

C언어 기본만 하자 !

<C언어란 ?>

- 1972년 미국 벨 연구소의 데이스 리치에 의해 개발된 시스템 기술용 언어이다.
- 유닉스 운영체제 개발에 사용할 목적으로 만들어졌다.
- ▶ 특징
 - 논리적이며 구조적인 시스템 프로그래밍 언어이다.
 - 하드웨어 제어가 가능하다.
 - 프로그램 이식성이 높다.
 - 간략한 문법 표현으로 함축적인 프로그램 작성이 용이하다.
 - 저급언어 특성을 가진 고급언어이다.

<기초1>

```
/* 기초 프로그램 코딩 연습 */
```

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

```
void main(void)
```

```
{
```

```
    printf("Welcome to C Language");
```

```
    printf("Hello everyone ! ^^");
```

```
}
```

→ 주석문

→ 기본 함수명령어를 불러냄

→ 본 프로그램의 시작(main)

→ 중괄호로 열고 프로그램 코딩을 시작함

→ printf 는 화면에 출력하는 명령어

→ 한 줄이 끝날때 ; 를 해줘야 함.

→ 중괄호를 닫아서 프로그램의 끝을 알림.

- ▶ 위 프로그램의 실행 결과는 "Welcome to C LanguageHello everyone ! ^^" 이렇게 한줄로 출력이 됩니다. 아래와 같이 "Wn"을 삽입하여 수정하면 줄바꿈 처리됩니다.

<기초2>

```
/* 기초 프로그램 코딩 연습 */
```

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

```
void main(void)
```

```
{
```

```
    printf("Welcome to C LanguageWn");
```

```
    printf("Hello everyone ! ^^");
```

```
}
```

→ 주석문

→ 기본 함수명령어를 불러냄

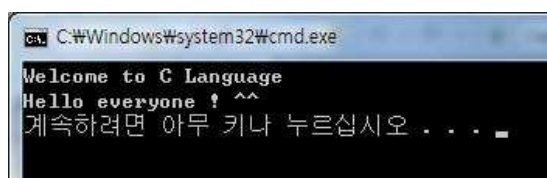
→ 본 프로그램의 시작(main)

→ 중괄호로 열고 코딩 시작

→ printf 는 화면에 출력하고 줄바꿈

→ 한 줄이 끝날때 ; 를 해줘야 함.

→ 중괄호를 닫고 프로그램 끝을 알림.



1강 C 프로그램의 구조

1. 주석문

- 주석이란 프로그램에 대한 참고사항, 부연설명으로 비실행문이다.

가. 단일라인 주석

예) `/* 주석내용 */`

나. 여러줄에 걸친 주석

예)

`/**`

`주석내용1`

`주석내용2`

`**/`

다. 필요할 때 주석달기

명령줄1;

명령줄2; `//주석내용`

명령줄3;

2. 전처리기(#) - 라이브러리 추가 설명은 C언어 보강에 있습니다.

- 컴파일 작업 전에 먼저 동작하는 명령을 의미하며 전처리기는 맨 앞에 '#'을 붙여서 구분한다.

예)

`#include<stdio.h>`

→ 입출력함수에 대한 코드정보를 가짐

`#include<math.h>`

→ 수학과관련함수에 대한 코드정보를 가짐

3. #include를 사용하는 이유.

- C언어 자체는 아주 작은 언어입니다. 그 이유는 외부의 라이브러리가 별도로 존재하기 때문입니다. 프로그램 코딩을 하기 전에 외부 라이브러리를 포함(include)시켜 사용하기 위해서 #include를 사용합니다.

예1) `stdio.h` 함수 : `printf()`, `scanf()`, `gets()`, `puts()` ...

예2) `math.h` 함수 : `sin()`, `cos()`, `tan()`...

4. main

- main() 함수는 C언어에서 반드시 존재해야하며 이 부분에서 프로그램 수행 내용을 기술
- 함수 블록의 시작과 끝에는 반드시 중괄호 { 로 시작해서 } 로 끝나야 한다. 예)

```
Void main(void)
```

```
{
기술내용1;
기술내용2;
}
```

※ Void의 의미

- main() 함수 앞에는 반환형이 오게 됩니다. 반환형이란 main() 함수를 쪽~~ 실행하여 반환되는 값이 숫자형이면 int, 반환되는 값이 아무것도 없는 경우 void를 기술합니다.

