 0;
17: } // end of main()
18: int plus(int a, int b)
19: {
20:     return a+b;
21: }
22: int minus(int a, int b)
23: {
24:     return a-b;
25: }
26: int multi(int a, int b)
27: {
28:     return a*b;
29: }

```

int (*fpAry[2][2])(int, int) = {{plus, minus}, {multi, plus}}; 로 선언하면?

8. void형 포인터

▶ void 포인터의 의미와 선언 형식

void 포인터는 가리키는 대상체의 type에 제한이 없는 포인터를 말한다

(선언형식)

void * 포인터명;

void *p;

int intArray[5] = {1,2,3,4,5}

double doubleArray[2][3] = { {1.5, 2.5, 3.5}, {4.5, 5.5, 6.5} };

const char *charPrtArray[4] = {"apple", "pear", "banana", "pineapple"};

p = intArray; // void형 포인터 p는 int형 1차원 배열을 가리킬 수 있고

printf("%d\\n", *(int *p+2));

p = doubleArray; // double형 2차원 배열을 가리킬 수도 있으며

printf("%lf\\n", *((double (*)[3])p+1)+2);

p = charPrtArray; // const char * 배열을 가리킬 수도 있다

printf("%s\\n", *((const char **)p+2));

[예제 11-15] 함수 포인터 & void 포인터 예제

```

1 :   #pragma warning (disable:4996)
2 :   #include<stdio.h>
3 :   #include<stdlib.h>
4 :   #include<memory.h>
5 :   void outputArray(void *, size_t, size_t, void (*printData)(void *));
6 :   void printInt(void *);
7 :   void printDouble(void *);
8 :
9 :   int main()
10:  {
11:      int iAry[5] = { 10,20,30,40,50 };
12:      double dAry[3] = { 3.5, 7.5, 8.5 };
13:      printf("\n[ iAry 배열 출력 ]\n");
14:      outputArray(iAry, sizeof(iAry)/sizeof(iAry[0]), sizeof(int), printInt);
15:      printf("\n[ dAry 배열 출력 ]\n");
16:      outputArray(dAry, sizeof(dAry) / sizeof(dAry[0]), sizeof(double),
17:                  printDouble);
18:      return 0;
19:  }
20:  //-----
21:  void outputArray(void *start, size_t count, size_t dataSize,
22:                  void(*printData)(void *))
23:  {
24:      int i;
25:      for(i=0; i< count; i++) {
26:          printf("%d. ", i+1);
27:          printData( (char *)start + i * dataSize );
28:      }
29:      return;
30:  }
31:
32:  //-----
33:  void printInt(void *p) {
34:      printf("%5d\n", *(int *)p);
35:  }
36:  //-----
37:  void printDouble(void *p) {
38:
39:      printf("%7.2lf\n", *(double *)p);
40:
41:  }

```

9. const의 이해

▶ const의 의미 및 사용목적

변수선언 시 데이터형 앞에 const라는 명령어를 붙여서 변수에게 상수의 성격을 부여한다.
이때 const로 선언된 변수는 상수처럼 사용된다. (읽기 전용변수)

<특징>

- const변수는 반드시 선언문에서만 초기화가 가능하다
- 초기화 이후에는 실행문에서 그 값을 변경할 수 없다

▶ 변수가 const인 경우

```
const double pi=3.14;
double b;
printf("%lf\n", pi);      (   )
printf("%lf\n", pi +1);   (   )
printf("%lf\n", pi ++);   (   )
b = pi;                   (   )
pi = 3.141592;            (   )
```

▶ 포인터와 const

포인터에 const를 사용할 경우에는 포인터의 대상이 상수화되는 경우와 포인터변수 자체가 상수화되는 경우가 있다

- 포인터의 대상이 const화 되는 경우

```
const int *p;
int abc = 7;
p = &abc;
*p -= 10;    (   )
abc -= 10;   (   )
```

- 포인터 변수가 const화 되는 경우

```
int abc[3] = {10,20,30};
int * const p = abc;
printf("%d\n", *p);
printf("%d\n", ++*p);    (   )
printf("%d\n", *++p);    (   )
```

▶ 포인터와 const의 활용

```
const char *cp="kiwi";
```

```
char * p= cp;          (불가능) - 이유는?
```

[예제 11-16] const와 포인터의 활용

```
1 : #include<stdio.h>
2 : const char *sub(const char *const);
3 : int main()
4 : {
5 :
6 :     char str[100]= "apple pie";
7 :     const char *resp;
8 :
9 :     resp = sub(str);
10:    printf("%s\n", resp);
11:
12:    return 0;
13: }
14: const char *sub(const char *const p)
15: {
16:     printf("%s\n", p);
17:     printf("%s\n", p+6);
18:     //printf("%s\n", ++p);
19:     return p;
20: }
```

제12장 구조체 (Structure)

학습 내용 소개

12장에서는 매우 유용한 사용자 정의 데이터형인 구조체에 대해서 배운다.
구조체는 다양한 데이터형을 간결하게 사용하게 해주는 매우 강력한 도구이다.
구조체의 특성과 활용에 대해 자세히 공부하며, bit field와 공용체, 나열형의 활용에 대해서도 살펴본다.

학습 목차

1. 구조체의 기본개념
2. 구조체 배열과 구조체 포인터
3. typedef를 이용한 다양한 타입 재정의 및 활용 기법
4. 구조체를 함수로 보내기
5. 구조체 Padding
6. Flexible array member
7. Bit Field의 기본개념 및 특징
8. 공용체(union) 및 나열형(enum)

1. 구조체의 기본개념

▶ 구조체란?

구조체(Structure)란 서로 다른 데이터형의 정보를 집합시킨 것으로 복잡한 데이터를 간결하게 단 일변수처럼 처리할 수 있도록 해주는 복합 데이터형이다.

구조체는 서로 연관관계가 있는 1개에서 여러 개의 member로 구성한다.

▶ 구조체의 기본 구조

```
struct 구조체명
{
    member-1의 선언문 ;
    member-2의 선언문 ;
    :
} 구조체 변수명1, 구조체 변수명2;

int main()
{
    struct 구조체명 구조체변수명-1, 구조체변수명-2, ... ;
}
```

▶ 구조체의 성질

1. 구조체형명은 식별자 만드는 규칙에 의거하여 사용자가 정의한다.
2. 구조체 형틀 선언 시에는 메모리는 할당 되지 않고,
구조체 변수를 선언해야만 실제로 메모리가 할당된다.
그러므로 구조체형틀 선언 시 멤버를 초기화하는 것은 불가능하다.
3. 구조체 member 로 선언 가능한 것으로는 변수, 배열, 또 다른 구조체와 같은 복합 데이터형 변수 등이다.
4. 구조체의 각 멤버에 접근할 때에는 멤버참조연산자(.)를 사용한다.

[구조체변수명. 구조체멤버명]

▶ 구조체 Member 의 초기화 및 Member 참조

[예제 12-1] 구조체 멤버 초기화

```

1 :    #pragma warning (disable : 4996)
2 :    #include<stdio.h>
3 :    struct Person
4 :    {
5 :        char name[20];
6 :        char addr[40];
7 :        int age;
8 :    };
9 :    struct Student
10:    {
11:        int number;
12:        struct Person info;
13:    };
14:    int main()
15:    {
16:        struct Student s1 = { 1, {"홍길동", "서울", 21} };
17:        struct Student s2 = { 2, {"이순신", "울릉도", 25} };
18:
19:        printf("학번: %d, 성명: %s, 주소: %s, 나이: %d\n", s1.number,
20:              s1.info.name, s1.info.addr, s1.info.age);
21:        printf("학번: %d, 성명: %s, 주소: %s, 나이: %d\n", s2.number,
22:              s2.info.name, s2.info.addr, s2.info.age);
23:        return 0;
24:    }

```

2. 구조체 배열과 구조체 포인터

▶ 구조체 배열의 사용

```

struct Person
{
    char name[20];
    char addr[40];
    int age;
};
int main()
{
    struct Person a[3]={ {"이순신", "울릉도", 23}, {"홍길동", "서울", 21},
                        {"강감찬", "부산", 22} };
    strcpy(a[0].name, "신사임당");
    a[0].age = 30;
    :
}

```

▶ 구조체 포인터의 사용

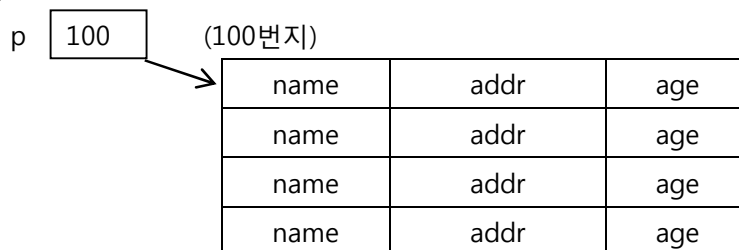
구조체 포인터를 이용하여 멤버에 접근할 때에는 간접멤버참조 연산자(->)를 이용함

[구조체 포인터 변수명->구조체 멤버명]

```
struct Person
{
    char name[20];
    char addr[40];
    int age;
};

int main() {
    struct Person a[4] = { {"이순신", "울릉도", 23}, {"홍길동", "서울", 21},
                           {"강감찬", "부산", 22}, {"신사임당", "인천", 27} };

    struct Person *p ;
    p = a ;
    :
}
```



```
a[0].name == (p+0)->name == p->name
a[3].age  == (p+3)->age
++p->age == ++(p->age) : p의 멤버 age의 값을 1증가
(++p)->age + 50 : 다음 번 구조체의 age항목을 가리킨다
p->age       : p가 가리키고 있는 구조체의age항목
p++->age     : p->age을 구하고 p를 1증가
p->age++     : p->age의 값을 사용 후, 그 값을 1 증가 시킨다
--p->age + 50 : p->age의 값을 1감소 시킨 후 50을 더한다
```

3. typedef를 이용한 다양한 타입 재정의 및 활용 기법

▶ typedef 명령어 : 형명 재지정 명령어

typedef는 기존에 사용하는 형명에 새로운 형명을 부여하는 명령어임
 새형명을 사용할 때의 장점 : 형명의 간결화, 형명에 의미부여 가능

- 형식1 : typedef 구형명 새형명;

```
typedef unsigned int  UI;
typedef unsigned int  size_t;
typedef int * IP;

typedef struct _person Person;
typedef struct _person
{
    char  name[10] ;
    char  addr[40] ;
    int   age ;
} Person;
```

- 형식2 : typedef return type (*함수포인터의 새형명) (parameter list);

```
void sub(int, int);
typedef void (*FuncPtr) (int, int);
FuncPtr fPtr;
fPtr = sub;
fPtr(10,20);
```

- 형식3 : typedef type (*포인터의 새형명) [n];

```
char str[3][10];
typedef char (*strPtr_t)[10]
strPtr_t p;
p = str;
p[1][2] = 'b';
```


▶ 함수 포인터의 타입 재정의

[예제 12-2] 함수포인터 타입의 재정의

```

1 :    #include<stdio.h>
2 :    void sub1(void);
3 :    void sub2(void);
4 :    void sub3(void);
5 :
6 :    int main()
7 :    {
8 :        typedef void (*func_t)(void); /* 함수포인터 변수의 타입 재정의 */
9 :        /* typedef void (*)(void) func_t; error 발생하는 타입 재정의 표현 */
10:       /* typedef 키워드로 재정의한 자료형은 관례적으로 _t 접미사를 갖는다 */
11:       int i;
12:       func_t *fptr; /* 함수포인터변수를 가리키는 포인터 선언 */
13:       func_t funAry[3] = {sub1, sub2, sub3};
14:       fptr=funAry;
15:       for(i=0; i<3; i++)
16:       {
17:           fptr[i]();
18:           funAry[i]();
19:           printf("\n");
20:       }
21:
22:       return 0;
23:   }
24:   void sub1(void)
25:   {
26:       printf("나는 sub1() 함수~\n");
27:   }
28:   void sub2(void)
29:   {
30:       printf("나는 sub2() 함수~\n");
31:   }
32:   void sub3(void)
33:   {
34:       printf("나는 sub3() 함수~\n");
35:   }

```

▶ 반환값의 재정의(함수포인터를 리턴하는 함수의 반환값 재정의)

[예제 12-3] 재정의된 반환 값

```

1 : #include<stdio.h>
2 : typedef void (*func_t)(void); /* 함수포인터의 형명재지정 */
3 : void vegetable(), fruit(), grain();
4 : func_t getGroup(int); /* 함수포인터를 리턴하는 함수 선언부 */
5 : int main()
6 : {
7 :     int menu;
8 :     void (*fptr)(void) = NULL;
9 :     while(1)
10 :    {
11 :        printf("1. vegetable/ 2. fruit / 3. grain / 4. quit : ");
12 :        scanf("%d", &menu);
13 :        if(menu==4){ break; }
14 :        fptr = getGroup(menu); /* menu에 따른 함수포인터값 리턴 받기 */
15 :        if(fptr != NULL) { fptr(); }
16 :    }
17 :    return 0;
18 : }
19 : func_t getGroup(int menu) /* 메뉴번호에 해당하는 함수의 시작주소를 리턴하는 함수*/
20 : /* void (*)(void) getGroup(int menu) 이런 형태의 리턴은 에러 발생 */
21 : /* void (*getGroup(int menu))(void) 이 표현은 가능 */
22 : {
23 :     switch(menu)
24 :     {
25 :         case 1 : return vegetable;
26 :         case 2 : return fruit;
27 :         case 3 : return grain;
28 :     }
29 :     return NULL;
30 : }
31 : void vegetable()
32 : {
33 :     char *vegetableName[] = {"celery", "cucumber", "potato"};
34 :     for(size_t i=0; i< sizeof(vegetableName)/sizeof(vegetableName[0]); i++)
35 :     {
36 :         printf("[%d] %s\n", i+1, vegetableName[i]);
37 :     }
38 : }
39 : void fruit()
40 : {
41 :     char *fruitName[] = {"grape", "kiwi", "strawberry", "banana", "orange"};
42 :     for(size_t i=0; i< sizeof(fruitName)/sizeof(fruitName[0]); i++)
43 :     {
44 :         printf("[%d] %s\n", i+1, fruitName[i]);
45 :     }
46 : }
47 : void grain()
48 : {
49 :     char *grainName[] = {"rice", "bean", "oats", "corn"};
50 :     for(size_t i=0; i< sizeof(grainName)/sizeof(grainName[0]); i++)
51 :     {
52 :         printf("[%d] %s\n", i+1, grainName[i]);
53 :     }
54 : }

```

▶ 반환값의 재정의2 (다차원 배열포인터를 리턴하는 함수의 반환값 재정의)

[예제 12-4] 다차원 배열포인터를 리턴하는 함수의 반환값 재정의

```

1 :   #include<stdio.h>
2 :   #include<malloc.h>
3 :   typedef int (*intArrayPtr)[4]; /* int [][][4] 배열을 가리키는 포인터변수의 형명
4 :   재지정 */
5 :   intArrayPtr  memoryAllocation(int);
6 :
7 :   int main()
8 :   {
9 :       int row;
10:      int(*aryPtr)[4] = NULL;
11:
12:      printf("행(row)의 수 입력 : ");
13:      scanf("%d", &row);
14:
15:      aryPtr = memoryAllocation(row); /* menu에 따른 함수포인터값 리턴 받기 */
16:      aryPtr[row-1][3] = 77;
17:      printf("aryPtr[%d][3] = %d", row-1, aryPtr[row-1][3]);
18:
19:      free(aryPtr);
20:
21:      getchar(); getchar();
22:      return 0;
23:  }
24:  intArrayPtr memoryAllocation(int row) /* 전달인자의 행수만큼 2차원 배열메모리를
25:  할당하여 주소를 리턴하는 함수*/
26:  /* int (*)[4] memoryAllocation(int row) 는 에러나는 표현 */
27:  /* int (*memoryAllocation(int row))[4] 는 가능한 표현 */
28:  {
29:      intArrayPtr p; /*동일표현 : int (*p)[4]=NULL;*/
30:      p=(intArrayPtr) malloc(row*sizeof(int[4])); /* 동일표현 : p = (int
31:  (*)[4])malloc(row*sizeof(int [4])); */
32:
33:      return p;
34:  }

```

[예제 12-5] 구조체 배열과 포인터

```

1 :   #include <stdio.h>
2 :   #define CNT 3
3 :   typedef struct person
4 :   {
5 :       char name[10] ;
6 :       char addr[40] ;
7 :       int age ;
8 :   } Person;
9 :   int main()
10:  {
11:      Person x[CNT] = {{“Kim”,“Seoul”,20},{“Lee”,“Pusan”,30},{“Park”,“Daegu”,25}};
12:      Person *ps ;
13:      int i ;
14:      ps = x ;
15:      for(i=0 ; i<CNT ; i++)
16:      {
17:          printf(“%s(%d) : %s\n”,(ps+i)->name,(ps+i)->age, (ps+i)->addr ) ;
18:      }
19:      return 0;
20:  }

```

4. 구조체를 함수로 보내기

- ▶ 구조체를 함수로 전달하는 방법은 3가지가 있다

전달 방법	피호출(호출된) 함수에서 받는 모양
각 멤버의 전달	각 멤버에 해당하는 type의 변수로 받음
변수의 전달	같은 type의 구조체 변수로 받음
변수의 주소, 배열의 전달	같은 type의 구조체 포인터변수로 받음

[예제 12-6] 구조체 각 멤버의 전달

```

1 :   #include<stdio.h>
2 :   typedef struct _score
3 :   {
4 :       char name[20] ;
5 :       int kor, eng, mat ;
6 :   }Score;
7 :   void struct_prn (char *, int, int, int) ;
8 :   int main()
9 :   {
10:      Score sd = {“Hong gil dong” , 80, 90, 100 } ;
11:      struct_prn (sd.name, sd.kor, sd.eng, sd.mat) ;
12:      return 0;
13:  }
14:  /*-----*/
15:  void struct_prn (char *cp, int k, int e, int m)
16:  {
17:      printf(“%s : %d %d %d \n” , cp, k, e, m ) ;
18:  }

```

[예제 12-7] 구조체 변수 전달

```

1 :  #include <stdio.h>
2 :  struct _score
3 :  {
4 :      char  name[20] ;
5 :      int   kor, eng, mat ;
6 :  } ;
7 :  typedef struct _score Score;
8 :  void struct_prn (Score) ;
9 :
10:  int main()
11:  {
12:      Score sd = {"Hong gil dong" , 80, 90, 100 } ;
13:      struct_prn (sd) ;
14:      return 0;
15:  }
16:  /*----- */
17:  void struct_prn (Score x)
18:  {
19:      printf("%s : %d %d %d \n", x. name, x.kor, x.eng, x.mat);
20:  }

```

[예제 12-8] 구조체 변수의 시작주소 전달

```

1 :  #include <stdio.h>
2 :  typedef struct _score
3 :  {
4 :      char  name[20] ;
5 :      int   kor, eng, mat ;
6 :  }Score ;
7 :  void struct_prn(Score *) ;
8 :  int main()
9 :  {
10:      Score sd = {"Hong gil dong", 80, 90, 100 };
11:      struct_prn(&sd) ;
12:      return 0;
13:  }
14:  /*----- */
15:  void struct_prn (Score *p)
16:  {
17:      printf("%s : %d %d %d\n", p->name, p->kor, p->eng, p->mat ) ;
18:  }

```

[예제 12-9] 구조체 배열의 전달

```

1 :   #include <stdio.h>
2 :   typedef struct
3 :   {
4 :       char  name[20] ;
5 :       int   kor, eng, mat ;
6 :   }Score ;
7 :   void struct_prn(Score *);
8 :   int  main()
9 :   {
10:      Score ary[2] = { {"Hong gil dong" , 80, 90, 100 } ,
11:                      {"Lee mong yong" , 75, 85, 95 } } ;
12:
13:      struct_prn(ary) ;
14:      return 0;
15:  }
16:  /*----- */
17:  void struct_prn (Score *p)
18:  {
19:      int i ;
20:      for(i=0; i<2; i++)
21:      {
22:          printf("%s : %d %d %d \n", p[i].name, p[i].kor,
23:                (p+i)->eng, (p+i)->mat ) ;
24:      }
25:  }

```

▶ 자기참조 구조체란?

구조체 멤버 중에 자기 자신을 가리키는 포인터를 포함하고 있는 구조체
주로 링크드 리스트를 구성할 때 사용된다.

* 다음의 각 구조체에 대해서 생각해보자

<pre> struct list { int i; struct list *ptr; }; </pre>	<pre> struct list { int i; struct list ptr; }; </pre>	<pre> struct person { char name[10]; int age; }; struct list { int i; struct person *ptr; }; </pre>
자기참조 구조체		

5. 구조체 Padding

▶ 구조체 padding 이란?

- 구조체의 멤버들이 메모리에 할당될 때 CPU의 성능 향상을 위한 align규정 상 멤버 사이에 사용하지 않는 공간이 존재하는데 이것을 padding이라 한다.
- 구조체 align은 기본 자료형 중에서 가장 큰 타입 기준으로 설정된다.
- align 크기는 일반적으로 2,4,8 Byte 크기로 이루어 짐 (단, 컴파일러에 따라 다를 수 있음)

[예제 12-10] 구조체 padding 예제1

```

1 : #include<stdio.h>
2 : struct Data1{
3 :     char c;
4 :     short h;
5 :     int i;
6 :     double d;
7 : };
8 : struct Data2{
9 :     char c;
10:    double d;
11:    short h;
12:    int i;
13: };
14: struct Data3{
15:    char c;
16:    short h;
17:    double d;
18:    int i;
19: };
20: int main()
21: {
22:     struct Data1 v1;
23:     struct Data2 v2;
24:     struct Data3 v3;
25:
26:     printf(" &v1.c : %u\n &v1.h : %u\n &v1.i : %u\n &v1.d : %u\n",
27:           &v1.c, &v1.h, &v1.i, &v1.d);
28:     printf("struct Data1 size : %u\n", sizeof(struct Data1));
29:     printf("----- \n\n");
30:
31:     printf(" &v2.c : %u\n &v2.d : %u\n &v2.h : %u\n &v2.i : %u\n",
32:           &v2.c, &v2.d, &v2.h, &v2.i);
33:     printf("struct Data2 size : %u\n", sizeof(struct Data2));
34:
35:     printf("----- \n\n");
36:
37:     printf(" &v3.c : %u\n &v3.h : %u\n &v3.d : %u\n &v3.i : %u\n",
38:           &v3.c, &v3.h, &v3.d, &v3.i);
39:     printf("struct Data3 size : %u\n", sizeof(struct Data3));
40:     return 0;
41: }

```

```

&v1.c : 3406460
&v1.h : 3406462
&v1.i : 3406464
&v1.d : 3406468
Data1 size : 16

```

```

-----
-
&v2.c : 3406428
&v2.d : 3406436
&v2.h : 3406444
&v2.i : 3406448
Data2 size : 24

```

```

-----
&v3.c : 3406396
&v3.h : 3406398
&v3.d : 3406404
&v3.i : 3406412
Data3 size : 24

```

[예제 12-11] 구조체 padding 예제2

```

1 :    #include<stdio.h>
2 :    struct Arr1{
3 :        char c[9]; // 10 Byte
4 :        short h;    // 2 Byte
5 :        int i;      // 4 Byte
6 :        double d;   // 8 Byte
7 :    };
8 :    struct Arr2{
9 :        char c[11]; // 12 Byte
10:        short h;     // 4 Byte
11:        int i;       // 8 Byte
12:        double d;    // 8 Byte
13:    };
14:    int main()
15:    {
16:        struct Arr1 a1;
17:        struct Arr2 a2;
18:
19:        printf(" &a1.c : %u\n &a1.h : %u\n &a1.i : %u\n &a1.d : %u\n",
20:            &a1.c, &a1.h, &a1.i, &a1.d);
21:        printf("Data1 size : %u\n", sizeof(struct Arr1));
22:        printf("----- \n\n");
23:
24:        printf(" &a2.c : %u\n &a2.h : %u\n &a2.i : %u\n &a2.d : %u\n",
25:            &a2.c, &a2.h, &a2.i, &a2.d);
26:        printf("Data1 size : %u\n", sizeof(struct Arr2));
27:        printf("----- \n\n");
28:        return 0;
29:    }

```

```

&a1.c : 1898796
&a1.h : 1898806
&a1.i : 1898808
&a1.d : 1898812
Data1 size : 24
-----
&a2.c : 1898756
&a2.h : 1898768
&a2.i : 1898772
&a2.d : 1898780
Data1 size : 32
-----

```


6. Bit Field의 기본개념 및 특징

▶ bit filed란?

비트 필드는 bit 단위로 멤버를 할당해서 사용하는 구조체의 변형된 복합 데이터형이다.

비트 필드는 구조체 정의를 사용하여 만들 수 있으며, 필드 멤버 선언 시 그 폭을 지정해준다.

비트필드에 값을 대입할 때에는 해당 필드 멤버가 저장 할 수 있는 값을 초과하지 않도록 주의 해야 한다.