예1) 숫자형으로 반환되는 경우

```
int main()
{
코드기술..
return 0;
}
```

예2) 반환 값이 없는 경우

```
void main()
{
코드기술..
}
```

5. 사용자 정의 함수

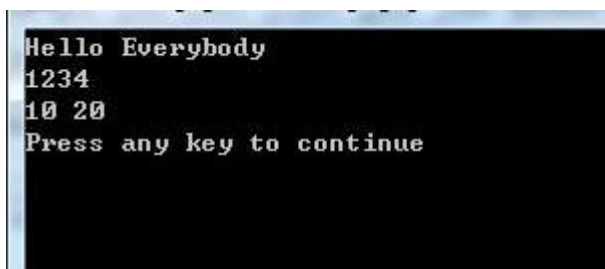
- main()에서 호출하기 위해 사용자가 작성한 함수

Printf 함수 사용 - 1

<printf 예제 1>

```
#include <stdio.h>
int main(void)
{
    printf("Hello Everybody \n");
    printf("%d \n", 1234);
    printf("%d %d \n", 10, 20);
    return 0;
}
```

<결과>

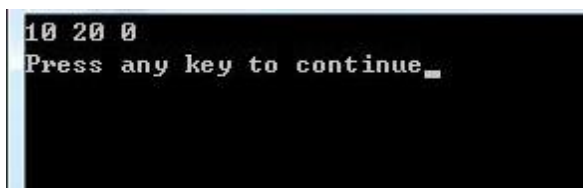
A screenshot of a terminal window showing the output of the first printf example. The text displayed is: "Hello Everybody", "1234", "10 20", and "Press any key to continue".

```
Hello Everybody
1234
10 20
Press any key to continue
```

<printf 예제 2>

```
#include <stdio.h>
int main(void)
{
    printf("%d %d %d \n", 10, 20);
    return 0;
}
```

<결과>

A screenshot of a terminal window showing the output of the second printf example. The text displayed is: "10 20 0" and "Press any key to continue".

```
10 20 0
Press any key to continue
```

※ 입력 개수를 맞추지 않아도 출력은 된다.

<printf 예제 3>

```
#include <stdio.h>

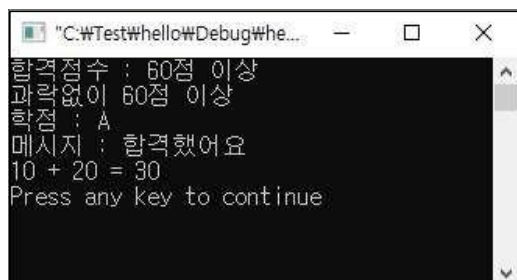
int main(void)
{
    printf("My age : %d\n", 20);
    printf("%d is my point\n", 100);
    printf("Good \n morning \n everybody\n");
    return 0;
}
```

※ 출력형식 변환문자

%문자	변환 형식
%d	10진 정수로 변환하여 출력
%f	부동소수점 형식(실수)으로 변환하여 출력
%c	한 문자로 변환하여 출력
%s	문자열로 변환하여 출력
%o (8진 정수), %x (16진 정수), %u (부호없는 10진 정수), %e (지수)	

<printf 예제5> - 출력형식 변환 문자 연습

```
#include <stdio.h>
int main(void)
{
    printf("합격점수 : %d점 이상\n", 60);
    printf("과락없이 %d점 이상\n", 60);
    printf("학점 : %c\n", 'A');
    printf("메시지 : %s\n", "합격했어요");
    printf("%d + %d = %d\n", 10, 20, 10+20);
    return 0;
}
```



```
"C:\Test\Hello\Debug\he..."
합격점수 : 60점 이상
과락없이 60점 이상
학점 : A
메시지 : 합격했어요
10 + 20 = 30
Press any key to continue
```

2강 변수, 자료형

1. 변수

- 어떤 값을 메모리 공간을 확보하여 저장하기 위한 컵이라고 생각하면 됩니다.

<변수 규칙>

- 문자, 숫자, 밑줄(_)로 구성, 변수의 첫 글자는 반드시 문자나 밑줄로 시작함
- C언어는 대소문자를 구분하며, 예약어(함수)를 사용할 수 없습니다.
- 올바른 변수 사용 예 : dulle, wonbi, kor_1, math2 등
- 잘못된 사용 예 : 1street, kor-1, int, float (int, float는 자료형을 나타내는 예약어임)

예)

```
main()
{
    int a;      → int 타입의 a 변수 선언
    a=123;     → a 변수에 초기값 123 대입 (집어 넣는다)
}
```

2. 자료형(데이터 타입)

- 프로그램에 필요한 데이터의 종류 및 데이터가 저장되는 방식 모두를 의미
- 변수를 선언하면 해당 변수에 어떠한 종류의 데이터가 들어가야 하는지를 기술해줘야 그에 맞는 메모리 공간을 할당받게 됨.

★ 자료형 종류

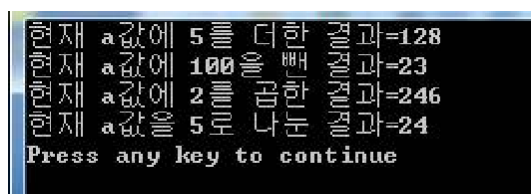
자료형	예약어	크기 (바이트)
정수형	int	4바이트
실수형	float	4바이트
	double	8바이트
문자형	char	1바이트

<변수 예제1>

```

/* 정수형 변수 선언 예 */
#include<stdio.h>
void main()
{
    int a=123;
    printf("현재 a값에 5를 더한 결과=%d\n", a+5);
    printf("현재 a값에 100을 뺀 결과=%d\n", a-100);
    printf("현재 a값에 2를 곱한 결과=%d\n", a*2);
    printf("현재 a값을 5로 나눈 결과=%d\n", a/5);
}

```

<결과>


```

현재 a값에 5를 더한 결과=128
현재 a값에 100을 뺀 결과=-23
현재 a값에 2를 곱한 결과=246
현재 a값을 5로 나눈 결과=24
Press any key to continue

```

<변수 예제2>

아래는 직사각형 둘레를 계산하는 C언어 코드입니다.

<공식> 직사각형 둘레 = (가로+세로)*2

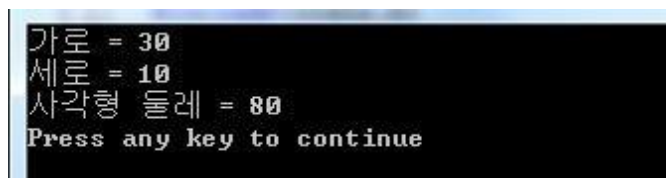
<변수>

a : 가로, b : 세로, c : 직사각형 둘레

```

#include<stdio.h>
void main()
{
    int a=30;
    int b=10;
    int c=(a+b)*2;
    printf("가로 = %d\n", a);
    printf("세로 = %d\n", b);
    printf("사각형 둘레 = %d\n", c);
}

```

<출력 형태>


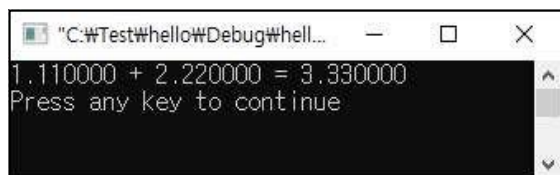
```

가로 = 30
세로 = 10
사각형 둘레 = 80
Press any key to continue

```


<변수 예제3>

```
#include<stdio.h>
void main()
{
    double num1, num2;
    double sum;
    num1 = 1.11;
    num2 = 2.22;
    sum = num1 + num2;
    printf("%f + %f = %f\n", num1, num2, sum);
}
```

**<변수 예제4> 형변환 연산자 (cast) 캐스팅 연산자**

```
#include<stdio.h>
void main()
{
    int x, y;
    x = 5;
    y = 2;
    printf("x / y : %d\n", x/y);
    printf("x / y : %f\n", (double)x / (double)y);
}
```



Scanf 함수

※ scanf 함수를 이용한 정수의 입력

<형식>

```
int main(void)
{
    int val;
    scanf("%d", &val);
    . . . . .
```

변수 val에 저장하라.
 ↑
 scanf("%d" , &val);
 ↓
 10진수 정수형으로 입력 받아서

▶ & 는 주소연산자로서 val 변수의 메모리 주소를 말한다.

<scanf 예제1>

```
#include <stdio.h>
int main(void)
{
    int result;
    int val1, val2;

    printf("첫 번째 숫자 : ");
    scanf("%d", &val1);

    printf("두 번째 숫자 : ");
    scanf("%d", &val2);

    result=val1+val2;
    printf("%d + %d = %d \n", val1, val2, result);

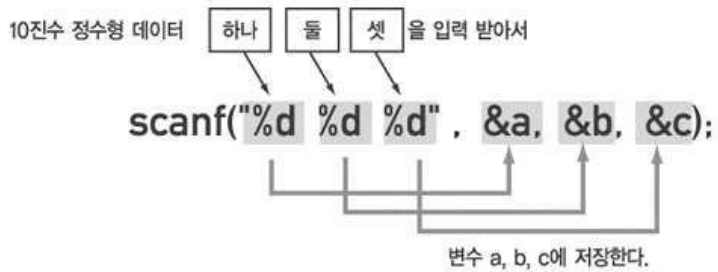
    return 0;
}
```

<결과 화면>

```
첫 번째 숫자 : 5
두 번째 숫자 : 2
5 + 2 = 7
Press any key to continue
```

<형식 2>

한번에 여러개를 입력받는 경우



<scanf 예제2>

```
#include <stdio.h>
int main(void)
{
    int result;
    int val1, val2;
    printf("숫자 둘을 입력 하세요 : ");
    scanf("%d %d", &val1, &val2);
    result=val1+val2;
    printf("%d + %d = %d Wn", val1, val2, result);
    return 0;
}
```

<결과 화면>

```
숫자 둘을 입력 하세요 : 10 20
10 + 20 = 30
Press any key to continue
```

<scanf 예제3> 입력받은 수의 제곱 구하기

```
#include<stdio.h>
int main(void)
{
    int num;
    printf("하나의 정수 입력 : ");
    scanf("%d", &num);
    printf("%d의 제곱의 결과 : %d Wn", num, num*num);
    return 0;
}
```

```
"C:\Test\hello\Deb..."
하나의 정수 입력 : 7
7의 제곱의 결과 : 49
Press any key to continue
```

3강 C의 연산자 1

비트 연산자는 C언어 보강에 있습니다.