```
struct small
{
    unsigned int  a : 2 ;
    unsigned int  b : 2 ;
    unsigned int  c : 3 ;
    unsigned int  d : 4 ;
}prcode ;
```

prcode는 4개의 멤버로 총 11비트로 구성되어 있으며, 다음과 같은 대입문을 사용할 수 있다. (멤버 참조 연산자 사용)

```
prcode.a = 0 ;
prcode.b = 3 ;
prcode.c = 7 ;
```

단, 대입하는 값은 해당 필드가 저장 할 수 있는 값을 초과하지 않도록 주의 해야 한다.

[예제 12-12] 비트필드 예제

```
1 :  #include <stdio.h>
2 :  struct bitsample
3 :  {
4 :      unsigned a : 2 ;
5 :      unsigned b : 3 ;
6 :      int      c : 4 ;
7 :      int      d : 10 ;
8 :  } ;
9 :  int main()
10:  {
11:      struct bitsample bx ;
12:      int a, b, c, d ;
13:      scanf("%u %u", &a, &b ) ;
14:      scanf("%d %d",&c,&d ) ;
15:      bx.a = a ;
16:      bx.b = b ;
17:      bx.c = c ;
18:      bx.d = d ;
19:      printf("%u %u %d %d\n" , bx.a, bx.b, bx.c, bx.d ) ;
20:      return 0;
21:  }
```

7. 공용체(union) 및 나열형(enum)

▶ 공용체란?

구조체는 각 구성원(멤버)을 연속적인 메모리에 모두 할당시키는데 반해 공용체(union)는 각 구성원(멤버)을 한 공간에 할당하여 선택적으로 사용하게 한다. 즉, 공용체에서 모든 멤버는 같은 시작번지를 갖으며 가장 큰 멤버의 크기만큼만 기억장소를 할당 받는다.

▶ 공용체의 기본구조

```
union  공용체명
{
    member-1의 선언문 ;
    member-2의 선언문 ;
    :
}공용체변수명1, 공용체변수명2 ;

int main()
{
    union  공용체명  공용체변수명3, 공용체변수명4 ;
}
```

[예제 12-13] 공용체 예제

```
1 :  #include <stdio.h>
2 :  union number
3 :  {
4 :      char  a ;
5 :      short b ;
6 :      long  c ;
7 :  } abc ;
8 :  int main()
9 :  {
10:      abc.c = (long) 0x12345678 ;
11:      printf("Long value : %08lx\n" , abc.c ) ;
12:      printf("short value : %08hx\n" , abc.b ) ;
13:      printf("Char value : %08x\n" , abc.a ) ;
14:      return 0;
15:  }
```

▶ Endian mode란?

컴퓨터 내부에 데이터가 저장될 때 인텔계열과 모토로라 계열의 데이터 저장 방식이 다르다.
이를 구분하기 위해서 "엔디안(Endian) 모드"라는 용어를 사용 함

▶ Endian mode의 종류

- Big endian : 메모리 주소의 정순으로 저장 (모토로라 계열)
- Little endian : 메모리 주소의 역순으로 저장 (인텔계열)

[예제 12-14] 구조체와 공용체와의 혼용

```

1 :    #include <stdio.h>
2 :    struct str
3 :    {
4 :        char  c1,c2,c3,c4 ;
5 :    } ;
6 :    union abc
7 :    {
8 :        struct str chr ;
9 :        char  a ;
10:        short b ;
11:        long  c ;
12:    } ;
13:    int  main()
14:    {
15:        union abc value ;
16:        value.c = (long)0x12345678 ;
17:        printf("a = %08x\n" , value.a ) ;
18:        printf("b = %08hx\n" , value.b ) ;
19:        printf("c = %08lx\n" , value.c ) ;
20:        printf("c1 = %08x\n" , value.chr.c1 ) ;
21:        printf("c2 = %08x\n" , value.chr.c2 ) ;
22:        printf("c3 = %08x\n" , value.chr.c3 ) ;
23:        printf("c4 = %08x\n" , value.chr.c4 ) ;
24:        return 0;
25:    }

```

▶ 나열형(열거형 : enumerated type)이란?

나열형을 이용하면 정수상수를 기호 이름으로 선언해서 사용하거나 정수기호상수를 저장하는 변수를 선언하여 사용할 수 있다.

정수기호상수를 사용하면 프로그램의 가독성을 높일 수 있는 장점이 있다.

▶ 나열형의 기본구조

```
enum 나열형명
{
    기호상수1,    ◀ 기호상수값은 지정되지 않으면 default로 0부터 1씩 증가한 값으로 매겨짐
    기호상수2,
    :
}나열형 변수명1;

int main()
{
    enum 나열형명 나열형변수명2 ;
}
```

[예제 12-15] 나열형의 간단한 예

```
1 :    #include<stdio.h>
2 :    enum Nation
3 :    {
4 :        KOREA,
5 :        USA,
6 :        JAPAN,
7 :        CHINA
8 :    };
9 :    int main()
10 :    {
11 :        // 나열형 변수를 선언하여 사용할 때의 특징
12 :        enum Nation x; //Nation 타입의 x변수 선언을 하면 x변수에는
13 :                        // 나열형 선언 시 지정한 상수값 만 저장가능 함
14 :        x = KOREA;
15 :        printf("x=%d\n", x);
16 :        //x = 0; <=== 에러발생
17 :
18 :        // 매크로 상수 대응으로 사용하는 나열형
19 :        printf("%d\n", KOREA); // 0 출력 #define KOREA (0) 한 효과
20 :        printf("%d\n", USA);   // 1 출력 #define USA (1) 한 효과
21 :        printf("%d\n", JAPAN); // 2 출력 #define JAPAN (2) 한 효과
22 :        printf("%d\n", CHINA); // 3 출력 #define CHINA (3) 한 효과
23 :        return 0;
24 :    } // end of main()
```

** KOREA = 3 으로 수행하면?

제13장 파일 입출력

학습 내용 소개

13장에서는 파일입출력을 통한 파일제어 기법을 배운다.

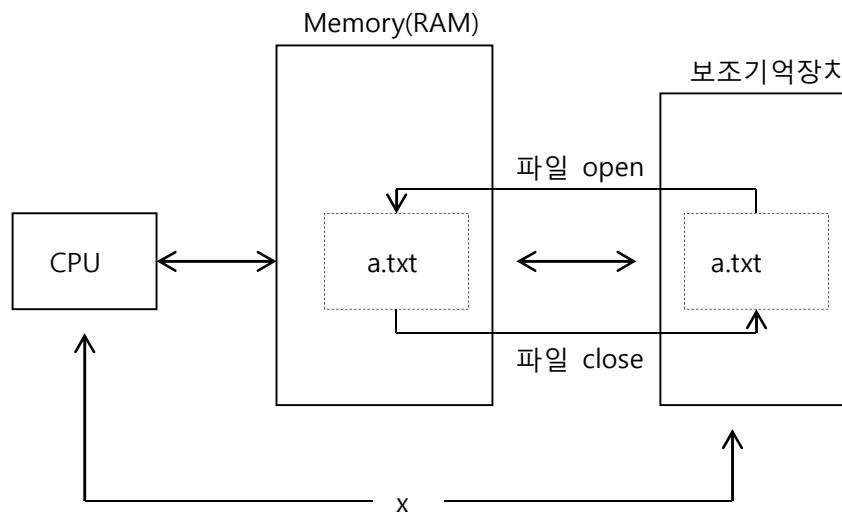
파일 입출력의 기본개념을 이해하고 프로그램의 결과물을 파일에 저장하거나 파일로 제공된 데이터를 읽어서 프로그램에서 사용하는 기법을 공부한다.

학습 목차

1. 파일 입출력의 기본개념
2. 고수준 입출력 함수
3. 고수준화일 I/O와 저수준화일 I/O의 차이
4. 저수준 입출력 함수

1. 파일 입출력의 기본개념

1.1 파일 I/O란?



[그림 1]

- 기본적으로 CPU는 보조기억장치내의 파일을 직접 읽어 들이는 것이 불가능하다. 그러므로 보조기억장치내의 내용을 I/O하고자 할 경우에는 일단 RAM안에 사용하고자 하는 파일을 올려놓아야 한다.
- 이렇게 보조기억장치내의 파일을 RAM에 올려 놓는 작업을 파일 open이라 하며 RAM안의 open되어 있는 파일을 사용 후 다시 보조기억장치로 저장하는 작업을 파일 close라 한다.
- 파일 I/O(Input/Output)란 보조기억장치에 들어있는 데이터 파일의 내용을 읽어 오거나 저장하는 것을 의미한다.

1.2 C에서의 파일 입출력의 구체적인 의미

[그림1 에서 파일 open 시 수행되는 작업의 내용]

- 보조기억 장치내의 파일의 일부가 RAM에 복사된다.(file buffer size)
- 파일구조체가 생성된다.
- OS로부터 file handle번호를 부여 받는다.
- 파일에 대한 모든 정보를 파일구조체에 저장한다.

[그림1에서 파일 close 시 수행되는 작업의 내용]

- RAM안의 파일을 보조기억장치로 복사한다.
- OS에게 file handle번호를 반환한다.
- 파일을 메모리 내에서 삭제한다.
- 파일구조체를 메모리 내에서 삭제한다.

[파일핸들 번호란?]

- 메모리(RAM)안에 open되어진 파일에 붙는 고유한 파일의 번호.
 - 이 번호는 운영체제(OS)가 관리하여 준다.
 - 파일핸들 번호는 0~19번까지 있는데 이는 C에서는 동시에 open할 수 있는 개수가 20개임을 나타낸다.
- (핸들번호 중 5개는 system에서 사용하며 나머지 15개는 user가 사용한다)

2. 고수준 입출력 함수

1. fopen()함수

FILE *fopen(const char *filename, const char *mode);

파일(stream)을 연다.

(전달인자) filename : 파일 이름

mode : 파일 모드

mode	기 능
r	읽기 위해 open. 기존에 파일이 존재하여야 함, 없으면 에러
w	쓰기 위해 open. 없으면 생성, 존재하면 빔(지워짐)
a	추가하기 위해 open. 없으면 생성, 존재하면 추가
r+	읽기/쓰기 위해 open. 기존에 파일이 존재하여야 함.
w+	읽기/쓰기 위해 open. 없으면 생성, 존재하면 빔
a+	읽기/추가하기 위해 open. 없으면 생성, 존재하면 추가
b	기계어 상태로 open함
t	텍스트 상태로 open함

(반환값) 성공 : 열린 파일의 파일구조체의 시작 주소 / 실패 : NULL pointer

2. int fclose(FILE *stream) : 파일을 닫는다

(전달인자) - stream : 닫을 파일의 파일포인터

(반환값) 성공 : 0 / 실패 : EOF

표준 입출력장치(기정의 스트림)

장치명	기본할당	파일구조체 포인터명(핸들번호)
표준입력	키보드	stdin (0)
표준출력	모니터	stdout (1)
표준에러출력	모니터	stderr (2)
표준보조입출력	RS-232C	stdaux (3)
표준프린터출력	프린터	stdprn (4)

4. `int fgetc(FILE *stream)` : 파일에서 한 문자를 읽어들인다.
 (전달인자) `stream` : 데이터를 입력 받을 파일의 파일포인터
 (반환값) 성공 : 정상적으로 읽어들이 문자 / 실패 : EOF
5. `int fputc(int c, FILE *stream)` : 파일로 한 문자를 출력한다.
 (전달인자) `c` : 출력할 문자
 `stream` : 데이터를 출력할 파일의 파일포인터
 (반환값) 성공 : 출력한 문자 / 실패 : EOF

잠시 쉬어가기

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int i;
    for(i=0; i<argc; i++)
    {
        printf("%s\n", argv[i]);
    }
    return 0;
}
```

(위의 예제 실행하는 법)

1. Visual studio내에서 실행파일을 만든 후
 : 위의 소스코드를 입력한 후 솔루션빌드(F7키)로 compile, link까지만 진행하고 run은 시키지 않음
2. 1에서 만들어진 실행파일을 찾아 c:w 위치에 실행파일을 옮겨 놓음
3. 윈도우 프로그램중 [보조프로그램] - [명령 프롬프트]를 선택하여 콘솔창을 띄운 후 cd명령어로 폴더의 위치를 c:w 위치로 이동 한 후 2의 실행파일을 실행 시킨다.
 (실행방법)
 C:\W~W~W> cd c:\W(엔터) ← c:\W 위치로 프롬프트 변경
 C:\W> ex11-17 apple banana kiwi(엔터) ← 실행파일명과 문자열 3개를 함께 입력하여 실행파일을 실행 시킴

[예제 13-1] 파일 입출력 첫 예제

```

1 :  #include <stdio.h>
2 :  #include <assert.h>
3 :  #define FILE_OPEN(fp, filename, mode) if((fp=fopen(filename, mode))==NULL){ \
4 :      printf("File open error!\n"); \
5 :      return 0; \
6 :      }
7 :
8 :  int main(int argc, char *argv[])
9 :  {
10:      int c;
11:      FILE *ifp, *ofp;
12:      if(argc != 3) {
13:          printf("사용법 : %s 원본파일 복사본파일\n", argv[0]);
14:          return 0;
15:      }
16:      FILE_OPEN(ifp, argv[1], "rb");
17:      FILE_OPEN(ofp, argv[2], "wb");
18:
19:      while((c=fgetc(ifp))!=EOF) {
20:          fputc(c, ofp);
21:      }
22:      fclose(ifp);
23:      fclose(ofp);
24:      return 0;
25:  }

```

6. int feof(FILE *stream) : 파일의 EOF 체크 함수(매크로 함수)
 (전달인자) stream : EOF 체크를 할 함수의 파일포인터 변수명
 (반환값) 파일의 끝이면 : nonzero
 파일의 끝이 아니면 : 0

7. `char *fgets(char *s, int n, FILE *stream)` : 파일에서 문자열을 읽어들인다.
(전달인자) `s` : 읽어들인 문자열을 저장하는 영역
`n` : 읽어들일 최대문자수 (NULL문자 포함)
`stream` : 데이터를 입력받을 파일의 파일포인터
(반환값) 성공 : 읽어들인 문자열의 시작 주소 / 실패 : NULL pointer
8. `int fputs(const char *s, FILE *stream)` : 파일에 문자열을 출력한다.
(전달인자) `s` : 출력할 문자열
`stream` : 데이터를 출력할 파일의 파일포인터
(반환값) 성공 : 출력된 마지막 문자 / 실패 : EOF
9. `char *gets(char *s)` : 표준입력(stdin)으로부터 문자열을 읽어 들인다.
(전달인자) `s` : 읽어들인 문자열을 저장하는 영역
(반환값) 성공 : 읽어들인 문자열을 저장한 영역의 선두번지
실패 : NULL pointer (EOF를 입력했을 경우 포함)
10. `int puts(char *s)` : 문자열을 표준출력(stdout)에 출력한다.
(전달인자) `s` : 출력하는 문자열의 저장 영역
(반환값) 성공 : 음수이외의 값 반환 / 실패 : EOF
11. `int fprintf(FILE *stream, const char *format, argument....)`
출력양식(format)에 의해 파일로 출력한다.
(전달인자) `stream` : 출력할 대상 파일의 파일포인터
`format` : 출력양식을 나타내는 문자열
`argument` : 출력항목
(반환값) 성공 : 출력한 데이터의 개수 / 실패 : EOF
12. `int fscanf(FILE *stream, const char *format, argument....)`
파일로부터 입력양식(format)에 의해 입력한다.
(전달인자) `stream` : 입력할 대상 파일의 파일포인터
`format` : 입력양식을 나타내는 문자열
`argument` : 입력항목
(반환값) 성공 : 성공적으로 입력한 데이터의 개수 / 실패 : EOF

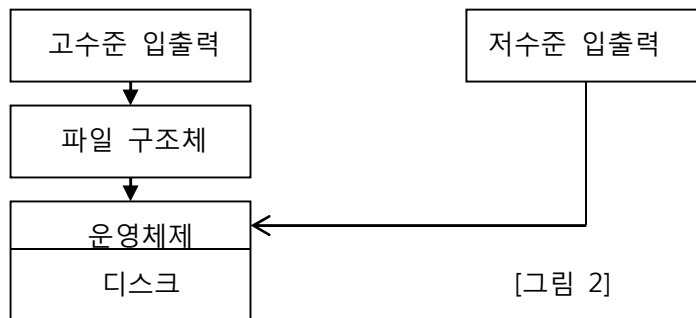
13. `size_t fread(void *ptr, size_t size, size_t n, FILE *stream)`
 파일에서 데이터를 블록단위로 읽어들이
 (전달인자) `ptr` : 읽어 들인 내용을 저장하는 버퍼
 `size` : 데이터 블록의 크기 (바이트 단위)
 `n` : 읽어 들일 블록의 개수
 `stream` : 입력파일의 파일포인터
 (반환값) 정상적으로 읽어 들인 데이터의 블록수
14. `size_t fwrite(void *ptr, size_t size, size_t n, FILE *stream)`
 파일에 데이터를 블록단위로 출력함
 (전달인자) `ptr` : 출력할 내용이 저장되어 있는 버퍼
 `size` : 한 블록의 크기
 `n` : 출력할 블록의 개수
 `stream` : 출력할 파일의 파일포인터
 (반환값) 정상적으로 출력한 데이터의 블록수
15. `void rewind(FILE *stream)` : 파일의 `current pointer`를 파일의 선두로 옮김
 (전달인자) `stream` : 대상파일의 파일포인터
 (반환값) 없음
16. `int fseek(FILE *stream, long offset, int origin)` : 파일의 `current pointer`를 지정된 위치로 옮긴다.
 (전달인자) `stream` : 대상 파일의 파일포인터
 `offset` : 새로운 위치까지의 상대거리
 `origin` : 기준위치 (`SEEK_SET` : 파일의 선두
 `SEEK_CUR` : 현재 `current pointer`의 위치
 `SEEK_END` : 파일의 맨 끝)
 (반환값) 성공 : 0
 실패 : 0 이외의 값
17. `long ftell(FILE *stream)` : 파일의 `current pointer`의 위치를 조사한다.
 (전달인자) `stream` : 대상 파일의 파일포인터
 (반환값) 성공 : 파일선두로부터 `current pointer`가 있는 곳까지의 바이트 수
 실패 : - 1L

[예제 13-2] 파일 입출력 예제

```
1 : #include<stdio.h>
2 : #include<assert.h>
3 : typedef struct _person
4 : {
5 :     char name[10];
6 :     int age;
7 :     double height;
8 : }Person;
9 : int main()
10: {
11:     FILE *fp;
12:     int i, res;
13:     int size;
14:     Person ary[5]={ {"홍길순", 500, 175.3},
15:                     {"이순신", 350, 179.5}, {"강감찬", 25, 175.5},
16:                     {"홍승연", 25, 175.3}, {"이길아", 46, 165.5}};
17:     Person temp;
18:     size = sizeof(ary) / sizeof(ary[0]);
19:
20:     fp=fopen("c:\\data\\ftest.txt", "wb");
21:     assert(fp!=NULL);
22:     for(i=0; i<size; i++)
23:     {
24:         fwrite(ary+i, sizeof(Person), 1, fp);
25:     }
26:     fclose(fp);
27:
28:     fp=fopen("c:\\data\\ftest.txt", "rb");
29:     assert(fp!=NULL);
30:     printf("[ 1차출력]\n");
31:     while(!feof(fp))
32:     {
33:         res=fread(&temp, sizeof(Person), 1, fp);
34:         if(res!=1){break;}
35:         printf("%s %d %.2lf\n",temp.name,temp.age,temp.height);
36:     }
37:
38:     rewind(fp);
39:
40:     printf("[ 2차출력]\n");
41:     while(!feof(fp)) ^
42:     {
43:         res=fread(&temp, sizeof(Person), 1, fp);
44:         if(res!=1){break;}
45:         printf("%s %d %.2lf\n",temp.name,temp.age,temp.height);
46:     }
47:     fclose(fp);
48: }
```

3. 고수준화일 I/O와 저수준화일 I/O의 차이

- 고수준 입출력 : FILE구조체를 통한 입출력, 처리속도가 약간 느리나 호환성 좋음
- 저수준 입출력 : 직접 운영체제에 접근하여 입출력을 수행.
처리속도가 빠르나 호환성이 떨어짐



4. 저수준 입출력 함수

1. `int open(char *pathname, int access)` : 저수준 입출력을 위하여 파일을 open한다
 (전달인자) `pathname` : open할 파일의 path명과 파일명을 지정한다.(path생략가능)
`access` : 파일 open mode 지정
 (반환 값) 성공 : 파일의 핸들번호 / 실패 : EOF (-1)

[파일 open mode]

open mode	mode의 의미
O_RDONLY	읽기 전용 모드
O_WRONLY	쓰기 전용 모드
O_RDWR	읽기 및 쓰기 겸용 모드
O_APPEND	기존 파일의 끝에다 추가
O_CREAT	신규로 생성
O_TRUNC	기존 파일의 내용을 삭제
O_BINARY	이진 파일 모드
O_TEXT	텍스트 파일 모드

파일의 모드는 Bit wise OR (|)로 연결하여 쓸 수 있다.

예) `int hd;`

`hd = open("score.dat", O_RDWR | O_CREAT | O_APPEND);`

score.dat 파일을 읽기 및 쓰기 겸용으로 열되, 존재하지 않으면 새로 생성하고,
존재할 경우에는 기존 파일 뒤에 자료를 추가한다.

2. `int close(int hd)` : 저수준으로 open된 파일을 닫는다.
 (전달인자) `hd` : 닫을 파일의 파일 핸들번호
 (반환 값) 성공 : 0
 실패 : EOF(-1)
3. `int read(int hd, void *buf, unsigned len)` : 파일로부터 지정한 byte 수 만큼 문자를 읽는다.
 (전달인자) `hd` : 읽어 들일 파일의 파일핸들 번호
`buf` : 읽어 들인 데이터를 저장할 기억장소
`len` : 읽어들일 데이터의 byte 수
 (반환 값) 성공 : 읽어들인 byte 수
 실패 : -1

4. `int write(int fd, void *buf, unsigned nbytes)` : 지정된 byte 수 만큼의 문자를 파일에 출력.

(전달인자) `fd` : 출력할 파일의 파일핸들 번호

`buf` : 출력할 데이터를 저장할 기억장소

`nbytes` : 출력할 데이터의 byte 수

(반환 값) 성공 : 출력된 byte 수

실패 : -1

[예제 13-3] 저수준 입출력 예제

```
1 :  #include <io.h>
2 :  #include <fcntl.h>
3 :  #include <stdio.h>
4 :  #include <conio.h>
5 :  int main()
6 :  {
7 :      char *buff = "ABCDEFGHIJKLMNOPQRSTUVWXYZ", str[100];
8 :      int handle,n;
9 :      handle=open("alphabet.dat", O_WRONLY | O_CREAT | O_TEXT);
10:     write(handle,buff, 26);
11:     write(handle, "\n", 1);
12:     close(handle);
13:     handle = open("alphabet.dat", O_RDONLY | O_TEXT);
14:     n=read(handle, str, 100);
15:     str[n] = NULL;
16:     printf("%d %s", n, str);
17:     close(handle);
18:     return 0;
19: }
```

제14장 기초 자료구조 & 알고리즘

학습 내용 소개

14장에서는 프로그램내에서의 데이터 처리를 위한 자료구조 개념을 배운다.
가장 기본이 되는 List, Stack, Queue를 배워서 프로그램내에서 사용되는 데이터를 다양하게 처리하는 기법을 배운다.

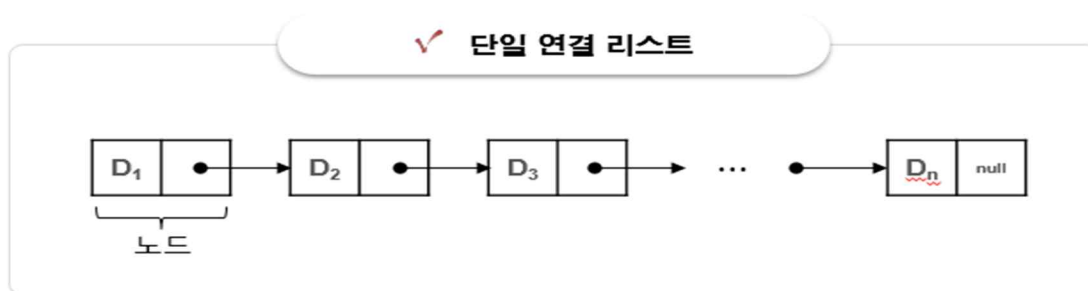
학습 목차

1. 연결리스트(Linked List)
2. 스택(Stack)
3. 큐(Queue)

1. 연결리스트(Linked List)

▶ 연결리스트(Linked List)

- 데이터를 저장한 단위 메모리가 서로 연결되어있는 자료구조
- 단일 연결 리스트(singly linked list) : 리스트의 각 노드에 다른 노드를 가리키는 포인터가 하나씩만 있는 연결리스트



▶ 연결리스트(linked list)와 배열의 비교

배열	비교 항목	연결 리스트
고정 크기 메모리(array) 사용	구현 방식	Node(구조체) 연결 방식
낮음	메모리 효율성	높음
느림(원소이동 필요)	삽입 & 삭제 속도	빠름(원소이동 필요 없음)
빠름	원소 접근 속도	느림
간단	구현 복잡도	복잡

▶ 단일 연결 리스트(Singly linked list) 관리를 위한 개념

- 연결 리스트를 실제 프로그램 내에서 사용하기 위해서는 첫 번째 node(head node)에 접근하기 위한 pointer 변수(head pointer)가 반드시 필요하다
- 또한 부수적으로 마지막 node(tail node)를 가리키는 pointer 변수(tail pointer)와 연결 리스트 내의 실제 데이터를 저장하고 있는 node의 개수(연결 리스트 크기)를 저장해 놓으면 연결 리스트 관리가 매우 수월 해진다

기본 용어 정리	연결 리스트의 주요 기능
<ul style="list-style-type: none"> • node : 연결 리스트의 요소로 [데이터 영역+포인터 영역(다음 node를 가리킴)]으로 구성됨 • head node & tail node : 연결 리스트내의 node 중 가장 첫 번째와 마지막에 위치한 node로 실제 데이터는 저장하지 않음(code작성 시 편의성을 고려한 개념) • head pointer : 연결 리스트내의 첫 번째 node를 가리키는 포인터 • tail pointer : 연결 리스트내의 마지막 node를 가리키는 포인터 • node count : 연결 리스트내의 실제 데이터를 저장하고 있는 데이터 노드의 개수 	<ol style="list-style-type: none"> 1. 연결 리스트 생성하기 2. node 추가하기 3. 전체 node 출력하기 4. node 검색하기 5. node 삭제하기 6. node 정렬하기 7. 연결 리스트 소멸하기

▶ 단일 연결 리스트 구현에 사용되는 데이터 형과 기능함수 목록

[linkedlist.h]

```
enum BOOL { FALSE, TRUE };
typedef struct _node Node;      /* 구조체 노드 형명재지정 */
struct _node{                  /* 노드 구조체 (자기참조 구조체 사용) */
    int data;                   /* 데이터 영역 : int형 데이터 저장 */
    Node *next;                 /* 포인터 영역 */
};
typedef struct _list{          /* 연결 리스트 관리 구조체 */
    Node *head;                /* head pointer (head node 가리킴) */
    Node *tail;                /* tail pointer (tail node 가리킴) */
    int size;                   /* 연결 리스트의 크기 - 실제 data node의 개수 */
}List;
BOOL createList(List *lp);      /* 연결 리스트 초기화 */
BOOL addFirst(List *lp, int data); /* head node 뒤에 node 추가(역순 저장) */
BOOL addLast(List *lp, int data); /* tail node 앞에 node 추가(정순 저장) */
void displayList(List *lp); /* 리스트 내의 모든 데이터 출력 */
Node * searchNode(List *lp, int data); /* data와 일치하는 node 검색 */
BOOL removeNode(List *lp, int data); /* data 노드 삭제 */
void sortList(List *lp);        /* 노드 정렬 - 오름차순 */
void destoryList(List *lp);     /* 리스트 내의 모든 노드 삭제 *
```

▶ 단일 연결 리스트 구현하기

[testMain.cpp]

```

int main()
{
    const char *menuList[] = { "입력하기", "출력하기", "검색하기",
                                "삭제하기", "정렬하기", "종료" };
    int menuNum; /* 메뉴번호 저장 변수 */
    int menuCnt; /* 메뉴개수 저장 변수 */
    List list; /* 리스트관리 구조체 변수 */
    BOOL bres;
    menuCnt = sizeof(menuList) / sizeof(menuList[0]);

    bres = createList(&list); /* 비어있는 리스트 초기화 */
    if (bres == TRUE)
        printf("@ list 생성 성공!\n");
    else
        printf("@ list 생성 실패!\n");
    while (1){
        menuNum = menu(menuList, menuCnt); /* 메뉴출력 및 메뉴번호 입력 */
        if (menuNum == menuCnt) { break; }
        switch (menuNum){
            case 1: mInput(&list); break; /* 입력메뉴 실행 */
            case 2: mOutput(&list); break; /* 출력메뉴 실행 */
            case 3: mSearch(&list); break; /* 검색메뉴 실행 */
            case 4: mDelete(&list); break; /* 삭제메뉴 실행 */
            case 5: mSort(&list); break; /* 정렬메뉴 실행 */
        }
    }
    printf("list내의 데이터 노드의 개수 : %d\n", list.size);
    destroyList(&list); /* 리스트내의 모든 데이터 삭제 */
    return 0;
}

```

[linkedlist.cpp]

```

/*-----
Function name      : createList - 연결 리스트 생성 및 초기화
Parameters       : lp - 리스트 관리 구조체의 주소
Returns          : 성공 - TRUE / 실패 - FALSE
----- */
BOOL createList(List *lp)
{
    /* TO DO */
}
/*-----
Function name      : addFirst - head node 뒤에 node 추가(역순 저장)
Parameters       : lp - 리스트 관리 구조체의 주소
                  : data - 추가할 데이터
Returns          : 성공 - TRUE / 실패 - FALSE
----- */
BOOL addFirst(List *lp, int data)
{

```

```

    /* TO DO */
}

/*-----
Function name      : addLast - tail node 앞에 새 node 추가(정순 저장)
Parameters       : lp - 리스트 관리 구조체의 주소
                  data - 추가할 데이터
Returns          : 성공 - TRUE / 실패 - FALSE
-----*/
BOOL addLast(List *lp, int data)
{
    /* TO DO */
}

/*-----
Function name      : displayList - 리스트 내의 모든 데이터 출력
Parameters       : lp - 리스트 관리 구조체의 주소
Returns          : 없음
-----*/
void displayList(List *lp)
{
    /* TO DO */
}

/*-----
Function name      : searchNode - data와 일치하는 첫 번째 node 검색
Parameters       : lp - 리스트 관리 구조체의 주소
                  data - 검색할 데이터
Returns          : 성공 - 검색된 노드의 주소 / 실패 - NULL pointer
-----*/
Node * searchNode(List *lp, int data)
{
    /* TO DO */
}

/*-----
Function name      : removeNode - data와 일치하는 첫 번째 노드 삭제
Parameters       : lp - 리스트 관리 구조체의 주소
                  data - 삭제할 데이터
Returns          : 성공 - TRUE / 실패 - FALSE
-----*/
BOOL removeNode(List *lp, int data)
{
    /* TO DO */
}

/*-----
Function name      : sortList - 노드 정렬(오름차순)
Parameters       : lp - 리스트 관리 구조체의 주소
Returns          : 없음
-----*/
void sortList(List *lp)
{
    /* TO DO */
}

/*-----
Function name      : destoryList - 리스트 내의 모든 노드(head, tail 노드 포함)를 삭제
Parameters       : lp - 리스트 관리 구조체의 주소
Returns          : 없음
-----*/

```

```

----- */
void destoryList(List *lp)
{
    /* TO DO */
}

```

[Algo-HW1] 단어 수 세기

- 지정한 파일로부터 데이터를 읽어서 파일 내의 단어의 개수와 단어의 평균 길이를 출력 한다
- 단어는 단일 링크드 리스트에 저장한다.
- 중복된 단어는 한번만 처리한다.
- 출력 시 단어를 오름차순으로 정렬하여 출력한다.

flower.txt	실행 결과
진달래 꽃 김소월 나 보기가 역겨워 가실 때에는 말 없이 고이 보내 드리오리다 영변에 약산 진달래 꽃 아름따다 가실 길에 뿌리오리다 나 보기가 역겨워 가실 때에는 죽어도 아니 눈물 흘리오리다	1번째 단어 : 가실 2번째 단어 : 고이 3번째 단어 : 길에 4번째 단어 : 김소월 5번째 단어 : 꽃 6번째 단어 : 나 7번째 단어 : 눈물 8번째 단어 : 드리오리다 9번째 단어 : 때에는 10번째 단어 : 말 11번째 단어 : 보기가 12번째 단어 : 보내 13번째 단어 : 뿌리오리다 14번째 단어 : 아니 15번째 단어 : 아름답다 16번째 단어 : 약산 17번째 단어 : 없이 18번째 단어 : 역겨워 19번째 단어 : 영변에 20번째 단어 : 죽어도 21번째 단어 : 진달래 22번째 단어 : 흘리오리다 전체 단어의 개수 : 22 평균 단어 길이 5.36

2. 스택(Stack)

▶ 스택(Stack)

- 스택은 자료를 차곡차곡 쌓아 올린 자료구조
- 자료의 입력과 출력은 한쪽 방향에서만 이루어짐
- 후입선출(後入先出)구조 (LIFO : Last Input First Output)
- 배열로 구현하거나 연결리스트로 구현할 수 있음



기본 용어 정리	스택의 주요 기능
<ul style="list-style-type: none"> ▪ size : 스택의 공간 크기 (즉, 저장 가능한 데이터의 양) ▪ push : 스택에 데이터를 넣는 작업 ▪ pop : 스택에서 데이터를 꺼내는 작업 ▪ top : 스택에서 입출력할 데이터의 위치 	<ol style="list-style-type: none"> 1. 스택 생성하기 2. 스택이 꽉 차있는지 검사 3. 스택이 완전히 비어있는지 검사 4. 스택 상단에 데이터 추가하기 5. 스택 상단에서 데이터 꺼내오기 6. 스택 내의 모든 데이터 출력하기 7. 스택 소멸하기

▶ 스택 구현에 사용되는 데이터 형과 기능함수 목록

[stack.h]

```
enum BOOL { FALSE, TRUE };
typedef struct _stack {
    int *stack; /* stack으로 사용되는 동적 할당 배열을 가리키는 포인터 변수 */
    int size; /* 스택의 크기(size) */
    int top; /* 스택의 입출구 위치 정보 저장 */
}Stack;
BOOL createStack(Stack *, int); /* 스택 메모리할당 및 멤버 초기화 함수 */
BOOL isStackFull(Stack *sPtr); /* 스택이 꽉 차있는지 검사 */
BOOL isStackEmpty(Stack *sPtr); /* 스택이 완전히 비어있는가 검사 */
BOOL push(Stack *, int); /* 스택에 데이터 저장하는 함수 */
BOOL pop(Stack *, int *); /* 스택에서 데이터를 꺼내는 함수 */
void printStack(const Stack*); /* 스택 내의 모든 데이터 출력 함수 */
void destroyStack(Stack *); /* 스택 메모리 해제 함수 */
```

[stack.cpp]

```

/*-----
Function name : createStack - 지정된 크기의 스택을 생성하는 함수
Parameters    : sp - 스택 관리 구조체의 주소
                size - 스택의 크기
Returns       : 성공 - TRUE / 실패 - FALSE
----- */
BOOL createStack(Stack *sp, int size)
{
    /* TO DO */
}
/*-----
Function name : isStackFull - 스택이 꽉 차있는지 검사
Parameters    : sp - 스택 관리 구조체의 주소
Returns       : 스택이 꽉 차있으면 TRUE, 아니면 FALSE 리턴
----- */
BOOL isStackFull(Stack *sp)
{
    /* TO DO */
}
/*-----
Function name : isStackEmpty - 스택이 완전히 비어있는지 검사
Parameters    : sp - 스택 관리 구조체의 주소
Returns       : 스택이 완전히 비어있으면 TRUE, 아니면 FALSE 리턴
----- */
BOOL isStackEmpty(Stack *sp)
{
    /* TO DO */
}
/*-----
Function name : push - 스택에 데이터 하나를 저장함
Parameters    : sp - 스택 관리 구조체의 주소
                inData - 스택에 저장할 데이터
Returns       : 성공적으로 저장하면 TRUE, 실패하면 FALSE 리턴
----- */
BOOL push(Stack *sp, int inData)
{
    /* TO DO */
}
/*-----
Function name : pop - 스택에서 데이터 하나를 꺼냄
Parameters    : sp - 스택 관리 구조체의 주소
                popData - 꺼낸 데이터를 저장할 기억 공간의 주소
Returns       : 성공적으로 꺼내면 TRUE, 실패하면 FALSE 리턴
----- */
BOOL pop(Stack * sp, int *popData)
{
    /* TO DO */
}
/*-----
Function name : printStack - 스택의 모든 데이터 출력(pop하면 나오는 순서대로 출력)
Parameters    : sp - 스택 관리 구조체의 주소
Returns       : 없음
----- */

```



```

void printStack(const Stack* sp, void(*print)(int *p))
{
    /* TO DO */
}
/*-----
Function name : destroyStack - 스택 소멸 함수
Parameters   : sp - 스택 관리 구조체의 주소
Returns      : 없음
-----*/
void destroyStack(Stack *sp)
{
    /* TO DO */
}

```

[Algo-HW2] 편집기(Editor)

한 줄로 된 간단한 라인 편집기(Editor) 를 구현하려고 한다.

이 편집기는 영어 소문자만을 기록할 수 있는 편집기로, 최대 500,000 글자까지 입력할 수 있다.

이 편집기에는 '커서'라는 것이 있는데, 커서는 문장의 맨 앞(첫 번째 문자의 왼쪽), 문장의 맨 뒤(마지막 문자의 오른쪽), 또는 문장 중간 임의의 곳(모든 연속된 두 문자 사이)에 위치할 수 있다.

즉 길이가 len 인 문자열이 현재 편집기에 입력되어 있으면, 커서가 위치할 수 있는 곳은 len+1 가지 경우가 있다.

- 이 편집기가 지원하는 명령어는 다음과 같다

L	커서를 왼쪽으로 한 칸 옮김(커서가 문장의 맨 앞이면 무시됨)
D	커서를 오른쪽으로 한 칸 옮김(커서가 문장의 맨 뒤이면 무시됨)
B	커서 왼쪽에 있는 문자를 삭제함(커서가 문장의 맨 앞이면 무시됨) 삭제로 인해 커서는 한 칸 왼쪽으로 이동한 것처럼 나타나지만, 실제로 커서의 오른쪽에 있던 문자는 그대로임
P\$	\$라는 문자를 커서 앞쪽에 추가함 - 커서는 \$의 오른쪽에 위치함

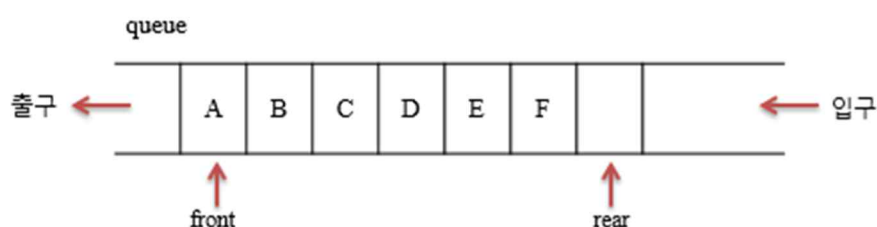
초기에 편집기에 입력되어 있는 문자열이 주어지고, 그 이후 입력한 명령어가 차례로 주어졌을 때, 모든 명령어를 수행하고 난 후 편집기에 입력되어 있는 문자열을 구하는 프로그램을 작성한다. 단, 명령어가 수행되기 전에 커서는 문장의 맨 뒤에 위치하고 있다고 가정한다.

데이터 파일의 내용	실행 결과	
<p>첫째 줄에는 초기에 편집기에 입력되어 있는 문자열이 주어진다. 이 문자열은 영어 소문자로만 이루어져 있으며, 길이는 100,000을 넘지 않는다</p> <p>둘째 줄에는 입력할 명령어의 개수를 나타내는 정수 $N(1 \leq N \leq 500,000)$이 주어진다</p> <p>셋째 줄부터 N개의 줄에 걸쳐 입력할 명령어가 순서대로 주어진다.</p> <p>명령어는 위의 네 가지 중 하나의 형태로만 주어진다.</p> <ul style="list-style-type: none"> - L : 커서 왼쪽으로 이동 - D : 커서 오른쪽으로 이동 - B : 커서 왼쪽의 문자 삭제 - P\$: 커서 앞에 문자 추가 	입력 예	출력 예
	abcd 5 L L P x D P y	abxcyd
	abcd 16 P x L P y D D P z B B L L L P r L B B B	rcdy

3. 큐(Queue)

▶ 큐(Queue)

- 큐는 입구와 출구가 구분된 자료 구조
- 선입선출(先入先出)구조(FIFO : First Input First Output)
- 배열로 구현하거나 연결리스트로 구현할 수 있음

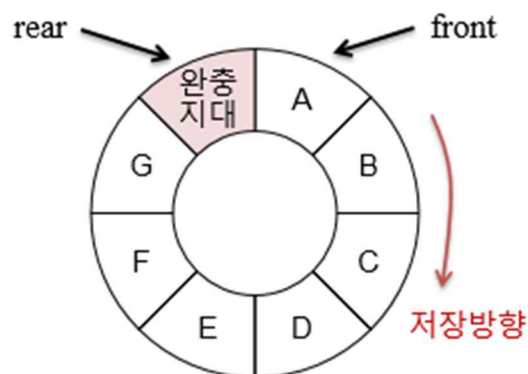


기본 용어 정리	큐의 주요 기능
<ul style="list-style-type: none"> • queue size : 큐의 공간 크기 (즉, 저장 가능한 데이터의 양) • put(enqueue) : 큐에 데이터를 넣는 작업 • get(dequeue) : 큐에서 데이터를 꺼내는 작업 • front : 큐에서 첫 데이터의 위치 (즉, 출력할 데이터의 위치) • rear : 큐의 마지막 데이터의 한 칸 다음 위치 (즉, 입력될 데이터가 들어올 위치) 	<ol style="list-style-type: none"> 1. 큐 생성하기 2. 큐가 꽉 차있는지 검사 3. 큐가 완전히 비어있는가 검사 4. 큐에 데이터 추가하기 5. 큐에서 데이터 꺼내오기 6. 큐 내의 모든 데이터를 출력하기 7. 큐 소멸하기

※ 일반 queue의 문제점은?