1. 산술 연산자 : 사칙연산자와 대입연산자

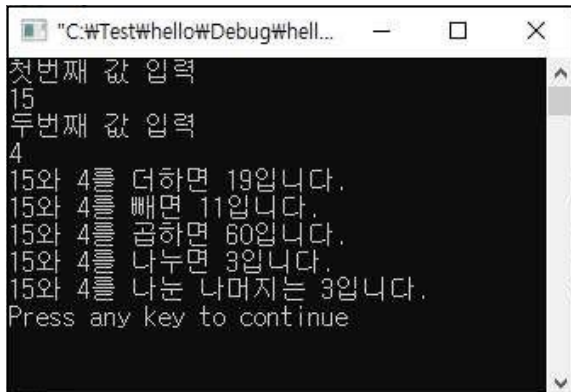
- 사칙연산자와 모듈러스 연산자 : 덧셈(+), 뺄셈(-), 곱셈(*), 나눗셈(/), 나머지(%)
- 기본 할당(대입) 연산자 : =
- ※ '=' 은 같다는 의미가 아니고 우측의 어떤 값을 좌측에 대입시키라는 의미입니다.

<연산자 예제1>

```
#include<stdio.h>

void main()
{
    int a,b,c,d,e,f;
    printf("첫번째 값 입력\n");
    scanf("%d",&a);
    printf("두번째 값 입력\n");
    scanf("%d",&b);
    printf("%d와 %d를 더하면 %d입니다.\n",a, b, a+b);
    printf("%d와 %d를 빼면 %d입니다.\n",a, b, a-b);
    printf("%d와 %d를 곱하면 %d입니다.\n",a, b, a*b);
    printf("%d와 %d를 나누면 %d입니다.\n",a, b, a/b);
    printf("%d와 %d를 나눈 나머지는 %d입니다.\n",a, b, a%b);
}
```

<출력형태>



```
"C:\Test\hello\Debug\hell..."
첫번째 값 입력
15
두번째 값 입력
4
15와 4를 더하면 19입니다.
15와 4를 빼면 11입니다.
15와 4를 곱하면 60입니다.
15와 4를 나누면 3입니다.
15와 4를 나눈 나머지는 3입니다.
Press any key to continue
```

2. 산술연산자 : 증가감소 연산자 - 2020년 1회, 2회차 기출문제

- 증가감소 연산자 : ++(증가), --(감소)

<증가감소 연산자 예>

연산자	연산의 예	의미	결합성
++a	printf("%d", ++a)	선 증가, 후 연산	←
a++	printf("%d", a++)	선 연산, 후 증가	←
--b	printf("%d", --b)	선 감소, 후 연산	←
b--	printf("%d", b--)	선 연산, 후 감소	←

- ▶ ++변수 : 현재 값에 1증가 후 연산작업 수행
- ▶ 변수++ : 연산작업을 수행 후 현재 변수값 1 증가
- ▶ --변수 : 현재 값에서 1 감소 후 연산작업 수행
- ▶ 변수-- : 연산작업 수행 후 현재 변수값 1 감소

<연산자 예제2>

```
#include<stdio.h>
void main()
{
    int a,b,c,d;
    a=10;
    b=5;
    c=a%b;
    printf("나머지=%d\n",c);
    printf("a=%d b=%d\n",a++,b);
    d=a%b;
    printf("나머지=%d   b결과=%d\n",d,++b);
}
```

<출력형태> - 괄호를 채우시오.

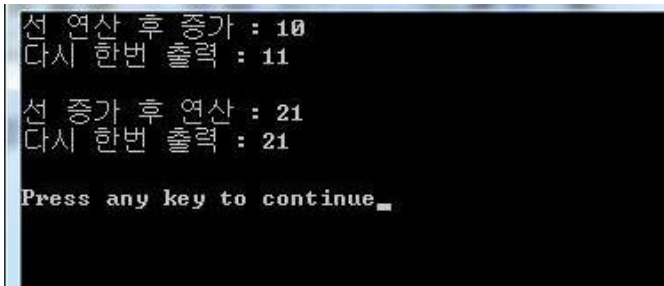
나머지=()

a=() b=()

나머지=() b결과=()