▶ 원형 큐(Circular Queue)

- 일반 큐의 단점을 해결한 것이 바로 원형 큐(circular queue)이다
- 원형 큐는 선입선출(先入先出)구조 (FIFO : First Input First Output)를 그대로 유지하면서 큐의 입구와 출구를 연결하여 원형으로 만들어 사용
- 완충지대 : 원형 큐의 비어있는 상태와 꽉 차있는 상태를 구분하기 위해 front와 rear사이에 완충지대가 필요함
- 원형 큐에서는 rear가 완충지대를 가리키고 있다면 꽉 찬 상태를 의미하며, front와 rear가 한 곳을 가리키면 비어있는 상태를 의미
- 원형 큐 역시 배열로 구현하거나 연결리스트로 구현할 수 있음



▶ 원형큐(circular queue) 구현에 사용되는 데이터 형과 기능함수 목록

[circularQueue.h]

```
enum BOOL { FALSE, TRUE };
typedef struct _node Node;
struct _node
{
    int data;
    Node *next;
};

typedef struct _queue {
    Node *head;
    Node *tail;
}Queue
BOOL createQueue(Queue * qp, int size);    /* 큐 생성 및 초기화 함수 */
BOOL isEmptyQueue(const Queue *qp); /* 큐가 완전히 비어있는가 */
BOOL enqueue(Queue * qp, int enqueueData); /* 큐에 데이터 하나를 저장 함 */
BOOL dequeue(Queue * qp, int * dequeData); /* 큐에서 데이터 하나를 꺼냄 */
void printQueue(const Queue * qp); /* 큐 내의 모든 데이터를 출력
                                   (dequeue하는 것은 아님) */
void destroyQueue(Queue * qp);    /* 큐 소멸 함수 */
```

[circularQueue.cpp]

```
/*-----
Function name : createQueue - 원형 큐 생성 및 초기화 함수
Parameters    : qp - 큐 구조체의 주소 , size - 큐의 크기
Returns       : 없음
-----*/
BOOL createQueue(Queue * qp, int size)
{
    /* TO DO */
}
/*-----
Function name : isEmptyQueue - 큐가 완전히 비어있는가 검사
Parameters    : qp - 큐 구조체의 주소
Returns       : 완전히 비어있으면 TRUE 리턴, 비어있지 않으면 FALSE 리턴
-----*/
BOOL isEmptyQueue(const Queue *qp)
{
    /* TO DO */
}
/*-----
Function name : isQueueFull - 큐가 꽉차있는가 검사
Parameters    : qp - 큐 구조체의 주소
Returns       : 꽉 차 있으면 TRUE 리턴, 아니면 FALSE 리턴
-----*/
BOOL isQueueFull(const Queue *qp)
{
    /* TO DO */
}
/*-----
Function name : enqueue() - 큐에 데이터 하나를 저장함
Parameters    : qp - 큐 구조체의 주소, enqueueData - 큐에 저장할 데이터
Returns       : 성공적으로 저장하면 TRUE, 실패하면 FALSE 리턴
-----*/
```

```

----- */
BOOL enqueue(Queue *qp, int enqueueData)
{
    /* TO DO */
}
/*-----
Function name : dequeue() - 큐에서 데이터 하나를 꺼냄
Parameters    : qp - 큐 구조체의 주소
                dequeData - 꺼낸 데이터를 저장할 기억공간의 주소
Returns       : 성공적으로 꺼내면 TRUE, 실패하면 FALSE 리턴
----- */
BOOL dequeue(Queue *qp, int *dequeData)
{
    /* TO DO */
}
/*-----
Function name : printQueue() - 큐 내의 모든 데이터를 출력 함
Parameters    : qp - 큐 구조체의 주소
Returns       : 없음
----- */
void printQueue(const Queue *qp)
{
    /* TO DO */
}
/*-----
Function name : destroyQueue() - 큐 소멸 함수
Parameters    : qp - 큐 구조체의 주소
Returns       : 없음
----- */
void destroyQueue(Queue * qp)
{
    /* TO DO */
}

```

[Algo-HW3] 조세퍼스의 순열

1번부터 N번까지 N명의 사람이 원을 이루면서 앉아있고, 양의 정수 $M(M \leq N)$ 이 주어진다

이제 순서대로 M번째 사람을 제거한다. 한 사람이 제거되면 남은 사람들로 이루어진 원을 따라 이 과정을 계속해 나간다. 이 과정은 N명의 사람이 모두 제거될 때까지 계속된다

원에서 사람들이 제거되는 순서를 (N, M)-조세퍼스 순열이라고 한다. 예를 들어 (7, 3)-조세퍼스 순열은 <3, 6, 2, 7, 5, 1, 4>이다. N과 M이 주어지면 (N,M)-조세퍼스 순열을 구하는 프로그램을 작성한다.

- 입력 : 첫째 줄에 N과 M이 빈 칸을 사이에 두고 순서대로 주어진다($1 \leq M \leq N \leq 5,000$)
- 출력 : 예제와 같이 조세퍼스 순열을 출력한다

N(인원수)와 M(간격수)를 입력하시오 ($M \leq N$) : 7 3

(7,3)조세퍼스의 순열 : 3 6 2 7 5 1 4

N(인원수)와 M(간격수)를 입력하시오 ($M \leq N$) : 10 5

(10,5)조세퍼스의 순열 : 5 10 6 2 9 8 1 4 7 3

과제집

[과제집]

※ 과제 제출 요령

1. 과제 제출메일 : myloveC@gmail.com
2. 메일 제목 : 수강생 본인성명 HW1 제출
3. 제출파일명 : 해당 과제의 source파일만 제출(project 폴더를 통째로 압축하지 말 것)
과제번호와 파일명의 번호가 일치하도록 할 것 (ex : HW1.cpp)

[HW1] 나에 대한 정보를 출력하는 프로그램을 작성하시오.

성명은 main()함수에서 출력하고 나이와 키는 각각을 출력하는 함수를 만들어서 출력하시오.
이때, 나이와 키의 값은 main()함수 내에서 선언된 변수에 저장해 놓은 후 나이와 키를 출력하는
함수를 호출할 때 전달인자로 보내서 출력하자.

```
#include<stdio.h>
// TODO          // printAge() 함수의 선언부
// TODO          // printHeight() 함수의 선언부

int main()
{
    char name[20] = "본인성명";
    int age = 500;
    double height = 170.8;

    printf("성명 : %s\n", name);
    printAge(age);
    printHeight(height);

    return 0;
}
// printAge() 함수의 정의부
```

```
// TODO
```

```
//printHeight() 함수의 정의부
```

```
// TODO
```


[HW2] 용돈 화폐단위 환산 프로그램

278970원(int형 변수로 저장)을 50000원권, 10000원권, 5000원권, 1000원권, 500원권, 100원권, 50원권, 10원권 단위로 환산하여 각각의 개수를 출력하시오.

(실행결과)

50000원권 => 5개

10000원권 => 2개

5000원권 => 1개

1000원권 => 3개

500원권 => 1개

100원권 => 4개

50원권 => 1개

10원권 => 2개

$$365 \overline{) 606} \\ 365$$

[HW3] 초 수를 시, 분, 초단위로 환산하는 프로그램

54321초를 시,분,초 단위로 환산하시오.

(실행결과)

54321초는 15시간 5분 21초 입니다.

$$t = 365.2422$$

$$day = 365 / 1$$

[HW4] 1년을 일,시,분,초로 환산하는 프로그램

1년은 365.2422일이다. 이는 몇 일, 몇 시간, 몇 분, 몇 초 인가 출력하시오.

단, 초수는 소수점 이하 둘째 자리까지 출력하시오

(실행결과)

24

365.2422일은 365일 5시간 48분 46.08초입니다.

$$(0.2422 * 24 * 60 * 60)$$

$$hour = (t - day) * 60 * 60$$

$$minute = (t - day - 0.6 * hour) * 60$$

$$second = (t - day - 0.6 * 0.1 * hour - 0.6 * minute)$$

[HW5] 전기요금 계산하기

전기 사용량을 kw단위로 입력하면 사용 요금을 계산해주는 프로그램을 작성한다. 조건은 다음과 같다.

(사용함수) 입력함수, 출력함수, 계산은 main함수에서 모두 계산 함

- 기본요금 : 660 원

- kw 당 사용요금 : 88.5 원

- 전체 요금 = 기본요금 + (사용량 * kw 당 사용요금)

- 세금은 전체 요금의 9%

- 최종 사용 요금 = 전체 요금 + 세금

(입출력 예)

전기 사용량을 입력하세요(kw) : 150(엔터) ← 입력함수에서 처리

전기 사용요금은 15189.150000 원 입니다. ← 출력 함수에서 처리

dragon_
money

--

[HW6] 온도 변환하기

키보드로 화씨(°F)온도를 입력 받고 섭씨(°C)온도를 계산하여 출력하는 프로그램을 작성한다.
 섭씨온도는 소수점 이하 첫째 자리까지 출력한다.
 화씨온도를 섭씨온도로 바꾸는 식은 다음과 같다.

$$C = 5/9(F - 32) \quad (C : \text{섭씨}, F : \text{화씨})$$

(사용함수) 입력함수, 출력함수

(입출력 예)

화씨 온도를 입력하세요 : 40(엔터)

섭씨 온도는 4.4 도입니다.

[HW7] 주행거리와 시속을 입력 받아 주행시간은 계산하기

주행거리의 단위는 Km, 시속은 km/h, 주행시간은 시,분,초단위로 출력하시오.
 단, 초수는 소수점 이하 셋째 자리까지 출력하시오

(실행결과)

* 거리를 입력하시오(km단위) : 250.7

* 시속을 입력하시오(km/h단위) : 67.3

250.70 km = >3시간 43분 30.401 초 소요됨

* 거리를 입력하시오(km단위) : 400

* 시속을 입력하시오(km/h단위) : 100

400.00 km = >4시간 0분 0.000 초 소요됨

* 거리를 입력하시오(km단위) : 12751.7

* 시속을 입력하시오(km/h단위) : 94.5

12751.70 km = >134시간 56분 19.048 초 소요됨

[HW8] 세 과목 점수 입력 받아 총점, 평균 출력하기

역사, 문학, 예능 세 과목의 점수를 입력 받아서 총점과 평균을 구한 후에 출력하는 프로그램을 작성한다. 입력 점수는 모두 양의 정수 값이며 평균이 실수 값이 나올 수 있도록 값을 입력하자.

(입출력 예)

역사, 문학, 예능 점수를 입력하세요 : 70 85 90(엔터)

총점은 245 이고 평균은 81.67 입니다.

[HW9] 입력된 정수를 8 진수와 16 진수로 출력하는 프로그램

키보드로부터 정수 값을 입력 받아 8 진수와 16 진수로 출력하는 프로그램을 작성한다.

이 때 진법을 나타내는 접두어도 함께 출력되도록 한다.
(사용함수) 정수 입력함수

(입출력 예)

정수값을 입력하세요 : 32165(엔터)
입력된 값은 8 진수로 076645 입니다.
입력된 값은 16 진수로 0x7da5 입니다.

```
#include<stdio.h>
// TODO - input() 함수 선언부
int main()
{
    int number;
    number = input();
    // TODO - 계산결과 출력하기
    return 0;
}
```

[HW10] 두 정수를 입력 받아 합, 차, 곱, 몫의 값을 출력하는 프로그램 작성
단, 실수 값은 소수점 이하 둘째 자리까지 출력하시오

(입출력 예)

두개의 정수를 입력하시오 : 10 20(엔터)
10+20 = 30
10-20 = -10;
10*20 = 200
10/20 = 0.50

[HW11] ASCII code 를 입력 받아 ASCII code 에 해당 문자 출력하기

(입출력 예)

ASCII code값을 입력하시오 : 67
67은 'C'의 ASCII code 입니다.

[HW12] 이름을 영문으로 입력 받아 다음과 같은 형태로 출력하시오.

- 이름 전체를 ""로 묶어서 출력하시오.
- 필드폭을 20 자로 이름을 출력하되 전 필드를 ""로 감싸서 출력 한다.
- 필드폭을 20 자로 이름을 출력하되 전 필드를 ""로 감싸서 왼쪽 정렬하여 출력한다.

(입출력 예)

이름을 입력하시오 : Hong GilSoon

(출력형태)

"Hong GilSoon"

```
"      Hong GilSoon"
"Hong GilSoon"
```

[HW13] 이름의 글자 수 출력하기

성과 이름을 입력 받아 입력된 이름을 출력하고 다음 라인에 성과 이름의 글자수를 각각 성과 이름 뒷자리에 맞추어 출력하시오.

```
(입력 예)
#성을 입력하시오 : Hong
#이름을 입력하시오 : Gildong
(출력 예)
Hong  Gildong
    4      7
```

[HW14] 문자열 생략하여 출력하기

문자열을 입력 받아 입력 받은 문자열을 앞에서부터 반만 출력하시오.

출력 시 대괄호[]로 묶어서 출력하되 문자열의 길이만큼의 필드폭에 문자열의 반만 출력하고 문자열 뒤에는 생략기호(...)를 붙여서 출력하시오.

```
(입출력 예)
* 문자열 입력 : stream
[□□□str. . .]
* 문자열 입력 : dream
[□□□dr. . .]
* 문자열 입력 : at
[□a. . .]
* 문자열 입력 : a
[□. . .]
```

chap4. 기초 제어문

[HW15] BMI 구하기

몸무게와 키를 입력 받아서 신체질량지수(BMI)를 구한 후 비만도를 출력해주는 프로그램을 작성하시오.

BMI 는 비만도를 측정하는 한가지 방법으로 다음과 같이 계산한다.

BMI = 몸무게(kg) / (키(m)의 제곱) ◀ 키의 단위는 미터 단위

* BMI 수치에 따른 소견표

저체중 : 18.5 미만	정상체중 : 18.5~25.0 미만
과체중 : 25.0~30.0 미만	비만 : 30.0~40.0 미만
고도비만 : 40.0 이상	

(사용함수) 입력함수, bmi계산 함수, 출력함수

(입출력 예)

몸무게를 입력하세요(kg) : 45(엔터)

키를 입력하세요(m) : 1.68(엔터)

당신의 BMI는 15.9으로 저체중입니다

[HW16] 지하철 요금 계산하기

역수에 따라 적용되는 규칙은 다음과 같습니다.

(계산 규칙) 기본 : 1~5 정거장까지는 600원, 6~10 정거장까지는 800원

10정거장 초과시 1~2정거장 100원, 3~4 정거장 200원 추가의 형식으로 계산

(사용함수) 입력함수, 요금 계산함수, 출력함수

5회

(입출력 예)

역수를 입력하시오 : 4

요금 : 600원

역수를 입력하시오 : 7

요금 : 800원

역수를 입력하시오 : 11

요금 : 900원

역수를 입력하시오 : 12

요금 : 900원

역수를 입력하시오 : 20

요금 : 1300원

[HW17] 성명, 키, 성별을 입력 받아 실행 예와 같이 수행되는 프로그램을 작성하시오.

(사용함수) main()함수만 구현

(실행예1)

성명 입력 : Lee soon sin(엔터)

키 입력(cm단위) : 175.4(엔터)

성별입력(M/F) : M(엔터)

Lee soon sin씨의 키는 175.40cm이고 남성입니다.

(실행예2)

성명 입력 : 신사임당(엔터)

키 입력(cm단위) : 158.54(엔터)

성별입력(M/F) : F(엔터)

신사임당씨의 키는 158.54cm이고 여성입니다.

[HW18] 입장료 계산 프로그램

놀이동산 입장객의 나이와 인원수를 입력 받아 입장료를 계산하여 출력하기
이때 인원수가 5명 이상이면 단체로 인정하여 총 입장료의 10%를 할인하여 계산한다.

연 령	입장료(1인 기준)
7세 이하	500원
8세~13세	700원
14세~19세	1000원
20세~55세	1500원
56세 이상	500원

(사용함수) 입력함수, 출력함수

(실행 예1)

입장객의 나이를 입력하시오 : 21(엔터)

입장객의 수를 입력하시오 : 3(엔터)

/ 함수

입장료 => 4500원

할인금액 => 0원

결제금액 => 4500원

(실행 예2)

입장객의 나이를 입력하시오 : 15(엔터)

입장객의 수를 입력하시오 : 6(엔터)

입장료 => 6000원

할인금액 => 600원

결제금액 => 5400원

[HW19] 윤,평년 구하는 프로그램

< 윤,평년 알고리즘 >

1. 년도를 4로 나누어 => 떨어지면 2번 조건을 검사한다.
=> 떨어지지 않으면 평년
2. 년도를 100으로 나누어 => 떨어지면 3번 조건을 검사한다.
=> 떨어지지 않으면 윤년
3. 년도를 400으로 나누어 => 떨어지면 윤년
=> 떨어지지 않으면 평년

위의 윤년계산법을 이용하여 년도를 입력 받아 검사를 한 후 결과를 다음과 같이 출력하는 프로그램을 작성하시오.

(사용함수) 입력함수, yearCheck 함수(전달 받은 년도를 검사하여 윤년이면 1, 평년이면 0 리턴)

(입출력 예)

년도를 입력하시오 : 2010

2010년은 평년(Common year)입니다.

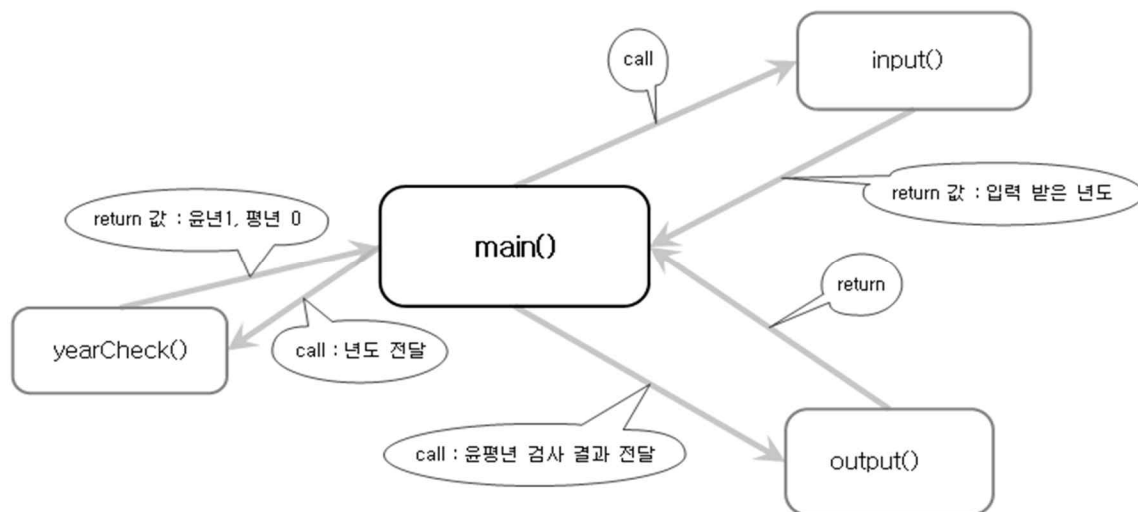
년도를 입력하시오 : 2004

2004년은 윤년(Leap year)입니다.

- test data

윤년 : 1600, 2000, 2008, 4, 1996

평년 : 1700, 2002, 2010, 7, 1995



[HW20] Pay 계산하기

1주일간 일한 시간을 입력하여 총수입(gross pay), 세금(taxes), 실수입(net pay)를 출력하시오.

(계산방법)

- 기본급여 = 시간당 3,000 원
- 초과근무수당 (40 시간 초과분에 대하여 적용) = 시간 * 기본급 * 1.5 배
- 세율 : 처음 100,000 원까지 15%, 100,000 만원 초과 금액은 25%

(사용함수) 입력함수, 총수입 계산함수, 세금 계산함수, 결과 출력함수

(입출력양식)

* 1주일간의 근무시간을 입력하시오 : 50

총수입 : 165000원

세 금 : 31250원

실수입 : 133750원

- test data

근무시간 40 시간 : 총수입 120000, 세금 20000, 실수입 100000

근무시간 45 시간 : 총수입 142500, 세금 25625, 실수입 116875

근무시간 60 시간 : 총수입 210000, 세금 42500, 실수입 167500

[HW21] 다섯 명의 학생의 키를 입력 받아 그 평균을 출력하기

이때 키는 실수값으로 입력하며, 평균은 소수점 이하 첫 째 자리까지 출력하시오.

(사용함수) main()함수 내에서 모두 처리

(입출력 예)

- 1 번 학생의 키는? 170.5 (엔터)

- 2 번 학생의 키는? 175.3 (엔터)

- 3 번 학생의 키는? 167.8 (엔터)

- 4 번 학생의 키는? 156.2 (엔터)

- 5 번 학생의 키는? 163.5 (엔터)

다섯 명의 평균 키는 xxx.x cm 입니다.

[HW22] 한줄에 5 개씩 별(*) 출력하기

키보드로부터 정수값을 입력 받아 그 숫자 만큼 별(*)을 출력하되 한 줄에 5 개씩 출력하도록 하자.

(힌트 : 단일 반복문과 if 문을 사용)

(사용함수) 입력함수(문자 입력 예외처리)

(입출력 예)

정수값을 입력하세요 : 17(엔터)

**

[HW23] 반복적으로 두 개의 정수를 입력 받아 두수의 차를 출력하시오.

단, 두수 중 큰 수에서 작은 수를 빼준다.

반복 종료조건은 숫자 대신에 문자를 입력하면 종료 함.

(사용함수) 모두 main()함수에서 처리

(실행 예)

두개의 정수를 입력하세요 : 10 15

15 - 10 = 5

두개의 정수를 입력하세요 : 9 5

9 - 5 = 4

두개의 정수를 입력하세요 : -3 8

8 - -3 = 11

두개의 정수를 입력하세요 : 2 # ← 두 숫자 중 하나 또는 모두 문자가 입력되면 반복을 종료

[HW24] 달팽이 우물탈출 프로그램

어느 날 달팽이 한 마리가 우물에 빠졌다.

우물의 깊이는 cm 단위(정수형)로 입력 받는다.

낮 동안 달팽이가 올라가는 거리는 50cm, 밤 동안 미끄러져 내려가는 거리는 20cm이다.

달팽이가 우물을 탈출하기까지 며칠이 걸리는지 계산하시오.

(사용함수) 입력함수, 탈출날짜 계산함수, 결과 출력함수

(입출력 예)

* 우물의 깊이를 입력하시오(cm단위) : 500

5.00 미터 깊이의 우물을 탈출하기 위해서는 16일이 걸립니다.

* 우물의 깊이를 입력하시오(cm단위) : 200

2.00미터 깊이의 우물을 탈출하기 위해서는 6일이 걸립니다.

- test data

우물높이 0cm : 0일 우물높이 1~50cm : 1일

우물높이 51cm : 2일 우물높이 80cm : 2일

우물높이 81cm : 3일 우물높이 200cm : 6일

[HW25] 숫자 맞추기 게임

1~100 까지의 수 중에서 난수(random number) 한 개를 발생시킨 후 사용자가 알아 맞추는 게임을 만드시오. 예를 들어 컴퓨터가 발생시킨 난수가 39 일 경우 아래와 같은 입출력 과정을 통해서 맞추어가는 과정을 출력해주고, 최종적으로 몇 번째 만에 맞췄는지 출력하시오.

(사용함수)

1. 입력함수 - 문자 및 범위(1~100)밖의 값 입력 시 재입력 요구

2. 출력함수 - 결과 출력 함수. 단, 최종 메시지는 main() 함수에서 출력

(입출력 예)

숫자를 입력하시오 : 50

0 보다는 크고 50 보다는 작습니다.

#숫자를 입력하시오 : 25

25 보다는 크고 50 보다는 작습니다.

#숫자를 입력하시오 : 35

35 보다는 크고 50 보다는 작습니다.

#숫자를 입력하시오 : 40

35 보다는 크고 40 보다는 작습니다.

#숫자를 입력하시오 : 39

우와~ 맞았당~~~ 추카추카~~ 5 번째 만에 맞추셨습니다.

(난수 발생 예제)

```
#include<stdlib.h> // rand(), srand() 함수를 사용하기 위해서 인클루드 함
#include<time.h> // time() 함수를 사용하기 위해서 인클루드 함
#include<stdio.h>
int random(int);

int main()
{
    int num;
    int i;
    srand( (unsigned int)time (NULL) ); // 필수 호출!!
    // 시스템 시간을 난수 씨앗 값으로 사용해서 실행할 때 마다 다른 난수가 발생함
    // 아래 random()함수 사용전에 꼭 한번만 호출해 주어야 함
    // 주의 사항 : 이 부분은 프로그램 시작할 때 딱 1회만 수행되게 해야 함
    // 여러 번 수행시키면 프로그램의 실행속도를 매우 저하시키는 현상이 나타남
    i=0;
    while(i<5){ // 반복문을 이용해서 5개의 난수를 발생시켜 출력
        num = random(10); // 0~9사이의 난수를 발생시킴
        printf("%d\n", num);
        ++i;
    }
    return 0;
}

int random(int n) // 난수 발생 함수
{
    int res;
    res = rand() % n; // 0부터 n-1까지의 수중 난수 발생
    return res;
}
```

[HW26] 가위 바위 보 게임을 만들어 보자.

사용자로부터 가위 바위 보 중에서 하나를 입력 받고, 컴퓨터는 난수(random number) 생성을 통해서 가위 바위 보 중에서 하나를 선택하게 한다.

이 둘을 비교해서 승자와 패자를 가려주는 프로그램을 작성해 보자.

단, 프로그램의 진행은 사용자가 질 때까지 계속되어야 하고, 마지막으로 가서는 게임의 결과(a 승 b 무)까지 출력해 주도록 하자.

(사용함수)

1. 입력함수 - 문자입력 및 1~3사이의 숫자가 아닐 시 재입력 요구
2. 출력함수 - 게임최종 결과는 main()함수에서 출력

(입출력 예)

- ```
바위는 1, 가위는 2, 보는 3 중에서 선택하세요 : 2(Enter key)
당신은 가위 선택, 컴퓨터는 가위 선택 : 비겼습니다.
바위는 1, 가위는 2, 보는 3 중에서 선택하세요 : 1(Enter key)
당신은 바위 선택, 컴퓨터는 가위 선택 : 이겼습니다.
바위는 1, 가위는 2, 보는 3 중에서 선택하세요 : 3(Enter key)
당신은 보 선택, 컴퓨터는 보 선택 : 비겼습니다.
바위는 1, 가위는 2, 보는 3 중에서 선택하세요 : 2(Enter key)
```

당신은 가위 선택, 컴퓨터는 바위 선택 : 당신이 졌습니다.  
게임결과 : 1 승 2 무

### [HW27] 달걀 포장 프로그램 작성하기

달걀의 무게를 입력 받아 150g~500g 이면 포장하고 150g 미만인 것과 500g 초과인 경우는 포장에서 제외하여 총 10 개의 달걀을 한 박스 포장하는 프로그램을 작성하시오.

(사용함수)

1. 입력함수 - 문자입력 시 재입력 요구

(실행 예)

```
계란의 무게를 입력하세요(단위 : g) : 160(엔터)
* 현재 달걀의 수 : 1
#계란의 무게를 입력하세요(단위 : g) : 100(엔터)
* 메추리알 가지고 장난하지 마시오~ ^^
계란의 무게를 입력하세요(단위 : g) : 550(엔터)
* 타조알 가지고 장난하지 마시오~ ^^
:
계란의 무게를 입력하세요(단위 : g) : 350
* 현재 달걀의 수 : 9
계란의 무게를 입력하세요(단위 : g) : 190
* 현재 달걀의 수 : 10
```

\*\*\* 달걀 포장이 끝났습니다.

### [HW28] 문자 종류별 카운트 프로그램. (getchar()함수 이용)

영문 문장을 키보드로 부터 입력 받아 각각 문자를 종류별로 카운트하여 출력하시오.  
입력 종료 조건 : 새줄(new line)에서 EOF 입력 시 종료 함

(실행 예)

```
영문 문장을 입력 하시오 :
Hello! My name is sophia.
My number : [010-9207-8234]
e-mail address : [mylovec@gmail.com]
Thank you. ^^
^Z ← EOF(ctrl+Z) 입력
```

```
* 영문자 대소문자 개수 : 62개
* 숫자문자 개수 : 11개
* 여백문자(space, tab, enter) 개수 : 16개
* 그 외 기타문자 개수 : 16개
```

|                |               |
|----------------|---------------|
| 시작 값(P1) : 100 | 고집수가 5인 숫자 출력 |
| 끝 값(P2) : 9999 | 679           |
| 고집수(N) : 5     | 688           |
|                | 697           |
|                | 769           |
|                | 796           |
|                | 868           |
|                | 886           |
|                | 967           |
|                | :             |
|                | 9755          |
|                | 9761          |
|                | 9767          |
|                | 9774          |
|                | 9776          |
|                | 9969          |
|                | 9996          |
|                | 총 개수 : 219개   |

**[HW29] 고집수**

양의 정수(100~10000)를 입력 받아, 각 자릿수를 분할하여 서로 곱합니다.

그 곱한 수를 마찬가지로 각 자릿수로 분할해서 곱해 나갑니다.

이러한 과정을 계속해 나가면 1의 자리수에 도달하게 되는데 이렇게 곱해 나가는 반복 횟수를 고집수라 한다.

예)  $77 \rightarrow 49 \rightarrow 36 \rightarrow 18 \rightarrow 8$  : 4회 반복이므로 고집 수 = 4

예)  $679 \rightarrow 378 \rightarrow 168 \rightarrow 48 \rightarrow 32 \rightarrow 6$  : 5회 반복이므로 고집 수 = 5

예)  $868 \rightarrow 384 \rightarrow 96 \rightarrow 54 \rightarrow 20 \rightarrow 0$  : 5회 반복이므로 고집 수 = 5

정수의 범위 P1, P2( $100 \leq P1$ ,  $P2 \leq 10000$ )가 주어지고 고집수 N( $1 \leq N \leq 10$ )이 주어질 때 범위내의 고집수가 N인 수를 출력하는 프로그램을 작성하세요. 마지막 줄엔 총 개수를 출력한다.

(사용함수)

1. transNumber(num) : 양의 정수 num을 각 자릿수로 분할해서 곱한 결과를 리턴 하는 함수
2. inputUInt() : 양의 정수만 입력 받아 리턴 하는 함수(음수와 문자에 대한 입력 예외 처리 필수)
3. 결과 출력함수

**[HW30] 2진수 변환 프로그램**

10진수를 입력 받아 2진수로 출력하는 프로그램을 작성하시오.

Bit조절 연산자 및 반복문을 사용하며 음수는 2의 보수 상태로 출력 함.

(입출력 예)

\* 10진수 정수를 입력하시오 : 123

[illegible]

\*10진 정수를 입력하시오 : -1

$$-1(10) = 11111111111111111111111111111111(2)$$

### [HW31] 은행계좌 입출금 관리 프로그램

아래 실행 예가 완벽하게 수행되도록 은행계좌 입출금 프로그램을 작성하시오.

- 주 메뉴의 의미 : i-입금, o-출금, q-종료  
(주 메뉴 이외의 문자를 입력하면 재입력 요구)
- 시작할 때 잔액은 0원으로 한다.
- 잔액보다 인출액이 많으면 "잔액이 부족합니다."라고 메시지를 출력한 후 주 메뉴로 돌아감
- 입금액과 출금액 입력 시 음수 값 입력하면 "잘못 입력하셨습니다." 출력 후 금액 재입력 요구
- 입금액과 출금액 입력 시 문자 입력하면 "잘못 입력하셨습니다." 출력 후 금액 재입력 요구

(사용함수)

1. menu() : 메뉴 출력 후 사용자가 선택한 메뉴 문자를 리턴 하는 함수 메뉴 문자( i, o, q)외 문자 입력 시 재입력 요구
  2. inputInt(string) : 전달인자로 받은 문자열을 출력하고, 정수(음,양수)를 입력 받아 리턴 하는 함수(문자 입력 시 재입력 요구)
  3. deposit() : 입금처리 함수
  4. withdraw() : 출금처리 함수
- 그 외 필요하다고 판단되는 함수 추가 작성 가능

(실행 예)

\* 현재 잔액은 0원 입니다.

# 메뉴를 선택하시오(i-입금, o-출금, q-종료) : i(엔터)

# 입금액을 입력하세요 : 1500(엔터)

\* 현재 잔액은 1500원 입니다.

# 메뉴를 선택하시오(i-입금, o-출금, q-종료) : i(엔터)

# 입금액을 입력하세요 : 2500(엔터)

\* 현재 잔액은 4000원 입니다.

# 메뉴를 선택하시오(i-입금, o-출금, q-종료) : q(엔터)

\* 잘못 입력하셨습니다.

# 메뉴를 선택하시오(i-입금, o-출금, q-종료) : o(엔터)

# 출금액을 입력하세요 : 2400(엔터)

\* 현재 잔액은 1600원 입니다.

# 메뉴를 선택하시오(i-입금, o-출금, q-종료) : o(엔터)

# 출금액을 입력하세요 : 2000(엔터)

\* 잔액이 부족합니다.

\* 현재 잔액은 1600원 입니다.

# 메뉴를 선택하시오(i-입금, o-출금, q-종료) : i(엔터)

# 입금액을 입력하세요 : -3000(엔터)

\* 잘못 입력하셨습니다. 다시 입력하십시오 : s(엔터)

\* 잘못 입력하셨습니다. 다시 입력하십시오 : -5000(엔터)

\* 잘못 입력하셨습니다. 다시 입력하십시오 : e(엔터)

\* 잘못 입력하셨습니다. 다시 입력하십시오 : 3000(엔터)

\* 현재 잔액은 4600원 입니다.

# 메뉴를 선택하시오(i-입금, o-출금, q-종료) : o(엔터)

# 출금액을 입력하세요 : -2000(엔터)

\* 잘못 입력하셨습니다. 다시 입력하십시오 : v(엔터)

\* 잘못 입력하셨습니다. 다시 입력하십시오 : -1000(엔터)

\* 잘못 입력하셨습니다. 다시 입력하십시오 : t(엔터)

\* 잘못 입력하셨습니다. 다시 입력하십시오 : 1500(엔터)

\* 현재 잔액은 3100원 입니다.

### [HW32] 누승 함수 만들기

정수 값을 n승하는 함수와 실수 값을 n승하는 함수를 작성하시오. (반환 값은 n승의 결과)

단, 0의 어떤 승도 0이고, 어떤 수의 0승은 항상 1이며, 밑은 양수값으로 제한하며, 양수승만 구하는 것으로 제한 함.

(사용함수)

1. inputUInt(string) : 전달인자로 받은 문자열을 출력하고, 양의 정수만 입력 받아 리턴 하는 함수
2. inputDouble(string) : 전달인자로 받은 문자열을 출력하고, 양의 실수만 입력 받아 리턴 하는 함수
3. ipow(num, N)함수 : 양의 정수 num의 N승 값을 계산하여 리턴 하는 함수
4. fpow(fnum, N)함수 : 양의 실수 fnum의 N승 값을 계산하여 리턴 하는 함수

(실행 예)

\* 양의 정수 밑을 입력 하시오 : 2(엔터)

\* 양의 승을 입력 하시오 : 10(엔터)

2의 10승은 1024입니다.

\* 양의 실수 밑을 입력 하시오 : 3.4(엔터)

\* 양의 승을 입력 하시오 : 3(엔터)

3.40의 3승은 39.304 입니다. ◀ 실수형 밑은 소수점 이하 둘째 자리까지, 결과는 셋째 자리까지 출력

**[HW33] 환풍구 관리 프로그램**

[개발 주제] 환풍구 관리 시스템 개발

[주요 기능] 총 8개의 환풍구 개폐(ON, OFF) 관리 시스템 구현

[세부 기능]

1. 특정 환풍구 만을 ON(OPEN)하는 기능 구현
2. 특정 환풍구 만을 OFF(CLOSE)하는 기능 구현
3. 전체 환풍구 상태를 전환(반전)하는 기능 구현 : ON된 환풍구는 OFF상태로 OFF된 환풍구는 ON상태로 전환
4. 처음 시작 상태는 환풍구가 모두 닫혀있는 것으로 시작 함

[사용 데이터형]

1. 환풍구 ON, OFF 상태를 나타내기 위한 데이터 : 8bit로 구성된 데이터형 이용
2. 메뉴번호를 입력 받을 변수 선언 : 데이터형 제한 없음
3. 비트 연산을 위한 보조적인 변수 선언 : 1의 데이터형과 동일한 데이터형 사용 권장
4. 그 외 필요한 변수 선언 : 데이터형 제한 없음

[주요 사용 기술]

- Bit 조절 연산자 : &, |, ^, ~, <<, >>를 활용한 비트 조절 기술 구현
- 간단한 Number select를 이용한 User Interface 작성
- 잘못 입력된 데이터에 대한 재입력 기회 제공 (메뉴번호 1~4, FAN번호 1~8 이외의 값 입력 시 재입력 하도록 요구)

(사용함수)

1. menu()함수 : 4가지 메뉴를 한 줄에 출력하고 메뉴번호를 입력 받음  
이때 1~4 이외의 메뉴번호가 입력되거나 문자가 입력되면 재입력 요구
  2. openFan()함수 : 환풍구 열기 함수
  3. offFan()함수 : 환풍구 닫기 함수
  4. reverseFan() 함수 : 환풍구 On, Off 상태를 역으로 바꾸어주는 함수
  5. displayFan() 함수 : fan의 상태를 출력하는 함수
- 그 외 필요하다고 판단되는 함수 추가 작성 가능

[화면 구성]

1. 환풍구 열기 메뉴 실행 화면

## 1. 환풍구 열기 / 2. 환풍구 닫기 / 3. 환풍구 전체 전환 / 4. 종료 : 1

-----  
 Fan 열기 작업 실행 화면  
 -----

\* OPEN할 FAN 번호를 입력하시오(1-8) : 8  
 -----

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 8번FAN | 7번FAN | 6번FAN | 5번FAN | 4번FAN | 3번FAN | 2번FAN | 1번FAN |
| ON    | OFF   | OFF   | OFF   | OFF   | ON    | ON    | OFF   |

-----

\*\* 환풍구 프로그램 첫 화면에는 모든 FAN이 닫혀있는 것으로 시작 함

## 2. 환풍구 닫기 메뉴 실행 화면

## 1. 환풍구 열기 / 2. 환풍구 닫기 / 3. 환풍구 전체 전환 / 4. 종료 : 2

-----  
 Fan 닫기 작업 실행 화면  
 -----

\* CLOSE할 FAN 번호를 입력하시오(1-8) : 3  
 -----

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 8번FAN | 7번FAN | 6번FAN | 5번FAN | 4번FAN | 3번FAN | 2번FAN | 1번FAN |
| ON    | OFF   | OFF   | OFF   | OFF   | OFF   | ON    | OFF   |

-----

## 3. 환풍구 전체 전환 실행 화면

## 1. 환풍구 열기 / 2. 환풍구 닫기 / 3. 환풍구 전체 전환 / 4. 종료 : 3

-----  
 Fan 전체 전환 작업 실행 화면  
 -----

전체 FAN의 상태가 전환되었습니다. (ON, OFF 상태 뒤바뀜)  
 -----

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 8번FAN | 7번FAN | 6번FAN | 5번FAN | 4번FAN | 3번FAN | 2번FAN | 1번FAN |
| OFF   | ON    | ON    | ON    | ON    | ON    | OFF   | ON    |

-----



**[HW34] 수도요금 계산 프로그램 (switch~case문 사용)**

사용자 코드와 사용량을 입력 받아 수도요금을 계산하여 출력하자.

사용자 코드를 출력할 때 사용자 코드의 종류를 한글로 함께 출력하시오.

(사용함수)

1. 입력함수
2. 수도요금 계산함수
3. 결과 출력 함수

(수도요금 계산방식)

사용자 코드 : 1 - 가정용 (ton당 50원)

2 - 상업용 (45원)

3 - 공업용 (30원)

수도 사용요금 : 사용량 \* ton당 가격

총 수도요금 : 수도사용요금 + 세금(수도사용요금의 5%)

(입출력 예)

\* 사용자 코드를 입력하시오(1:가정용/2:상업용/3:공업용) : 1

\* 사용량을 입력하시오(ton단위) : 20

# 사용자코드 : 1(가정용)

# 사용량 : 20 ton

# 총수도요금 : 1050원

**[HW35] 총 5 개의 정수를 입력 받아 그 수의 합 출력하기**

정수는 반드시 양수 값 이어야 한다. 만약 0 이하의 수를 입력 받을 경우에는 입력으로 인정하지 않고 다시 입력 받도록 한다.

(사용함수) main()함수 하나로 처리. 단일 for문을 이용해서 프로그래밍 할 것.

(입출력)

0 보다 큰수를 입력하시오(1 번째) : 3

0 보다 큰수를 입력하시오(2 번째) : 7

0 보다 큰수를 입력하시오(3 번째) : 0

0 보다 큰수를 입력하시오(3 번째) : -7

0 보다 큰수를 입력하시오(3 번째) : 4

0 보다 큰수를 입력하시오(4 번째) : -5

0 보다 큰수를 입력하시오(4 번째) : 5

0 보다 큰수를 입력하시오(5 번째) : 12

입력된 값의 총 합 : 31

**[HW37]** 1~100 까지의 숫자를 출력할 때 3 의 배수는 '\*'로, 5 의 배수는 '#'으로 출력하고, 3 과 5 의 공배수는 정상숫자로 출력하는 프로그램을 작성하시오.

출력 시 한 줄에 숫자를 10 개씩 출력하세요

(사용함수) main()함수 하나로 처리

(실행 예)

```

1 2 * 4 # * 7 8 * #
11 * 13 14 15 16 17 * 19 #
 * 22 23 * # 26 * 28 29 30
...
91 92 * 94 # * 97 98 * #

```

[HW36] 피보나치 수열 구하기

1항부터 제 N항까지 피보나치 수열의 합을 계산하여 출력하는 프로그램 작성  
 피보나치 수열은 첫 번째 항과 두 번째 항을 더해서 세 번째 항을 만들고, 두 번째 항과 세 번째 항을 더해서 네 번째 항을 만드는 방법으로 계속해서 다음 항을 만들어가는 수열이다.

(수행 예)

```

피보나치 수열의 항수를 입력하시오 : 15
1 + 1 + 2 + 3 + 5 + 8 + 13 + 21 + 34 + 55 + 89 + 144 + 233 + 377 + 610 = 1596

```

[HW37] 황금동전

왕은 그의 충성스러운 기사에게 금화를 주기로 했다.

- 첫 째 날 기사는 금화 한 닢을 받았다.
- 2,3 일째는 금화 두 닢을 받았다.
- 4,5,6 일째는 금화 세 닢을 받았다.

```

...
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 ...

```

날 수가 주어질 때 첫 날부터 받은 총 금화의 수를 구하는 것이 문제이다.

(입출력 예)

```

* 기사의 근무일수를 입력하시오 : 10(엔터)
 근무일 : 10 일 / 총 금화 수 : 30 개

```

- test data

```

근무일 : 11 일 / 총 금화 수 35 개
근무일 : 16 일 / 총 금화 수 61 개
근무일 : 100 일 / 총 금화 수 945 개

```

**[HW38] 정해진 금액으로 물건구입하기**

입력 받은 현금 금액으로 동네 슈퍼에서 크림빵(500 원), 새우깡(700 원), 콜라(400 원)를 사려한다.  
잔돈을 하나도 남기지 않고 이 세가지 물건을 구입한다면 각각 몇 개씩 사야 하는가?  
물론 여러 가지 경우의 수가 있을 것이다. 어떠한 선택을 할 수 있는지 제시해보아라.  
단, 모든 품목을 한 개 이상은 꼭 구입하는 것으로 한다.

반복수행하고 금액 입력란에서 문자 입력 시 종료.

(입출력 예)

현재 당신의 소유 금액 입력 : 3500(엔터)

크림빵(1 개), 새우깡(2 봉지), 콜라(4 병)

크림빵(2 개), 새우깡(3 봉지), 콜라(1 병)

크림빵(4 개), 새우깡(1 봉지), 콜라(2 병)

어떻게 구입하시겠습니까?

현재 당신의 소유 금액 입력 : 4900(엔터)

크림빵(1 개), 새우깡(4 봉지), 콜라(4 병)

크림빵(2 개), 새우깡(1 봉지), 콜라(8 병)

크림빵(2 개), 새우깡(5 봉지), 콜라(1 병)

크림빵(3 개), 새우깡(2 봉지), 콜라(5 병)

크림빵(4 개), 새우깡(3 봉지), 콜라(2 병)

크림빵(6 개), 새우깡(1 봉지), 콜라(3 병)

어떻게 구입하시겠습니까?어떻게 구입하시겠습니까?

현재 당신의 소유 금액 입력 : 1600(엔터)

크림빵(1 개), 새우깡(1 봉지), 콜라(1 병)

어떻게 구입하시겠습니까?

현재 당신의 소유 금액 입력 : \$(엔터) ← 문자 입력 시 종료

**[HW39] 0부터 99까지의 정수를 한 줄에 10개씩 수직으로 출력하기**

(사용함수) main()함수 하나로 처리

(출력 예)

```
0 10 20 30 40 50 60 70 80 90
1 11 21 31 41 51 61 71 81 91
2 12 22 32 42 52 62 72 82 92
3 13 23 33 43 53 63 73 83 93
4 14 24 34 44 54 64 74 84 94
5 15 25 35 45 55 65 75 85 95
6 16 26 36 46 56 66 76 86 96
7 17 27 37 47 57 67 77 87 97
8 18 28 38 48 58 68 78 88 98
9 19 29 39 49 59 69 79 89 99
```

**[HW40] 대칭되는 별찍기**

라인 수를 입력 받아 2중 for문을 이용하여 다음과 같이 출력하시오.

반복수행하고 라인 수 입력란에서 문자 입력 시 종료.

단, 변수는 iterator변수 i, j와 라인수 저장변수 1개 총 3개만 사용하여 작성 할 것.

(사용함수) main()함수에서 작성

(입출력 예)

# 출력 라인수를 입력하시오 : 7

```
* *
** **
*** ***
**** ****
***** *****
***** *****
***** *****
***** *****
```

# 출력 라인수를 입력하시오 : @ ◀ 문자 입력 시 종료

**[HW41] 삼각 알파벳 출력하기**

영문자 대문자를 입력 받아 그 문자부터 'A' 문자까지 다음과 같이 출력하시오.

반복 수행하고 영문자 대문자 이외에 다른 문자 입력 시 종료하시오.