<연산자 예제 3>

```
#include <stdio.h>
int main(void)
{
    int val1=10;
    int val2=20;
    printf("선 연산 후 증가 : %d Wn", val1++);
    printf("다시 한번 출력 : %d WnWn", val1);
    printf("선 증가 후 연산 : %d Wn", ++val2);
    printf("다시 한번 출력 : %d WnWn", val2);
    return 0;
}
```

<출력형태>**<연산자 예제 4>**

아래 소스를 보고 val1 과 val2 의 결과값을 적으시오.

```
#include <stdio.h>

int main(void)
{
    int val1=10;
    int val2=(val1--)+2;

    printf("val1 : %d Wn", val1);
    printf("val2 : %d Wn", val2);

    return 0;
}
```

<결과>

val1 :

val2 :

3. 산술 연산자 - 혼합대입(할당)연산자

- 가감승제(+,-,*,/) 연산작업을 수행한 후 해당 변수에 결과 값을 재할당 할 경우 사용

- ▶ += : 기존 변수값에 특정값을 더한 후 결과를 기존변수에 다시 할당
- ▶ -= : 기존 변수에서 특정 값을 뺀 후 결과를 기존변수에 다시 할당
- ▶ *= : 기존 변수에 특정 값을 곱한 결과를 기존 변수에 다시 재할당
- ▶ /= : 기존 변수의 값을 특정값으로 나눈 결과를 다시 기존 변수에 재할당

<예>

a = a + b;	← 같은 의미 →	a += b;
a = a - b;	← 같은 의미 →	a -= b;
a = a * b;	← 같은 의미 →	a *= b;
a = a / b;	← 같은 의미 →	a /= b;
a = a % b;	← 같은 의미 →	a %= b;

<연산자 예제5>

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

```
int main(void)
{
    int val1=2;
    int val2=4;
    int val3=6;

    val1+=3;      //val1=val1+3;
    val2*=4;      //val2=val2*4;
    val3%=5;      //val3=val3%5;

    printf("result : %d %d %d \n", val1, val2, val3);

    return 0;
}
```

<출력형태>

```
result : 5 16 1
Press any key to continue
```

4. 정보처리기능사 C언어 연산자 우선순위

우선순위	종류	연산자
1	단항	(), ++, --
2		!, &, *, sizeof()
3	산술	*, /, %
4		+, -
5	관계	>, >=, <, <=
6		==, !=
7	논리	&&
8		
9	대입	=, +=, *=, -=, /=, &=

5. 문제를 풀어보고, 실습한 후에 답안 확인하기

★ a=3, b=5 인 경우 아래 두개의 식의 연산결과는 ?

(1) a=a*7+b

a 결과 ? ()

(2) a*=7+b

a 결과 ? ()

4. 관계 연산자

연산자	연산의 예	의미	결합성
<	a<b	a가 b보다 작은가	→
>	a>b	a가 b보다 큰가	→
==	a==b	a와 b가 같은가	→
!=	a!=b	a와 b가 같지 않은가	→
<=	a<=b	a가 b보다 작거나 같은가	→
>=	a>=b	a가 b보다 크거나 같은가	→

– 관계연산자 종류(왼쪽기준)

: 크다(>), 작다(<), 크거나같다(>=), 작거나같다(<=), 같다(==), 같지않다(!=)

– 결과값은 참(true), 거짓(false) 중의 하나를 가짐

– 참 : 1, 거짓 : 0

(예)

5>3 : 참(1), 4==5 : 거짓(0)

<연산자 예제 6>

```
#include<stdio.h>

void main()
{
    int a,b;
    printf("두 값을 입력 Wn");
    scanf("%d %d", &a, &b);
    printf("결과=%d",a>b);
}
```

a에 10, b에 12를 입력했다고 가정합니다.

<결과>

10

12

결과 = ()

--

<연산자 예제 7>

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

```
int main(void)
{
    int val1=10;
    int val2=12;
    int result1, result2, result3;

    result1=(val1==val2);
    result2=(val1<=val2);
    result3=(val1>=val2);

    printf("result1 : %d Wn", result1);
    printf("result2 : %d Wn", result2);
    printf("result3 : %d Wn", result3);

    return 0;
}
```

<결과>

result1 :

result2 :

result3 :

5. 논리연산자 – and, or, not

– 논리연산자 종류 : and(&&), or(||), not(!)

연산자	연산의 예	의미	결합성
&&	a&&b	true면 true 리턴	→
	a b	하나라도 true면 true 리턴	→
!	!a	true면 false를, false면 true 리턴	→

- ▶ && : 관계식1과 관계식2가 모두 참일때만 참이 됨. 나머지는 거짓
- ▶ || : 관계식1 또는 관계식2 중 어느 하나만 참이면 참이 되는 연산자. 둘다 거짓일때만 거짓
- ▶ ! : 현재 식의 값을 부정하는 연산자로 참은 거짓, 거짓은 참이 됨.