단, 변수는 iterator변수 i, j와 입력된 문자 저장변수 1개만 사용하여 총 3개의 변수만 사용할 것

(사용함수) main()함수에서 작성

\* 영문자 대문자 입력('A'~ 'Z') : F

F

FE

FED

FEDC

FEDCB

FEDCBA

\* 영문자 대문자 입력('A'~ 'Z') : # ◀ 영문자 대문자 외 다른 문자 입력 시 종료

**[HW42] 구구단 출력하기 (for문 사용)**

구구단 2~9단을 한 화면에 다음과 같이 출력하되 2중 for문을 2set를 이용해서 한번 출력하고

바로 이어서 3중 for문 1set만을 이용해서 똑같이 출력되도록 작성하시오.

(출력 예)

<2중 for문을 이용한 출력>

```
2*1= 2 3*1= 3 4*1= 4 5*1= 5
2*2= 4 3*2= 6 4*2= 8 5*2=10
2*3= 6 3*3= 9 4*3=12 5*3=15
```

|        |        |        |        |
|--------|--------|--------|--------|
| 2*4= 8 | 3*4=12 | 4*4=16 | 5*4=20 |
| 2*5=10 | 3*5=15 | 4*5=20 | 5*5=25 |
| 2*6=12 | 3*6=18 | 4*6=24 | 5*6=30 |
| 2*7=14 | 3*7=21 | 4*7=28 | 5*7=35 |
| 2*8=16 | 3*8=24 | 4*8=32 | 5*8=40 |
| 2*9=18 | 3*9=27 | 4*9=36 | 5*9=45 |

|        |        |        |        |
|--------|--------|--------|--------|
| 6*1= 6 | 7*1= 7 | 8*1= 8 | 9*1= 9 |
| 6*2=12 | 7*2=14 | 8*2=16 | 9*2=18 |
| 6*3=18 | 7*3=21 | 8*3=24 | 9*3=27 |
| 6*4=24 | 7*4=28 | 8*4=32 | 9*4=36 |
| 6*5=30 | 7*5=35 | 8*5=40 | 9*5=45 |
| 6*6=36 | 7*6=42 | 8*6=48 | 9*6=54 |
| 6*7=42 | 7*7=49 | 8*7=56 | 9*7=63 |
| 6*8=48 | 7*8=56 | 8*8=64 | 9*8=72 |
| 6*9=54 | 7*9=63 | 8*9=72 | 9*9=81 |

<3중 for문을 이용한 출력>

|        |        |        |        |
|--------|--------|--------|--------|
| 2*1= 2 | 3*1= 3 | 4*1= 4 | 5*1= 5 |
| 2*2= 4 | 3*2= 6 | 4*2= 8 | 5*2=10 |
| 2*3= 6 | 3*3= 9 | 4*3=12 | 5*3=15 |
| 2*4= 8 | 3*4=12 | 4*4=16 | 5*4=20 |
| 2*5=10 | 3*5=15 | 4*5=20 | 5*5=25 |
| 2*6=12 | 3*6=18 | 4*6=24 | 5*6=30 |
| 2*7=14 | 3*7=21 | 4*7=28 | 5*7=35 |
| 2*8=16 | 3*8=24 | 4*8=32 | 5*8=40 |
| 2*9=18 | 3*9=27 | 4*9=36 | 5*9=45 |

|        |        |        |        |
|--------|--------|--------|--------|
| 6*1= 6 | 7*1= 7 | 8*1= 8 | 9*1= 9 |
| 6*2=12 | 7*2=14 | 8*2=16 | 9*2=18 |
| 6*3=18 | 7*3=21 | 8*3=24 | 9*3=27 |
| 6*4=24 | 7*4=28 | 8*4=32 | 9*4=36 |
| 6*5=30 | 7*5=35 | 8*5=40 | 9*5=45 |
| 6*6=36 | 7*6=42 | 8*6=48 | 9*6=54 |
| 6*7=42 | 7*7=49 | 8*7=56 | 9*7=63 |
| 6*8=48 | 7*8=56 | 8*8=64 | 9*8=72 |
| 6*9=54 | 7*9=63 | 8*9=72 | 9*9=81 |

#### [HW43] 소수(prime number)출력하기

한 정수를 입력 받아 그 수까지의 모든 소수를 출력하는 프로그램을 작성하시오.  
단, 출력할 때 한 줄에 5개씩 출력하시오.

(사용함수) 입력함수, 소수판별 함수, 기타 필요하다고 판단되는 함수 추가 해도 됨

※ 소수판별 함수는 다음의 규칙대로 만들 것.

- 함수명 : primeNumber
- 전달인자 : int number - 소수여부를 판별할 정수 값

- 리턴 값 : int - 검사한 숫자가 소수이면 1, 소수가 아니면 0 리턴
- 기능 : 전달인자로 받은 숫자가 소수인지 아닌지 판별 하여 결과를 리턴  
ex) 전달인자가 11인 경우 11이 소수이면 1, 아니면 0을 리턴 함

(입출력 예)

\*정수값 하나를 입력하시오 : 20

1~20까지의 소수 값은 다음과 같습니다.

2    3    5    7    11

13   17   19

1~20까지의 총 소수는 8개 입니다.

(test data) 1~1000까지의 소수의 개수 → 168개

**[HW44] 키보드로부터** 시작 값과 끝 값을 입력 받고, 홀수를 구할 것인지 짝수를 구할 것인지를 입력 받아 두 값 사이에 존재하는 홀수나 짝수를 출력하고 그 합을 구하여 출력하시오.  
단, 이때 끝 값이 시작 값 보다 작거나 같을 경우에는 다시 입력 받으며, 짝홀수 구분 시 e 나 o 문자 이외의 문자 입력 시 재입력을 요구한다.

(사용함수)

1. 범위 입력함수 : 시작 값과 끝 값을 입력 받는 함수(call by pointer 기법 사용). 문자나 음수 입력 시 재입력 요구
2. 짝홀수 입력 함수 : e 또는 o 입력 이외의 다른 문자입력 시 재입력 요구
3. 결과 출력 함수

(실행 예 1)

# 시작 값을 입력하시오 : 10(엔터)

# 끝 값을 입력하시오 : 3(엔터)

# 끝 값을 입력하시오 : 5(엔터)

# 끝 값을 입력하시오 : 25(엔터)

\* 10~25까지의 짝수의 합을 구할까요? 홀수의 합을 구할까요?(짝수:e/홀수:o) : w(엔터)

\* 10~25까지의 짝수의 합을 구할까요? 홀수의 합을 구할까요?(짝수:e/홀수:o) : e(엔터)

10~25까지의 짝수(10 12 14 16 18 20 22 24)의 합은 136 입니다.

(실행 예 2)

# 시작값을 입력하시오 : 7(엔터)

# 끝값을 입력하시오 : 3(엔터)

# 끝값을 입력하시오 : 7(엔터)

# 끝값을 입력하시오 : 15(엔터)

\* 7~15까지의 짝수의 합을 구할까요? 홀수의 합을 구할까요?(짝수:e/홀수:o) : o(엔터)

7~15까지의 홀수(7 9 11 13 15)의 합은 55 입니다.

**[HW45] move box 프로그램**

우리가 통합환경에서 수행한 결과가 출력되는 결과 화면은  
 가로(x 좌표)로 1~80 개의 칸, 세로(y 좌표)로 1~25 라인으로 구성되어있습니다.  
 이 과제를 하기 위해서는 화면에서 출력위치를 조정하기 위한 커서 위치 조절 기법을 알아야 함.  
 커서 위치조절 함수는 다음의 gotoxy()함수를 사용하자.

```
#include <windows.h>
#include <stdio.h>
void gotoxy(int x, int y);
int main(void)
{
 gotoxy(5, 10); // x 좌표 5, y 좌표 10 위치로 커서 위치 이동
 printf("Hello, World~~~");
 return 0;
}
void gotoxy(int x, int y)
{
 COORD Pos = {x, y};
 SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
}
```

box 과제는 다음과 같이 수행되게 작성하세요.

sx, sy, ex, ey 좌표를 입력 받아 해당 좌표에 '\*'문자를 이용해서 속이 비어있는 box 를 출력한 후

'w','a','d','s'키를 이용해서 박스를 움직이게 합니다. (문자 입력 시에는 getch()함수를 이용)

'w' : 박스를 한줄 위로 이동

'a' : 박스를 왼쪽으로 한칸 이동

'd' : 박스를 오른쪽으로 한칸 이동

's' : 박스를 한줄 아래로 이동

ESC 키 입력 시 프로그램 종료 (getch()함수로 ESC 키 입력 시 인식되는 코드 값 : 27)

(좌표입력 실행화면 예)

sx 좌표 입력(1~70) : 30(엔터)

ex 좌표 입력(31~80) : 60(엔터) <<<=== 이때 sx 보다는 크고 80 이하의 값을 입력해야 함. 그 외값  
 입력 시 재 입력 요구.

sy 좌표 입력(1~20) : 10(엔터)

ey 좌표 입력(11~24) : 20(엔터)

입력 후 화면을 깨끗이 지운 후 ( system("cls"); 사용) 해당 좌표에 박스를 출력하고 키입력을 통해서  
 박스를 이동하기.

박스 이동시 화면 좌측끝까지 이동했는데 또 좌측으로 이동하기 위해 '4'를 입력하면 박스는  
 제자리에 있고 '뽕'소리만 낸다. (상하좌우 모두 같은 룰을 적용 함)

**[HW46] 요일 구하는 프로그램**

이용하여 년, 월, 일을 입력하면 요일을 출력하는 프로그램을 작성하시오.

반복 수행하고 년월일 입력 시 하나라도 문자가 들어오면 종료하시오.

오류 데이터가 입력되면 재입력을 요구할 것

(사용함수)

1. 년월일 입력함수
  2. 날짜 오류 체크함수 : 잘못된 날짜(예) 2013 13 7, 2011 2 29, 2016 -7 30 ...) 입력 시 재입력 요구
  3. 총 날수 구하는 함수 : 년, 월, 일을 전달인자로 받아 1900년 1월 1일부터 해당날짜까지의 총날수를 리턴 함
  4. 결과 출력 함수
- 그 외 필요하다고 판단되는 함수 추가 작성 가능

(요일 구하는 알고리즘)

1900.1.1~입력된 년, 월, 일까지의 총 일수를 구한 후

(총 일수 % 7) 의 결과 나머지가 0이면 일요일, 1이면 월요일 ... 6이면 토요일로 계산 함

(입출력 예)

\* 년 월 일을 입력하시오 : 2010 7 8

2010년 7월 8일은 목요일입니다.

(test data) 1968 2 22 목요일 / 2016 12 25 일요일 / 2017 5 5 금요일

**[HW47] 최소값, 최대값 찾기**

임의의 숫자 5 개를 키보드로부터 입력 받아서 배열에 저장한다.

저장된 숫자 중에서 가장 큰 값과 작은 값을 찾아 출력하는 프로그램을 작성한다.

일단 첫 번째 배열요소의 값이 가장 크다고(또는 작다고) 가정한 후에 이 값을 나머지 배열요소의 값들과 비교하면 된다. (Sort 하여 값을 찾지 말 것)

(사용함수)

1. 입력 함수 - 배열요소의 개수만큼 입력 받는 함수, 문자 입력 시 재입력 요구
2. 최대값 구하는 함수 - 배열 내의 값 중 최대값을 구하여 리턴
3. 최소값 구하는 함수 - 배열 내의 값 중 최소값을 구하여 리턴
4. 출력함수 - 결과 출력 함수

(입출력 예)

0 번 방 값 : -100

1 번 방 값 : 6.5

2 번 방 값 : 168

3 번 방 값 : 34

4 번 방 값 : 58.2

가장 큰 값 : 168.00

가장 작은 값 : -100.00



**[HW48] 숫자 개수 세기**

1 부터 20 까지의 숫자로만 초기화된 ary 배열을 다음과 같이 선언한다.

이 배열 내에 저장되어 있는 1 부터 20 까지 모든 숫자에 대해서 개수를 세어 출력하자.

각 숫자에 대하여 개수를 누적 시킬 변수들을 배열로 선언하여 작성한다.

(사용함수) main()함수 내에서 모두 처리

```
int ary[]={2,8,15,1,8,10,5,19,19,3,5,6,6,2,8,2,12,16,3,8,17,
 12,5,3,14,13,3,2,17,19,16,8,7,12,19,10,13,8,20,
 16,15,4,12,3,14,14,5,2,12,14,9,8,5,3,18,18,20,4};
```

```
int count[20]={0}; // 각 숫자의 개수를 누적 시킬 배열 (0 번 방은 1 의 개수 저장, 1 번 방은 2 의
개수 저장 ... 19 번 방은 20 의 개수 저장)
```

(출력 예)

```
1 - 1 개
2 - 5 개
3 - 6 개
:
19 - 4 개
20 - 2 개
```

**[HW49] 배열 내의 데이터를 역순으로 저장하기**

배열에 임의의 숫자를 초기화한 후에 각 숫자들의 위치를 반대로 바꾸는 프로그램을 작성한다.

배열은 하나만 사용하며 배열의 크기가 바뀌더라도 코드를 수정할 필요가 없도록 작성한다.

두 변수의 값을 바꾸기 위해서는 swap()함수를 구현하여 바꾸도록 하자.

(사용함수) swap() 함수

(출력 예)

```
처음 배열에 저장된 값 : 1 2 3 4 5
바뀐 배열에 저장된 값 : 5 4 3 2 1
```

**[HW50] 문자검색 프로그램(단일 검색)**

문자열 하나 문자 하나를 입력 받아 문자열중에서 문자가 포함되어 있는가를 검사하여 문자의 위치를 출력하는 프로그램을 작성하시오.

단, 문자열 내에 찾는 문자가 여러 개인 경우 첫 번째 위치만 리턴 함 (단일검색만 지원)

검색 작업을 반복하며 입력 문자열로 "end" 입력 시 종료하기

(제한조건) 문자열의 길이 L은 ( $1 \leq L \leq 99$ )의 범위로 제한 함

(사용함수)

1. 입력 함수 : 문자열, 문자 입력 함수
2. strcheck(char \*, char)함수 : 문자열과 문자를 전달인자로 받아 문자열 내의 문자위치를 검사하는

함수. 존재하지 않을 경우 -1을 리턴, 존재할 경우 문자의 index 리턴

(입출력 예)

# 문자열을 입력하시오 : family

# 문자를 입력하시오 : m

"family"문자열 안에 'm'문자는 2번 위치에 존재합니다.

# 문자열을 입력하시오 : goguma

# 문자를 입력하시오 : k

"goguma"문자열 안에 'k'문자는 존재하지 않습니다.

# 문자열을 입력하시오 : end    ← "end" 입력 시 종료

### [HW51] 전광판 프로그램

문자열을 입력 받아 char배열에 저장한 후 이 문자열을 전광판 형태로 출력하시오.

이때 저장한 문자열의 최대 길이는 50바이트로 제한 함.

(사용함수)

1. 문자열 입력함수
  2. 스크롤 처리 함수 : 문자열의 길이를 고려해서 문자열을 총 2회 rotate하여 출력
  3. Blink 처리 함수 : 문자열을 총 3회 점멸 시켜 출력
  4. 화면의 특정 영역만 지우는 함수
- 기타 필요하다고 판단되는 함수 추가 작성 가능

(출력방법)

1. 화면의 x 좌표 : 30, y 좌표 : 12 위치에 출력
2. 좌측으로 스크롤 하면서 출력. 이때 좌측으로 빠져나간 글자가 우측으로부터 이어지면서 출력  
(전체 문자열이 크게 2회 rotate 하는 형태로 수행)
3. 스크롤 출력이 끝난 후에는 지정된 좌표(30,12) 위치에서 3번 Blink(점멸) 시킴.

(입출력 예)

문자열을 입력하시오 : Happy Birthday!!!!(엔터)

|                     |             |
|---------------------|-------------|
| Happy Birthday!!!!  | ← 첫 출력      |
| appy Birthday!!!!H  | ← 1회 스크롤 후  |
| ppy Birthday!!!!Ha  | ← 2회 스크롤 후  |
| py Birthday!!!!Hap  | ← 3회 스크롤 후  |
| :                   |             |
| !Happy Birthday!!!! | ← 18회 스크롤 후 |
| Happy Birthday!!!!  | ← 19회 스크롤 후 |
| appy Birthday!!!!H  | ← 20회 스크롤 후 |
| ppy Birthday!!!!Ha  | ← 21회 스크롤 후 |
| :                   |             |
| Happy Birthday!!!!  | ← 38회 스크롤 후 |

위의 내용을 출력하고 지정된 좌표(30,12) 위치에서 3번 Blink(점멸) 하기

### [HW52] 정수 값 소트 프로그램 (1 차원 배열사용)

hw52\_sortData.txt 파일 내의 임의의 개수의 정수를 입력 받아 오름차순으로 정렬(sort)하는 프로그램을 작성하시오.

(사용함수)

1. 소트할 데이터 입력 함수 : 데이터 파일로부터 한 줄의 데이터를 읽어 들여 배열에 저장하는 함수
2. sort()함수 : 소트할 배열의 시작주소와 소트할 데이터의 개수를 전달인자로 받아 오름차순으로 소트 하는 함수
3. 배열 출력 함수 : 배열 내의 데이터를 index 순으로 출력하는 함수

(제한조건)

1. 파일의 첫 번째 줄에는 처리해야 할 테스트 총 건수
2. 두 번째~ 마지막 줄까지는 임의의 개수의 정수 값 저장 (한 줄의 끝은 0으로 표기 - 0은 실제 데이터가 아니므로 소트 대상에서 제외)
3. 수열데이터는 C:/data/(hw52\_sortData.txt 파일에 저장 되어 있다.

(hw52\_sortData.txt 파일 내용)

```
3 ← 총 테스트 건수
10 4 2 9 7 17 5 22 3 0
8 2 4 11 1 6 0
27 5 31 13 3 15 21 9 12 5 0
```

(출력 예)

```
소트 전 데이터 : 10 4 2 9 7 17 5 22 3
소트 후 데이터 : 2 3 4 5 7 9 10 17 22
```

```
소트 전 데이터 : 8 2 4 11 1 6
소트 후 데이터 : 1 2 4 6 8 11
```

```
소트 전 데이터 : 27 5 31 13 3 15 21 9 12 5
소트 후 데이터 : 3 5 9 12 13 15 21 27 31
```

### [HW53] 회문(Palindrome) 검사 프로그램

회문은 "level", "bob"과 같이 앞으로 읽으나 뒤로 읽으나 똑 같은 단어들을 말한다.

문자열이 회문인지 아닌지를 검사하는 함수를 작성하여 회문 검사 프로그램을 작성하자.

반복 수행하고 "end" 입력 시 종료.

(사용함수)

1. 입력함수 : 문자열 입력 함수

2. 회문 검사함수 : 회문이면 1 을 아니면 0 을 리턴 하는 함수. 단, "LevEl"과 같이 단어가 대,소문자로 섞여 있더라도 회문으로 인식되어야 함 (toupper() 또는 tolower() 시스템 라이브러리 함수 참조)

3. 결과 출력함수

그 외 필요하다고 판단되는 함수 추가 가능

(실행 예)

# 단어 입력 : level

"level" : 회문입니다!

# 단어 입력 : Bob

"Bob" : 회문입니다!

# 단어 입력 : apple

"apple" : 회문이 아닙니다!

# 단어 입력 : end    ← 반복 종료 조건

#### [HW54] 문자열 내의 숫자의 합 구하기

문자열을 입력 받아 문자열 내의 숫자들의 총 합을 계산한다. (문자열의 길이는 99 이하로 제한 함)

예를 들어서 입력 받은 문자열이 "ab123@5f#4\_31"이면 123+5+4+31 을 계산하여 163 을 출력해

준다. 단, 양수 값 만을 인식한다는 전제하에서 프로그래밍 해보자

반복 수행하고 "end" 입력 시 종료

(사용함수)

1. 입력함수 : 문자열 입력 함수

2. 숫자부분 계산 함수

3. 결과출력함수

그 외 필요하다고 판단되는 함수 추가 작성 가능

(힌트) 숫자로 변환 가능한 문자상수에서 '0'(zero 문자)를 빼면 숫자가 된다.

즉, '3'-'0' => 숫자상수 3 이 된다. 두 문자의 아스키코드 값 차이를 이용한 계산법임

(실행 예)

# 문장을 입력하시오 : ab123@5f#4\_31

"ab123@5f#4\_31" 내의 총 숫자는 [163]입니다.

# 문장을 입력하시오 : 7ma21^^0a%

"7ma21^^0a%" 내의 총 숫자는 [28]입니다.

# 문장을 입력하시오 : kingkong~~

"kingkong~~" 내의 총 숫자는 [0]입니다.

# 문장을 입력하시오 : end    ← 반복 종료 조건

**[HW55] 3 행 4 열짜리 2 차원 int 배열의 행, 열의 합을 구하기**

int 3행 4열짜리 num배열을 선언하여 각방의 값을 1~9 사이의 난수로 초기화 한 후 각 방의 내용 및 행의 합계, 열의 합계를 출력하는 프로그램을 작성하시오.

(사용함수)

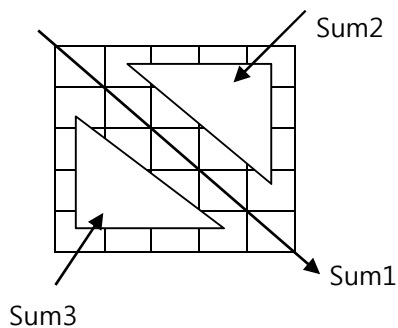
1. 난수로 배열을 초기화 시키는 함수
2. 배열의 내용과 행렬의 합계를 출력하는 함수

(출력 예)

|      |   |    |   |    |    |        |      |
|------|---|----|---|----|----|--------|------|
| 0행   | : | 8  | 3 | 9  | 4  | 0 행의 합 | : 24 |
| 1행   | : | 2  | 1 | 2  | 5  | 1 행의 합 | : 10 |
| 2행   | : | 6  | 3 | 9  | 2  | 2 행의 합 | : 20 |
| 열의 합 | : | 16 | 7 | 20 | 11 |        |      |

**[HW56] 배열의 부분합 구하기 (이차원배열 이용)**

int형 5행 5열짜리 2차원 배열을 선언하여 1~20까지의 난수(random number)를 발생시켜 초기화 한 후 다음의 각각의 합을 구하시오.



(사용함수)

1. 배열초기화 함수: 5 행 5 열짜리 배열을 난수를 발생시켜 행우선 순으로 초기화 하는 함수
  2. 배열의 각 부분합을 구하는 함수
  3. 결과 출력 함수
- 그 외 필요하다고 판단되는 함수 추가 작성 가능

(출력 예)

```

0번 행 : x x x x x
1번 행 : x x x x x
2번 행 : x x x x x
3번 행 : x x x x x
4번 행 : x x x x x

```

sum1 = xx

sum2 = xx

sum3 = xx

**[HW57] 문자열 첫 문자와 마지막 문자 출력하기**

다섯 개의 문자열을 입력 받아 char 5행 20열의 2차원 배열(배열명 : str)에 순서대로 저장한 후, 알파벳순서대로(a~z순으로) sort 하여 문자열 전체, 첫 문자, 마지막 문자를 출력하자.

(사용함수)

1. 2차원 char배열에 문자열 입력하는 함수
  2. 문자열 sort함수
  3. 2차원 char배열의 문자열, 첫 문자, 마지막 문자 출력 함수
- 그 외 필요하다고 판단되는 함수 추가 작성 가능

(참고자료) 문자열 소트를 위한 문자열 비교함수 ( strcmp()함수 )

- 원형 : int strcmp(const char \*s1, const char \*s2);
- 기능 : 문자열 s1과 문자열 s2를 비교한다.
- 사용 형태 : n = strcmp(s1, s2);
- 리턴 값 :
  - s1 > s2 : 양수값 리턴
  - s1 == s2 : 0리턴
  - s1 < s2 : 음수값 리턴

(입력 예)

```
1번 문자열을 입력하시오 : pear
2번 문자열을 입력하시오 : banana
3번 문자열을 입력하시오 : apple
4번 문자열을 입력하시오 : melon
5번 문자열을 입력하시오 : strawberry
```

(출력 예)

```
str[0] = apple a e
str[1] = banana b a
str[2] = melon m n
str[3] = pear p r
str[4] = strawberry s y
```

**[HW58] 히스토그램을 출력하기**

1 이상 9 이하의 정수를 N 개 입력 받아 아래 출력 예와 같이 히스토그램을 출력하는 프로그램을 작성하시오.

(제한조건)

1. 입력 받는 데이터의 개수 N 은 ( $2 \leq N \leq 30$ )의 범위로 입력. 입력 마지막에는 0 으로 입력 종료
2. "c:HW58\_histogram.txt" 파일의 내용을 히스토그램으로 출력 함

(hw58\_histogram.txt 파일 내용)

```
2 ← 총 테스트 건수
6 2 9 8 3 4 7 0
7 5 4 3 1 6 0
```

(사용함수)

1. 데이터를 배열에 저장하는 함수 : 파일내의 한 줄의 데이터를 배열에 저장하는 함수
2. 히스토그램 출력 함수

(출력 예)

```

 *
 * *
 * * *
 * * * *
 * * * *
 * * * * *
* * * * *
* * * * *
* * * * *
6 2 9 8 3 4 7

```

```

*
* *
* * *
* * * *
* * * * *
* * * * *
* * * * *
7 5 4 3 1 6

```

**[HW59] 서로 다른 수의 개수 출력하기**

n 개의 수를 입력으로 받아 서로 다른 수의 개수를 구하는 프로그램

(사용함수) main()함수 외에 가능하면 다양한 사용자 함수를 만들어서 작성 할 것

(처리조건)

입력의 첫 줄에는 숫자의 개수 N ( $2 \leq N \leq 1000$ )이 입력되고, 다음 줄에는 N 개의 정수가 입력 된다.

입력되는 수는 1 이상 10000 이하의 수이다.

그 수중 서로 다른 수의 개수를 출력한다.

반복 수행하고 입력할 숫자의 개수 란에 문자 입력 시 종료

(입출력 예)

\* 입력할 숫자의 개수 : 10(엔터)

\* 숫자 입력 : 1 2 3 4 5 4 3 2 1 2(엔터)

서로 다른 수의 개수 : 5 ◀ 위에 입력된 숫자 중 1,2,3,4,5 이렇게 5 개가 서로 다른 수가 됨

\* 입력할 숫자의 개수 : 7(엔터)

\* 숫자 입력 : 12 3 7 8 7 3 8(엔터)

서로 다른 수의 개수 : 4 ◀ 위에 입력된 숫자 중 12, 3, 7, 8 이렇게 4 개가 서로 다른 수가 됨

\* 입력할 숫자의 개수 : g(엔터) ◀ 문자 입력 시 종료

**[HW60] 주민등록번호 유효성 검사 프로그램**

1. 주민등록번호는 첫 6 자리는 태어난 날의 연도, 월, 일을 나타내며, YYMMDD 형태로 표시된다.

( 예를 들어 1982 년 12 월 20 일 출생 시 첫 6 자리 주민등록번호는 '821220' 이다)

2. 주민등록번호의 7 번째 숫자는 성별을 나타냄.

1900 년~1999 년에 태어난 경우 남자는 '1', 여자는 '2'이다.

2000 년~2099 년에 태어난 경우 남자는 '3', 여자는 '4'이다.

3. 주민등록번호의 마지막 13 번째 숫자는 이 주민번호가 유효한지 확인하는 인식자로 사용된다.

(실제 주민등록번호 생성원리 공개는 금지되어 있으므로, 인식자 생성원리는 임의로 정의한다)

4. 인식자 생성 원리 : 1 번째 자리부터 12 번째 자리까지의 각 숫자들을 더한 값을 10 으로 나눈 나머지가 마지막 13 번째 숫자가 됨.

(유효성 검사 예)

'6212201234564'는 1 번째부터 12 번째까지 숫자의 합이 34 ( $6+2+1+2+2+0+1+2+3+4+5+6 = 34$ )이며, 이를 10 으로 나눈 나머지는 4 이다. 이는 13 번째 숫자와 일치하므로 유효한 주민등록번호 이고, '7212201234567'은 13 번째 숫자가 5 가 아닌 7 이므로 유효하지 않은 주민등록번호이다.

5. 주민등록번호의 유효성 검사를 위해서 윤년인 경우도 고려해야 한다.

윤년이 평년과 다른 점은 2 월이 29 일까지 존재한다는 것이다. 윤년 계산 방법은 그레고리력 달력의 계산원리에 의해 다음과 같다



(윤평년계산 알고리즘)

- ① 년도를 4로 나누어 떨어지면, ②번 조건을 검사.  
   년도를 4로 나누어 떨어지지 않으면 평년.
- ② 년도를 100으로 나누어 떨어지면, ③번 조건을 검사.  
   년도를 100으로 나누어 떨어지지 않으면 윤년.
- ③ 년도를 400으로 나누어 떨어지면 윤년.  
   년도를 400으로 나누어 떨어지지 않으면 평년

1600년, 2000년은 윤년이며 1700년, 1800년, 1900년은 윤년이 아니다.

'0402291000008'은 1904년이 윤년이므로 2월 29일이 존재하기 때문에 유효한

주민번호이나 '9702292312412'는 1997년이 평년이므로 2월 29일이 존재하지 않기 때문에 유효하지 않은 주민번호이다.

(다음의 코드를 완성하시오)

// 필요한 헤더파일 include

#define TRUE 1

#define FALSE 0

#define RESIDENT\_NUMBER\_LENGTH 13 /\* 주민등록번호 길이\*/

int availabilityCheck(char \*resident\_number);

int checkLengthCharacter(char \*resident\_number);

int checkDate(char \*resident\_number);

int checkIdentification(char \*resident\_number);

int checkGender(char \*resident\_number);

int checkYear (int year);

int main()

{

/\* 테스트할 주민등록번호 저장 배열\*/

char resident\_number[][20]={"0402291000008", "870401102321", "00031541949179",  
                              "0003154194917", "801203#201122", "7804155328845", "7804150328840",  
                              "9612241068382", "9902292194322", "0230174326176", "8811391042219",  
                              "8100122042213", "8112002042213", "9210101069415", "0802294012345",  
                              "8806311069417", "8807311069418" };

int i, count;

/\* 검사할 주민등록번호의 개수 계산\*/

count = sizeof(resident\_number)/ sizeof(resident\_number[0]);

for(i=0; i<count; i++) /\* 주민등록번호 유효성 검사를 반복적으로 수행 함\*/

{

    if(availabilityCheck(resident\_number[i]) == TRUE)

    {

        printf("(+) 주민번호%s 는(은) 유효한 번호입니다.\n", resident\_number[i]);

    }

    else

    {

```

 printf("(-) 주민번호%s 는(은) 유효하지 않은 번호입니다.\n", resident_number[i]);
 }
}

return 0;
}

/*-----*/
availabilityCheck()함수: 주민등록번호 유효성 검사 함수
전달인자: 유효성 검사할 주민등록번호(문자열)
리턴값: 유효하면 TRUE, 유효하지 않으면 FALSE 리턴
-----*/
// TODO

/*-----*/
checkLengthCharacter()함수: 주민등록번호 길이 및 문자 유효성검사 함수
전달인자: 검사할 주민등록번호(문자열)
리턴값: 주민등록번호의 길이가 맞고 숫자문자로만 구성되어 있으면 TRUE,
 길이가 짧거나 길고, 숫자 문자가 아닌 문자가 섞여 있으면 FALSE 리턴
-----*/
// TODO

/*-----*/
checkDate()함수: 첫 6 자리(연,월,일)의 유효성 검사 함수
전달인자: 유효성 검사할 주민등록번호(문자열)
리턴값: 유효한 날짜이면 TRUE, 유효하지 않은 날짜이면 FALSE 리턴
-----*/
// TODO

/*-----*/
checkGender()함수: 7 번째 자리의 성별식별번호 유효성 검사함수
전달인자: 유효성 검사할 주민등록번호(문자열)
리턴값: 성별식별번호가 '1'~'4'이면 TRUE, 그 외 숫자 문자이면 FALSE 리턴
-----*/
// TODO

/*-----*/
checkIdentification()함수: 주민등록번호 끝자리(인식자) 유효성 검사 함수
전달인자: 유효성 검사할 주민등록번호(문자열)
리턴값: 유효한 날짜이면 TRUE, 유효하지 않은 날짜이면 FALSE 리턴
-----*/
// TODO

/*-----*/
checkYear ()함수: 년도의 윤,평년 여부 검사
전달인자: 윤,평년 검사할 년도
리턴값: 윤년이면 TRUE, 평년이면 FALSE 리턴
-----*/
// TODO

```

(출력 예)

- (+) 주민번호0402291000008는(은) 유효한 번호입니다.
- (-) 주민번호870401102321는(은) 유효하지 않은 번호입니다.
- (-) 주민번호00031541949179는(은) 유효하지 않은 번호입니다.
- (+) 주민번호0003154194917는(은) 유효한 번호입니다.
- (-) 주민번호801203#201122는(은) 유효하지 않은 번호입니다.
- (-) 주민번호7804155328845는(은) 유효하지 않은 번호입니다.
- (-) 주민번호7804150328840는(은) 유효하지 않은 번호입니다.
- (-) 주민번호9612241068382는(은) 유효하지 않은 번호입니다.
- (-) 주민번호9902292194322는(은) 유효하지 않은 번호입니다.
- (-) 주민번호0230174326176는(은) 유효하지 않은 번호입니다.
- (-) 주민번호8811391042219는(은) 유효하지 않은 번호입니다.
- (-) 주민번호8100122042213는(은) 유효하지 않은 번호입니다.
- (-) 주민번호8112002042213는(은) 유효하지 않은 번호입니다.
- (-) 주민번호9210101069415는(은) 유효하지 않은 번호입니다.
- (+) 주민번호0802294012345는(은) 유효한 번호입니다.
- (-) 주민번호8806311069417는(은) 유효하지 않은 번호입니다.
- (+) 주민번호8807311069418는(은) 유효한 번호입니다.

#### [HW61] 년월변경 달력 프로그램

- 달력 첫 화면은 2016 년 7 월 달력부터 시작해서 다음과 같은 키로 년월변경된 달력을 출력하는 프로그램을 작성하시오. 달력 출력 시 일요일은 밝은 빨간색, 토요일은 밝은 파란색, 평일은 밝은 회색으로 출력하자. (키 입력은 getch()함수 사용 - ESC key(Code : 27) : 종료)
- 위 화살표('w') key : 전년도 같은 월
- 아래 화살표('s') key : 다음 년도 같은 월
- 좌측 화살표('a') key : 전월 (1 월 상태였을 경우 전년도 12 월로 변경)
- 우측 화살표('d') key : 다음 월 (12 월 상태였을 경우 다음 년도 1 월로 변경)

| [ 2016 . 7 ] |     |     |     |     |     |     |
|--------------|-----|-----|-----|-----|-----|-----|
| SUN          | MON | TUE | WED | THU | FRI | SAT |
|              |     |     |     |     | 1   | 2   |
| 3            | 4   | 5   | 6   | 7   | 8   | 9   |
| 10           | 11  | 12  | 13  | 14  | 15  | 16  |
| 17           | 18  | 19  | 20  | 21  | 22  | 23  |
| 24           | 25  | 26  | 27  | 28  | 29  | 30  |
| 31           |     |     |     |     |     |     |

w : 전 년도 / s : 다음 년도 / a : 전 월 / d : 다음 월 / ESC : 종료

(사용함수) 총 날수 및 요일 구하는 함수, 한달치 달력 출력 함수, 년월변경 처리 함수 등

(color출력 예제)

```
#define BLACK 0
#define BLUE 1
#define GREEN 2
```

```

#define CYAN 3
#define RED 4
#define MAGENTA 5
#define BROWN 6
#define LIGHTGRAY 7
#define DARKGRAY 8
#define LIGHTBLUE 9
#define LIGHTGREEN 10
#define LIGHTCYAN 11
#define LIGHTRED 12
#define LIGHTMAGENTA 13
#define YELLOW 14
#define WHITE 15
int main()
{
 textcolor(LIGHTRED, BLACK);
 printf("MT가자~\n");
 textcolor(YELLOW, BLUE);
 printf("그래 가자~\n");
 return 0;
}
void textcolor(int foreground, int background)
{
 int color=foreground+background*16;
 SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), color);
}

```

### [HW62] 파리 배열

배열의 행과 열의 크기 N을 입력 받아 아래 와 같이 형태로 데이터를 저장한 후

0행부터 마지막 행까지의 데이터 전체를 출력하는 프로그램 작성

(5행 5열인 경우 데이터 저장 모양)

|    |    |    |    |   |
|----|----|----|----|---|
| 1  | 2  | 3  | 4  | 5 |
| 16 | 17 | 18 | 19 | 6 |
| 15 | 24 | 25 | 20 | 7 |
| 14 | 23 | 22 | 21 | 8 |
| 13 | 12 | 11 | 10 | 9 |

(6행 6열인 경우 데이터 저장 모양)

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  |
| 20 | 21 | 22 | 23 | 24 | 7  |
| 19 | 32 | 33 | 34 | 25 | 8  |
| 18 | 31 | 36 | 35 | 26 | 9  |
| 17 | 30 | 29 | 28 | 27 | 10 |
| 16 | 15 | 14 | 13 | 12 | 11 |

(입출력 예)

# 행/열의 수 입력 : 5(엔터)

```
1 2 3 4 5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
```

# 행/열의 수 입력 : 6(엔터)

```
1 2 3 4 5 6
20 21 22 23 24 7
19 32 33 34 25 8
18 31 36 35 26 9
17 30 29 28 27 10
16 15 14 13 12 11
```

### [HW63] Jolly jumper

$n$  개의 수로 이루어진 수열이 주어질 때 인접한 수의 차이가 1 부터  $n-1$  까지의 수를 만들 수 있을 때 이를 Jolly jumper 라 한다. 단 차례대로 1 부터 만들어 질 필요는 없다.

예를 들어 아래와 같이 4 개의 수로 이루어지는 수열이 주어질 때

1 4 3 1 이 수열은 인접한 수의 차가 3 1 2 이므로 Jolly jumper 이다.

1 4 2 3 도 인접한 수의 차가 3 2 1 이므로 Jolly jumper 이다.

수가 하나만 있는 경우는 Jolly jumper 로 간주한다.

수열이 Jolly jumper 인지 아닌지를 구하는 프로그램을 작성하시오.

(제한조건)

1. hw63\_jolly.txt 파일의 내용을 읽어 들여 처리 함
2. 첫째 번 줄에 총 테스트 건수
3. 둘째 줄의 첫 번째 데이터는 숫자의 개수  $N$  이 ( $0 \leq N \leq 100$ )의 범위로 주어진다.
4. 둘째 줄의 두 번째부터 마지막 숫자까지는 검사할 수열이 주어진다.

(hw63\_jolly.txt 파일 내용)

```
5 ← 총 테스트 건수
4 1 4 2 3 ← 첫 번째 숫자는 수열내의 숫자의 개수, 두 번째~마지막은 검사할 수열
5 1 4 2 -1 6
8 77 80 84 79 78 76 82 89
6 -9 -5 -3 -2 1 4
6 888 886 887 892 889 885
```

(출력 예)

검사한 수열 : 1 4 2 3 (Jolly jumper)

검사한 수열 : 1 4 2 -1 6 (Not jolly jumper)

검사한 수열 : 77 80 84 79 78 76 82 89 (Jolly jumper)

검사한 수열 : -9 -5 -3 -2 1 4 (Not jolly jumper)

검사한 수열 : 888 886 887 892 889 885 (Jolly jumper)

**[HW64] 야구게임 만들기**

여행 갈 때 기차안에서 친구들과 함께 즐겼던 재미있는 야구게임을 기억하시나요?

야구게임은 상대방의 숫자를 알아 맞추는 게임입니다.

컴퓨터가 사용자 몰래 정한 4자리의 숫자를 알아 맞추는 게임을 작성해보자.

(세부기능)

1. 컴퓨터가 발생시킨 4자리의 숫자는 서로 중복되지 않아야 한다.
2. 컴퓨터가 발생시킨 4자리의 숫자는 화면에 출력해준다.
3. 사용자는 컴퓨터가 발생시킨 4자리의 숫자를 맞출 때까지 게임은 계속 진행된다.
4. 야구게임에서의 볼카운트는 다음의 규칙으로 정해진다.
  - 숫자와 자리가 일치 하면 :strike
  - 숫자는 맞는데 자리가 일치하지 않으면 : ball
  - 일치하는 숫자가 하나도 존재하지 않으면 : No! 라고 출력
  - 4자리의 숫자와 자리가 정확히 일치하면 : OK!!! 출력 후 프로그램 종료

(출력화면) 예를 들어 컴퓨터가 발생한 난수가 ( 7 4 2 5 ) 일 경우

# 1차 : 7 6 5 4 1S 2B

# 2차 : 1 3 8 9 No!

:

# n차 : 7 4 2 5 OK!!! (네 자리의 숫자를 맞출 때까지 반복한다.)

(사용함수)

1. 중복되지 않는 난수로 초기화 하는 함수
  2. 볼카운트 체크 함수
  3. 결과출력함수
- 기타 필요하다고 판단되는 함수로 나누어 작성할 것

**[HW65] string 제어 함수 만들기**

다음의 함수를 만든 후 (각각의 기능은 표준 library로 제공되는 string제어 함수와 같음)

기능을 테스트하는 프로그램을 작성하여 각 기능이 정확히 수행되는지 확인 하시오.

1. char \* strchr(char \*str, int ch) : str 문자열 내에 문자 ch 가 있으면 찾은 위치의 주소를 리턴/못찾으면 NULL pointer 반환
2. char \* strcat(char \*s1, char \*s2) : s1 문자열 뒤에 s2 문자열을 덧붙인 후 s1 의 시작주소 리턴
3. char \* strstr(char \*s1, char \*s2) : s1 문자열 내에 s2 문자열이 부분 문자열로 존재하는 검사. 찾으면 부분 문자열의 시작주소 리턴/못찾으면 NULL pointer 리턴
4. void strcpy(char \*s1, char \*s2) : s1 주소에 s2 문자열을 복사(s1 주소 기억공간이 부족해서 오류가 발생할 수 있음), 리턴값 없음
5. unsigned int strlen(char \*str) : str 문자열의 길이를 구하여 리턴 함. 'W0'(NULL 문자)는 길이에 포함되지 않음
6. int strcmp(char \*s1, char \*s2) : s1, s2 두 문자열의 문자를 순차적으로 비교하여 같으면 0, s1 이 크면 1, s1 이 작으면 -1 리턴

**[HW66] 가변 배열 만들기**

2차원 char 포인터 배열( char \*p[5] )을 선언하여 문자열을 5개 입력 받아 다음과 같은 가변배열을 만들어 문자열을 저장하고, 오름차순으로 정렬하여 출력하자.

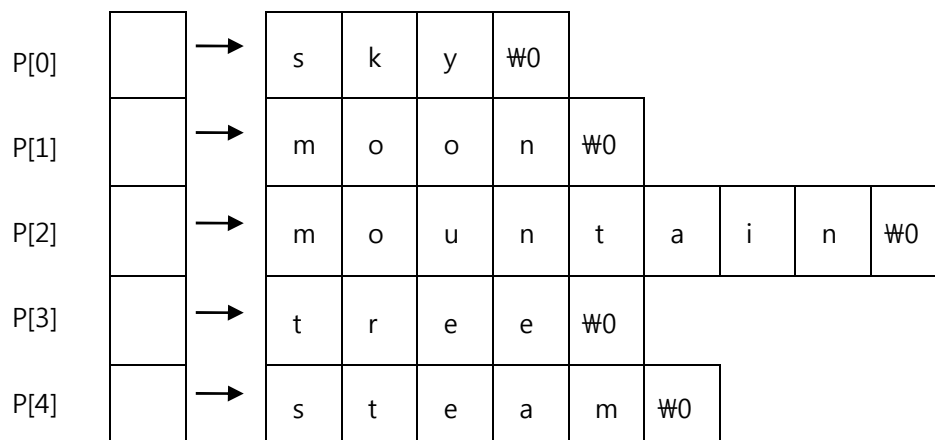
(입력화면)

```
문자열 1 : sky
문자열 2 : moon
문자열 3 : mountain
문자열 4 : tree
문자열 5 : stream
```

(출력화면)

```
1. moon
2. mountain
3. sky
4. stream
5. tree
```

(가변배열 생성)



가변배열의 각행의 길이는 입력 받은 문자열의 길이를 구하여 그 길이+1만큼의 길이로 동적메모리 할당을 해준다.

**[HW67] 성적관리 프로그램**

학생수를 입력 받아 다음의 내용을 처리한다.

1. 학생 성명과 세 과목 성적을 저장할 기억공간은 동적메모리(2 차원 배열의 형태) 할당하여 사용한다.
2. 학생 성명을 저장하는 기억공간은 실제 이름의 길이에 맞춰 할당 후 저장 한다.
3. 석차를 매길 때 총점기준으로 동점자는 동일한 석차를 부여하며 1 등이 3 명인 경우는 1 등의 다음 등수는 4 등으로 처리한다.
4. 성적 입력 시 0~100 까지의 점수만 가능토록 한다. 음수 또는 100 을 초과하는 성적 및 문자 입력 시에는 재입력을 요구한다.

(기능) 성적관리 프로그램은 개인성적을 모두 입력 받은 후 3 개의 메뉴로 관리한다.(성적 수정, 전체성적 출력, 종료)

1. 개인성적 입력 기능 : 학생 성명, 국어, 영어, 수학 세 과목의 성적을 입력 받는다.  
학생수 만큼 모두 입력 받은 후 주 메뉴를 출력한다.
2. 성적 수정 메뉴의 기능 : 국어, 영어, 수학 세 과목의 성적을 수정 가능토록 한다.  
반복적으로 수정이 가능하도록 하되 2 번 메뉴의 종료 조건은 프로그래머가 정해주어 사용자가 알 수 있도록 화면에 안내해준다.
3. 전체성적 출력 메뉴의 기능 : 총점 순으로 소트하여 석차를 매긴 후  
[석차 성명 국어 영어 수학 총점 평균 ]의 순으로 한 줄에 한 명씩 성적을 출력한다.