(예)

!(3>5) : 거짓을 부정하므로 참(1)이 됨

(a==3) && (b>=5) : 두 조건식이 모두 참일때만 참이 됨.

<연산자 예제8>

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

```
void main()
```

```
{
    int a = 3, b = 2, c;
    c = a == ++b;
    printf("c결과 = %d Wn",c);
    printf("결과치 = %d Wn", !(++a!=b));
}
```

<결과>

c결과=()

결과치=()

<연산자 예제9>

```
#include <stdio.h>
int main(void)
{
    int val1=10;
    int val2=12;
    int result1, result2, result3;

    result1=(val1==10 && val2==12);
    result2=(val1<12 || val2>12);
    result3=!val1;

    printf("result1 : %d \n", result1);
    printf("result2 : %d \n", result2);
    printf("result3 : %d \n", result3);

    return 0;
}
```

<실행결과>

```
result1 :
result2 :
result3 :
```

4강 선택구문 if

1. if - 2020년 1회, 2회에서 출제됨

- 조건에 따라 문장을 선택적으로 실행
- 문장이 2개 이상인 경우 복문(블록{}) 처리함 (한 문장일 때 블록처리해도 됨)

<형식1>

if(조건식)

문장1;

<형식2>

if(조건식)

{

문장1

문장2

}

<if 예제1>

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

```
void main()
```

```
{
```

```
    int a;
```

```
    printf("값을 입력하세요\n");
```

```
    scanf("%d", &a);
```

```
    if (a>10)
```

```
    {
```

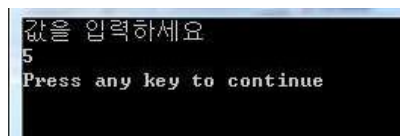
```
        a+=10;
```

```
        printf("결과=%d\n",a);
```

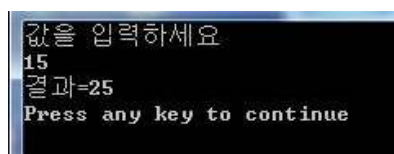
```
    }
```

```
}
```

<5를 입력>



<15를 입력>



<if 예제2>

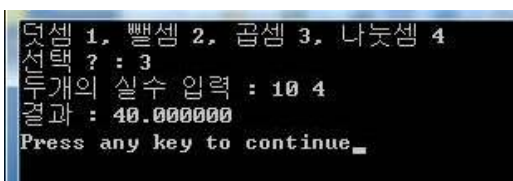
```

#include <stdio.h>
int main(void)
{
    int opt;
    float val1, val2;
    float result;
    printf("덧셈 1, 뺄셈 2, 곱셈 3, 나눗셈 4 Wn");
    printf("선택 ? : ");
    scanf("%d", &opt);
    printf("두개의 실수 입력 : ");
    scanf("%f %f", &val1, &val2);

    if(opt==1)
    {
        result = val1 + val2;
        printf("결과 : %f Wn", result);
    }
    if(opt==2)
    {
        result = val1 - val2;
        printf("결과 : %f Wn", result);
    }
    if(opt==3)
    {
        result = val1 * val2;
        printf("결과 : %f Wn", result);
    }
    if(opt==4)
    {
        result = val1 / val2;
        printf("결과 : %f Wn", result);
    }
    return 0;
}

```

<출력형태>



```

덧셈 1, 뺄셈 2, 곱셈 3, 나눗셈 4
선택 ? : 3
두개의 실수 입력 : 10 4
결과 : 40.000000
Press any key to continue_

```

<if 예제3>

```

#include <stdio.h>
int main(void)
{
    int opt;
    float val1, val2;
    float result;

    printf("덧셈 1, 뺄셈 2, 곱셈 3, 나눗셈 4 Wn");
    printf("선택 ? : ");
    scanf("%d", &opt);
    printf("두개의 실수 입력 : ");
    scanf("%f %f", &val1, &val2);

    if(opt==1)
        result = val1 + val2;

    if(opt==2)
        result = val1 - val2;

    if(opt==3)
        result = val1 * val2;

    if(opt==4)
        result = val1 / val2;
    printf("결과 : %f Wn", result);
    return 0;
}

```

2. if ~ else

– 기존 if문에 조건식이 거짓일 경우 수행할 문장을 실행하기 위한 else 구절을 추가.

<형식>

```

if(조건식)
{
    문장1;
    문장2;
}
else
{
    문장1;
    문장2;
}

```

<if 예제4>

```

/* if~else문 */
#include <stdio.h>
int main(void)
{
    int val;
    printf("정수를 하나 입력하세요 : ");
    scanf("%d", &val);

    if(val<0)
    {
        printf("입력 값은 0보다 작다 Wn");
    }
    else
    {
        printf("입력 값은 0이거나 그보다 크다 Wn");
    }
    return 0;
}

```

<if 예제5> - 2020년 1회차 기출문제와 같은 문제

한개의 숫자를 직접 입력받아서 짝수인지 홀수인지 판별하는 프로그램을 분석하세요.