위의 기능을 수행하는 성적관리 프로그램을 작성할 때 화면디자인 및 세부 기능은 본인의 의사에 따라 결정하여 진행한다.

단, 프로그램을 사용자의 편의를 최대한 고려하여 작성하시오.



**[HW68] 빙고게임 만들기**

컴퓨터와 사용자의 대전용 빙고게임 구현

(세부 기능)

1. 개인연습용 빙고게임과 컴퓨터와 대전할 수 있는 빙고게임을 구현한다.
2. 빙고게임의 가로세로 크기를 입력 받아 빙고게임에 사용되는 배열은 `int [N][N]`; 형태의 2차원 배열을 동적메모리 할당 하여 사용하며 중복되지 않도록  $1 \sim (N*N)$ 의 숫자로 배열을 채우고 게임을 시작한다.  
 \*\* 이때 N값은 양수이어야 함.
3. 빙고게임에서 가로 N줄, 세로 N줄, 대각선 2줄 중에서 N개 이상의 줄이 먼저 지워지는 쪽이 승리하는 것으로 처리하되 컴퓨터와 사용자가 동시에 N개 이상 지워질 경우에는 무승부로 처리한다.
4. 게임 진행 중 이미 지워진 숫자를 입력하거나  $1 \sim (N*N)$ 의 숫자가 아닌 값을 입력할 경우에는 재입력을 요구한다.
5. 기타 부수적인 내용은 개발자가 상황을 고려해서 작성한다.

(사용 데이터형)

- extern 변수나 배열을 사용하지 않음
- 그 외 필요한 변수 선언 : 데이터형 제한 없음

(주요 사용 기술)

- 간단한 Number select 메뉴를 이용한 User Interface 작성
- 빙고판의 내용 설정 시 난수 발생 기법 사용
- 프로그램 작성시 함수화 시킬 수 있는 부분은 최대한 함수화 시키시오.  
 예) 빙고판에 숫자를 세팅하는 함수  
 빙고판에 지워진 라인수 count하는 함수  
 이미 지워진 난수 체크하는 함수  
 메뉴 출력하고 메뉴번호 입력하는 함수 등...

(실행 예)

1. 초기메뉴 화면

```
1. 연습게임(개인 연습용)
2. 대전게임(컴퓨터와 대전용)
3. 종료

메뉴선택 : 1(엔터)
```

## 2. (1번) 연습게임 메뉴선택 시 화면 (다음은 5행 5열 빙고판을 기준으로 설명 함)

연습용 빙고게임을 시작합니다.

# 빙고판의 가로,세로 크기를 입력해주세요(양수값 입력) : 5(엔터)

|    |    |    |    |    |
|----|----|----|----|----|
| 9  | 10 | 20 | 3  | 1  |
| 8  | 21 | 7  | 15 | 19 |
| 22 | 2  | 13 | 25 | 24 |
| 4  | 11 | 18 | 5  | 14 |
| 12 | 6  | 16 | 23 | 17 |

<=== 빙고게임판의 내용은 1~25까지의 숫자를 난수로 발생시켜서 출력해준다.

# 지울 숫자 입력(1~25) : 1(엔터)

|    |    |    |    |    |
|----|----|----|----|----|
| 9  | 10 | 20 | 3  | X  |
| 8  | 21 | 7  | 15 | 19 |
| 22 | 2  | 13 | 25 | 24 |
| 4  | 11 | 18 | 5  | 14 |
| 12 | 6  | 16 | 23 | 17 |

<=== 입력되어서 삭제된 숫자란의 내용은 'X'문자로 표기

# 지울 숫자 입력(1~25) : 1(엔터)

\* 이미 지워진 숫자 입니다. 다시 입력하세요.

# 지울 숫자 입력(1~25) : 23(엔터)

|    |    |    |    |    |
|----|----|----|----|----|
| 9  | 10 | 20 | 3  | X  |
| 8  | 21 | 7  | 15 | 19 |
| 22 | 2  | 13 | 25 | 24 |
| 4  | 11 | 18 | 5  | 14 |
| 12 | 6  | 16 | X  | 17 |

(위의 작업을 5줄이 지워질 때까지 반복한 후 연습게임이 끝나고 아무 키나 치면 주메뉴로 돌아감)

## 3. (2번) 대전게임 메뉴 선택 시 화면

사용자:컴퓨터 대전 빙고게임을 시작합니다.

# 빙고판의 가로,세로 크기를 입력해주세요(양수값 입력) : 5(엔터)

사용자 빙고게임판 내용을 생성중입니다.

컴퓨터 빙고판 내용을 생성중 입니다.

(모든 생성이 끝나고 나면 아래와 같이 출력한다)

[user]

[computer]

|    |    |    |    |    |  |   |   |   |   |   |
|----|----|----|----|----|--|---|---|---|---|---|
| 9  | 10 | 20 | 3  | 1  |  | ? | ? | ? | ? | ? |
| 8  | 21 | 7  | 15 | 19 |  | ? | ? | ? | ? | ? |
| 22 | 2  | 13 | 25 | 24 |  | ? | ? | ? | ? | ? |
| 4  | 11 | 18 | 5  | 14 |  | ? | ? | ? | ? | ? |
| 12 | 6  | 16 | 23 | 17 |  | ? | ? | ? | ? | ? |

<<=== 사용자의 숫자는  
값으로 출력하고 컴퓨터  
의 숫자는 ?로 출력

# 지운 숫자 입력(1~25) : 23(엔터) <<=== 이미 지운 숫자 입력시 재입력 요구

| [user] |    |    |    |    | [computer] |   |   |   |   |   |
|--------|----|----|----|----|------------|---|---|---|---|---|
| 9      | 10 | 20 | 3  | 1  |            | ? | ? | ? | ? | ? |
| 8      | 21 | 7  | 15 | 19 |            | ? | ? | ? | ? | ? |
| 22     | 2  | 13 | 25 | 24 |            | ? | ? | X | ? | ? |
| 4      | 11 | 18 | 5  | 14 |            | ? | ? | ? | ? | ? |
| 12     | 6  | 16 | X  | 17 |            | ? | ? | ? | ? | ? |

<<=== 사용자가 입력한  
숫자를 지움

# 컴퓨터가 선택한 숫자는 (23)입니다. <<=== 컴퓨터의 숫자는 난수를 발생시켜서 처리함.  
이때 이미 지운 숫자가 난수로 발생하면 다시 난수를 발생시켜야 함

# 컴퓨터가 선택한 숫자는 (9)입니다.

| [user] |    |    |    |    | [computer] |   |   |   |   |   |
|--------|----|----|----|----|------------|---|---|---|---|---|
| X      | 10 | 20 | 3  | 1  |            | ? | ? | X | ? | ? |
| 8      | 21 | 7  | 15 | 19 |            | ? | ? | ? | ? | ? |
| 22     | 2  | 13 | 25 | 24 |            | ? | ? | X | ? | ? |
| 4      | 11 | 18 | 5  | 14 |            | ? | ? | ? | ? | ? |
| 12     | 6  | 16 | X  | 17 |            | ? | ? | ? | ? | ? |

<<=== 컴퓨터가 발생시킨  
숫자를 지움

(위의 작업을 반복적으로 수행하며 사용자나 컴퓨터 둘 중 어느 한쪽이라도 먼저 N줄 이상이  
지워지면 게임종료)

:

:

# 사용자 승! (또는 컴퓨터 승! 또는 사용자, 컴퓨터 무승부! 의 형태로 게임 최종결과  
출력하기)

# 아무 키나 치면 주 메뉴로 돌아갑니다.

**[HW69] 시계출력하기**

화면 우측 하단에 다음의 내용을 참고하여 시계(00:00:00)를 출력하시오.

- 시계는 10 초간 시간을 보여준 후 프로그램을 종료
- 시간은 1 초에 1 회씩만 출력할 것 (Sleep()함수 또는 delay()함수 사용하지 말 것)

(사용함수)

- timer() : 이미 출력한 시간은 출력하지 않아야 함.  
시간을 출력했으면 1, 출력하지 않았으면 0을 리턴 함

**[HW70] 사전 프로그램 (struct 배열 이용)**

다음과 같은 구조체를 선언하여 단어와 뜻을 입력 받아 5가지 메뉴(입력하기, 출력하기, 검색하기, 삭제하기, 종료)로 처리하는 프로그램을 작성하시오.

단어의 길이는 19문자 이하이며, 단어의 뜻은 79문자 이하로 입력된다고 가정한다.

사전에 저장 가능한 최대 단어의 개수는 10개이며, 단어 입력 및 삭제를 반복하더라도 10개를 초과하지 않도록 해야 한다.

(데이터 저장을 위한 구조체 사용)

```
struct Dic{
 char word[20]; // 단어를 저장하는 멤버
 char mean[80]; // 단어의 뜻을 저장하는 멤버
 int len; // 단어의 길이를 저장하는 멤버
};
struct Dic ary[10]; <- 이곳에 단어와 뜻, 단어의 길이를 저장함
```

(입력하기 메뉴 선택 시) 단어와 뜻을 반복적으로 입력 받아 저장하다가 단어 입력란에서 "end"입력하면 입력메뉴기능을 종료하고 주 메뉴로 돌아감(입력한 단어의 개수가 10개를 초과하지 않도록 해야 함)

# 단어를 입력하시오 : king

# 뜻을 입력하시오 : king is king

# 단어를 입력하시오 : queen

# 뜻을 입력하시오 : queen is queen

# 단어를 입력하시오 : end      <- 종료조건 (뜻을 입력란에서 end를 입력해도 종료 함)

(출력하기 메뉴 선택 시) 단어기준으로 알파벳 오름차순으로 소트하여 출력함.

출력 시 단어의 뜻이 길 경우 50자까지만 출력하고 뒤에 생략기호(~)를 붙여서 출력한다.  
모두 출력하고 잠시 멈춘 후 아무 키나 입력하면 주 메뉴로 돌아감.

1.                   apple( 5) : apple is apple
2.                   gogumi( 6) : gogumida! gogumida!!
- :

(검색하기 메뉴 선택 시) 찾는 단어가 발견되면 그 단어의 뜻을 출력하고 없을 시에는 "Not found!!!" 메시지 출력하기. 반복적으로 검색하다가 검색할 단어 입력란에서 "end"입력하면 검색메뉴기능을 종료하고 주 메뉴로 돌아감

# 찾을 단어를 입력하시오 : queen  
단어의 뜻 : queen is queen

# 찾을 단어를 입력하시오 : prince  
Not found!!!

# 찾을 단어를 입력하시오 : end    <= 종료조건

(삭제하기 메뉴 선택 시) 입력된 단어를 삭제하고 없는 단어의 경우 "없는 단어입니다."라는 메시지 출력하기. 반복적으로 삭제하다가 삭제할 단어 입력란에서 "end"입력하면 삭제메뉴기능을 종료하고 주 메뉴로 돌아감.

# 삭제할 단어를 입력하시오 : queen  
# 정말로 삭제하시겠습니까?(y/n) : n    ◀ y 입력 시 삭제, n과 그 외 문자 입력 시 삭제취소  
삭제가 취소되었습니다.

# 삭제할 단어를 입력하시오 : king  
# 정말로 삭제하시겠습니까?(y/n) : y    ◀ y 입력 시 삭제, n과 그 외 문자 입력 시 삭제취소  
삭제되었습니다.

# 삭제할 단어를 입력하시오 : prince  
Not found!!!

# 삭제할 단어를 입력하시오 : end    <= 종료조건

**[HW71] 포인트 끝말잇기 게임 만들기**

우리가 일반적으로 알고 있는 끝말잇기 게임을 약간 변형하여 재미있는 포인트 끝말잇기 게임을 만들어봅시다.

(기능 명세)

1. 점수로 계산될 5개의 포인트 단어를 hw71\_pointWord.txt파일로 부터 입력 받아 오름차순으로 데이터를 소트하여 출력한다.

Hw71\_pointWord.txt 내용

```
tiger
hen
cow
rabbit
lion
```

2. 끝말잇기를 하는 중에 포인트 단어로 끝말을 잇게 되면 해당하는 포인트 단어가 지워지며 포인트 단어 하나가 지워질 때마다 20점의 점수를 얻게 된다.(5개의 포인트 단어를 모두 맞추면 100점을 얻게 되고 게임은 종료 됨)

3. 끝말잇기 게임은 1게임당 총 10회의 입력만 허용한다.

이때 끝말잇기 게임의 룰에 따라 이전 글자의 마지막 글자로 시작하는 단어를 적어준다.  
만일 끝말잇기가 안 되는 단어를 입력했을 경우에는 "잘못 입력하셨습니다"라는 메시지를 출력한 후 재입력을 요구한다.

4. 시작단어는 "pointer"로 시작합니다.

5. 기타 자세한 실행 결과는 아래 실행 예를 참조하여서 작성한다.

(실행 예)

# 포인트 단어를 파일로부터 입력 받는 중입니다.... (메시지를 띄워주고 파일로부터 입력 받을 것)

\* 포인트단어 : cow / hen / lion / rabbit / tiger / ◀ 파일로부터 입력 받은 포인트 단어를 화면에 출력

\* 사용자 입력 단어 : pointer /

끝말잇기 단어 입력(1회차) : rabbit

\* 포인트단어 : cow / hen / lion / tiger / ◀ 위에서 입력한 rabbit을 포인트단어에서 찾아 지웠으므로 출력하지 않는다.

\* 사용자 입력 단어 : pointer / rabbit / ◀ 위에서 입력한 rabbit을 사용자 입력 단어에 추가하여 출력  
끝말잇기 단어 입력(2회차) : tiger

\* 포인트단어 : cow / hen / lion / ◀ tiger도 지워졌으므로 포인트단어란에 출력되지 않는다.

\* 사용자 입력 단어 : pointer / rabbit / tiger /

끝말잇기 단어 입력(3회차) : lab      ← 끝말 연결이 안되는 단어 입력 시 오류 메세지 출력  
잘못 입력하셨습니다.

\* 포인트단어 : cow / hen / lion /

\* 사용자 입력 단어 : pointer / rabbit / tiger /

끝말잇기 단어 입력(3회차) : ribbon

\* 포인트단어 : cow / hen / lion /

\* 사용자 입력 단어 : pointer / rabbit / tiger / ribbon /

끝말잇기 단어 입력(4회차) : nail

\* 포인트단어 : cow / hen / lion /

\* 사용자 입력 단어 : pointer / rabbit / tiger / ribbon / nail /

끝말잇기 단어 입력(5회차) : lion

\* 포인트단어 : cow / hen /

\* 사용자 입력 단어 : pointer / rabbit / tiger / ribbon / nail / lion /

끝말잇기 단어 입력(6회차) : noel

\* 포인트단어 : cow / hen /

\* 사용자 입력 단어 : pointer / rabbit / tiger / ribbon / nail / lion / noel /

끝말잇기 단어 입력(7회차) : lauph

\* 포인트단어 : cow / hen /

\* 사용자 입력 단어 : pointer / rabbit / tiger / ribbon / nail / lion / noel / lauph /

끝말잇기 단어 입력(8회차) : hen

\* 포인트단어 : cow /

\* 사용자 입력 단어 : pointer / rabbit / tiger / ribbon / nail / lion / noel / lauph / hen /

끝말잇기 단어 입력(9회차) : need

:

:

\* 포인트단어 : cow /

\* 사용자 입력 단어 : pointer / rabbit / tiger / ribbon / nail / lion / noel / lauph / hen / need /

끝말잇기 단어 입력(10회차) : draw

\*\* 당신의 점수는 80점 입니다

**[HW72] 답안채점프로그램**

다음과 같은 형식의 hw72\_answer.txt파일로부터 학생의 답을 읽어드려 채점한 결과를 저장한 hw72\_res.txt파일에 저장한 후 hw85\_res.txt파일의 내용을 화면에 출력하시오. (fgets(), fscanf(), fputs(), fprintf()함수 이용)

(hw72\_answer.txt파일의 내용)

```
Jung HK ← 학생이름
1 2 3 4 1 2 3 4 1 2 ← 학생이 적은 1~10번 문제까지의 답(총 문제 수는 10문항으로 한다.)
Lee SS
1 2 3 3 4 2 3 4 1 3
Hong GD
2 2 3 3 4 1 2 4 2 4
:
EOF ← 파일의 끝 표시 (데이터 파일을 만들 때 입력하는 것이 아니고 파일을 생성하면
자동으로 들어오는 표시)
```

(채점후의 hw72\_res.txt파일의 내용)

```
Jung HK
OOOXOOXOOX 70 ← 채점결과(OX)와 점수를 한 줄에 저장
Lee SS
OOOOOOOOOOX 90
Hong GD
XOOOOXXOXO 60
:
EOF
```

(출력 예)

| [ 이름 ]     | [ 점수 ] | [ 1 2 3 4 5 6 7 8 9 10 ] |
|------------|--------|--------------------------|
| 1. Jung HK | ( 70)  | ○ ○ ○ X ○ ○ X ○ ○ X      |
| 2. Lee SS  | ( 90)  | ○ ○ ○ ○ ○ ○ ○ ○ ○ ○      |
| 3. Hong GD | ( 60)  | X ○ ○ ○ ○ X X ○ X ○      |
| :          |        |                          |



## ASCII Code Table

| DEC | HEX | OCT | Char       | DEC | HEX | OCT | Char | DEC | HEX | OCT | Char |
|-----|-----|-----|------------|-----|-----|-----|------|-----|-----|-----|------|
| 0   | 00  | 000 | Ctrl-@ NUL | 43  | 2B  | 053 | +    | 86  | 56  | 126 | V    |
| 1   | 01  | 001 | Ctrl-A SOH | 44  | 2C  | 054 | ,    | 87  | 57  | 127 | W    |
| 2   | 02  | 002 | Ctrl-B STX | 45  | 2D  | 055 | -    | 88  | 58  | 130 | X    |
| 3   | 03  | 003 | Ctrl-C ETX | 46  | 2E  | 056 | .    | 89  | 59  | 131 | Y    |
| 4   | 04  | 004 | Ctrl-D EOT | 47  | 2F  | 057 | /    | 90  | 5A  | 132 | Z    |
| 5   | 05  | 005 | Ctrl-E ENQ | 48  | 30  | 060 | 0    | 91  | 5B  | 133 | [    |
| 6   | 06  | 006 | Ctrl-F ACK | 49  | 31  | 061 | 1    | 92  | 5C  | 134 | \    |
| 7   | 07  | 007 | Ctrl-G BEL | 50  | 32  | 062 | 2    | 93  | 5D  | 135 | ]    |
| 8   | 08  | 010 | Ctrl-H BS  | 51  | 33  | 063 | 3    | 94  | 5E  | 136 | ^    |
| 9   | 09  | 011 | Ctrl-I HT  | 52  | 34  | 064 | 4    | 95  | 5F  | 137 | _    |
| 10  | 0A  | 012 | Ctrl-J LF  | 53  | 35  | 065 | 5    | 96  | 60  | 140 | `    |
| 11  | 0B  | 013 | Ctrl-K VT  | 54  | 36  | 066 | 6    | 97  | 61  | 141 | a    |
| 12  | 0C  | 014 | Ctrl-L FF  | 55  | 37  | 067 | 7    | 98  | 62  | 142 | b    |
| 13  | 0D  | 015 | Ctrl-M CR  | 56  | 38  | 070 | 8    | 99  | 63  | 143 | c    |
| 14  | 0E  | 016 | Ctrl-N SO  | 57  | 39  | 071 | 9    | 100 | 64  | 144 | d    |
| 15  | 0F  | 017 | Ctrl-O SI  | 58  | 3A  | 072 | :    | 101 | 65  | 145 | e    |
| 16  | 10  | 020 | Ctrl-P DLE | 59  | 3B  | 073 | ;    | 102 | 66  | 146 | f    |
| 17  | 11  | 021 | Ctrl-Q DC1 | 60  | 3C  | 074 | <    | 103 | 67  | 147 | g    |
| 18  | 12  | 022 | Ctrl-R DC2 | 61  | 3D  | 075 | =    | 104 | 68  | 150 | h    |
| 19  | 13  | 023 | Ctrl-S DC3 | 62  | 3E  | 076 | >    | 105 | 69  | 151 | i    |
| 20  | 14  | 024 | Ctrl-T DC4 | 63  | 3F  | 077 | ?    | 106 | 6A  | 152 | j    |
| 21  | 15  | 025 | Ctrl-U NAK | 64  | 40  | 100 | @    | 107 | 6B  | 153 | k    |
| 22  | 16  | 026 | Ctrl-V SYN | 65  | 41  | 101 | A    | 108 | 6C  | 154 | l    |
| 23  | 17  | 027 | Ctrl-W ETB | 66  | 42  | 102 | B    | 109 | 6D  | 155 | m    |
| 24  | 18  | 030 | Ctrl-X CAN | 67  | 43  | 103 | C    | 110 | 6E  | 156 | n    |
| 25  | 19  | 031 | Ctrl-Y EM  | 68  | 44  | 104 | D    | 111 | 6F  | 157 | o    |
| 26  | 1A  | 032 | Ctrl-Z SUB | 69  | 45  | 105 | E    | 112 | 70  | 160 | p    |
| 27  | 1B  | 033 | Ctrl-[ ESC | 70  | 46  | 106 | F    | 113 | 71  | 161 | q    |
| 28  | 1C  | 034 | Ctrl-\ FS  | 71  | 47  | 107 | G    | 114 | 72  | 162 | r    |
| 29  | 1D  | 035 | Ctrl-] GS  | 72  | 48  | 110 | H    | 115 | 73  | 163 | s    |
| 30  | 1E  | 036 | Ctrl-^ RS  | 73  | 49  | 111 | I    | 116 | 74  | 164 | t    |
| 31  | 1F  | 037 | Ctrl_ US   | 74  | 4A  | 112 | J    | 117 | 75  | 165 | u    |
| 32  | 20  | 040 | Space      | 75  | 4B  | 113 | K    | 118 | 76  | 166 | v    |
| 33  | 21  | 041 | !          | 76  | 4C  | 114 | L    | 119 | 77  | 167 | w    |
| 34  | 22  | 042 | "          | 77  | 4D  | 115 | M    | 120 | 78  | 170 | x    |
| 35  | 23  | 043 | #          | 78  | 4E  | 116 | N    | 121 | 79  | 171 | y    |
| 36  | 24  | 044 | \$         | 79  | 4F  | 117 | O    | 122 | 7A  | 172 | z    |
| 37  | 25  | 045 | %          | 80  | 50  | 120 | P    | 123 | 7B  | 173 | {    |
| 38  | 26  | 046 | &          | 81  | 51  | 121 | Q    | 124 | 7C  | 174 |      |
| 39  | 27  | 047 | '          | 82  | 52  | 122 | R    | 125 | 7D  | 175 | }    |
| 40  | 28  | 050 | (          | 83  | 53  | 123 | S    | 126 | 7E  | 176 |      |