<출력형태>

숫자를 입력하세요 322

입력하신 숫자 322는 짝수입니다.

```

#include<stdio.h>
void main()
{
    int a;
    printf("숫자를 입력하세요 Wn");
    scanf("%d", &a);
    if (a%2==0)
        printf("입력하신 숫자 %d는 짝수입니다.",a);
    else
        printf("입력하신 숫자 %d는 홀수입니다.",a);
}

```

3. 다중 if

- 여러 조건들 중 어느 하나를 선택해야 하는 경우 if~else if 구문을 사용

<형식>

if (조건1)

```
{
문장1;
문장2;
}
```

else if(조건2)

```
{
문장1;
문장2;
}
```

else

```
{
문장1;
문장2;
}
```

<if 예제6>

- 두개의 수를 입력받아 큰값에서 작은값을 뺀 숫자를 출력하시오.
- 만약 두개의 수가 같으면 “같은 수를 입력했습니다.”라고 출력할 것.

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

```
void main()
```

```
{
    int a,b,c;
    scanf("%d %d", &a, &b);
    if (a>b)
    {
        c=a-b;
        printf("a-b=%d\n",c);
    }
    else if (a<b)
    {
        c=b-a;
        printf("b-a=%d\n",c);
    }
    else
        printf("같은 수를 입력했습니다.\n");
}
```


<if 예제 7>

- 1~4가 아닌 다른 수를 입력했을 경우 오류메시지 출력문 추가

```
#include <stdio.h>
int main(void)
{
    int opt;
    float val1, val2;
    float result;

    printf("덧셈 1, 뺄셈 2, 곱셈 3, 나눗셈 4 Wn");
    printf("선택 ? : ");
    scanf("%d", &opt);
    printf("두개의 실수 입력 : ");
    scanf("%f %f", &val1, &val2);

    if(opt==1)
    {
        result = val1 + val2;
        printf("결과 : %f Wn", result);
    }

    else if(opt==2)
    {
        result = val1 - val2;
        printf("결과 : %f Wn", result);
    }

    else if(opt==3)
    {
        result = val1 * val2;
        printf("결과 : %f Wn", result);
    }

    else if(opt==4)
    {
        result = val1 / val2;
        printf("결과 : %f Wn", result);
    }

    else
        printf("잘못된 선택을 하셨습니다. Wn");

    return 0;
}
```

4. 조건 연산자 (삼항 연산자) - ?:

if~else문을 간결히 표현하는데 사용될 수 있다.

<형식> 조건 ? A : B

<예1>

$x = (y < 0) ? 10 : 20$;

-> y가 0보다 작으면 x에 10을 대입하고 아니면 20을 대입한다.

<예2>

$x = (y > 0) ? a * b : a / b$;

-> y가 0보다 크면 a*b의 결과를 x에 대입하고 아니면 a/b의 결과를 x에 대입

<삼항연산자 예제1>

```
#include<stdio.h>

void main()
{
    int a,b;
    printf("두 값을 입력하세요\n");
    scanf("%d %d", &a, &b);
    printf("두 값 중 큰 값은 %d입니다.\n", (a>b)?a:b);
}
```

<결과>

두 값을 입력하세요

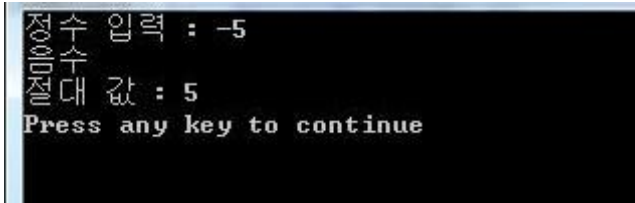
123

370

두 값 중 큰 값은 370입니다.

<삼항연산자 예제2> 아래 출력형태처럼 음수를 입력받아서 절대값 만들기를 코딩하시오.

<출력형태>



```
정수 입력 : -5
음수
절대 값 : 5
Press any key to continue
```

<소스>

```
#include <stdio.h>
int main(void)
{
    int x;
    char ch;
    printf("정수 입력 : ");
    scanf("%d", &x);

    ch=(x<0)? '-' : '+';
    (ch=='-')? printf("양수 %d\n", x): printf("음수 %d\n", x);
    printf("절대 값 : %d\n", (x<0)? -1*x: x);
    return 0;
}
```

제어문 switch~case, break, goto

1. switch ~ case

- 특정 식 혹은 변수의 상태에 따라 case 값에 일치하는 내용이 있을 경우 이를 선택적으로 실행
- switch문은 하나의 변수만을 가지고 값을 검사합니다.
- 시스템은 해당 변수의 값과 일치하는 case문의 코드를 수행합니다.
- case는 break문을 만나기 전까지 계속 실행됩니다.

<형식>

switch(식 혹은 변수)

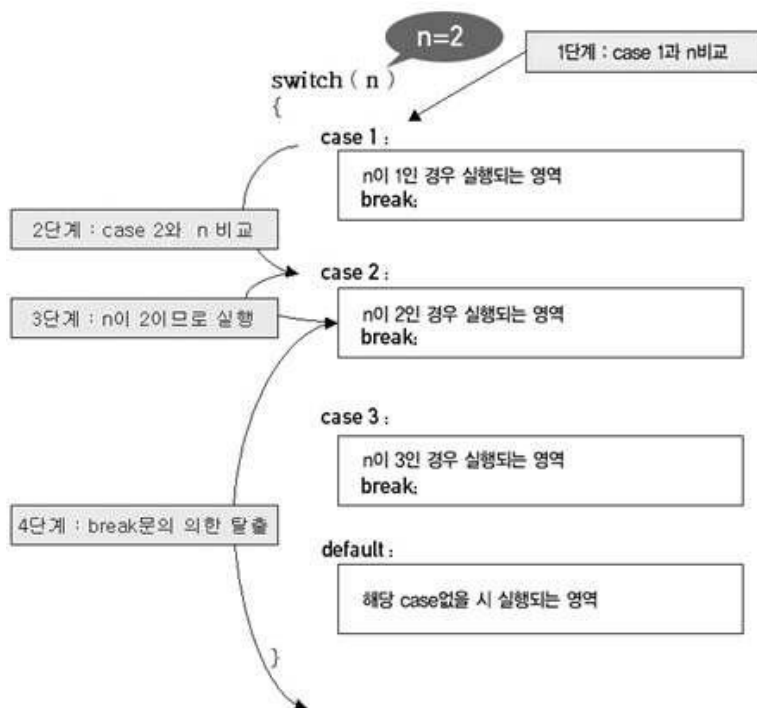
```
{
case 값1:
    문장1
case 값2:
    문장2
.
```

default:

위 내용 어디에도 해당사항 없는 경우에 수행되는 문장

```
}
```

<흐름도>



<switch 예제 1>

```
#include<stdio.h>

void main()
{
    int re;
    printf("메뉴를 입력하세요 1 to 3");
    scanf("%d", &re);
    switch(re)
    {
        case 1:
            printf("당신은 1을 입력했네요");
        case 2:
            printf("당신은 2를 입력했네요");
        case 3:
            printf("당신은 3을 입력했네요");
        default:
            printf("올바른 값을 입력하세요");
    }
}
```

<출력형태> 2를 입력한 경우 어떻게 되는지 적으시오.

메뉴를 입력하세요 1 to 3

2

당신은 2를 입력했네요

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

2. break문을 활용한 switch~case문

- 반복문 구조, switch~case문 등을 빠져나올때 사용
- 한번에 하나의 반복문만 빠져나올때 사용

<형식>

break;

<switch 예제2>

```
#include<stdio.h>
void main()
{
    int re;
    puts("메뉴를 입력하세요 1 to 3");
    scanf("%d", &re);
    switch(re)
    {
        case 1:
            puts("당신은 1을 입력했네요");
            break;
        case 2:
            puts("당신은 2를 입력했네요");
            break;
        case 3:
            puts("당신은 3을 입력했네요");
            break;
        default:
            puts("올바른 값을 입력하세요");
    }
}
```

<switch 예제3>

```
#include<stdio.h>
void main()
{
    char grade;
    int point=0;
    printf("당신의 등급을 입력해주세요(a,b,c,d,e,f)Wn");
    scanf("%c", &grade);
    switch(grade)
    {
        case 'a':
        case 'c':
        case 'e':
            point = 10;
            break;
        case 'b':
        case 'd':
            point = 15;
            break;
        case 'f':
            point = 20;
            break;
        default:
            point = 0;
    }
    printf("당신의 point는 %d 입니다.",point);
}
```

3. 제어문 – goto문

– 지정한 레이블로 무조건 분기

<형식>

goto 레이블명;

<goto 예제1>

```
#include<stdio.h>
void main()
{
    int a;
re:    → 레이블명 (콜론 사용)
    printf("값입력");
    scanf("%d", &a);
    if (a>10)
        printf("%d는 10보다 큰 값입니다.Wn",a);
    else
    {
        printf("잘못 입력 하셨습니다. 다시 입력하세요Wn");
        goto re;    → 10보다 큰 값이 입력될 때까지 무조건 분기작업 수행
    }
}
```

<출력형태>

값입력

3

잘못 입력 하셨습니다. 다시 입력하세요

값입력

7

잘못 입력 하셨습니다. 다시 입력하세요

값입력

12

12는 10보다 큰 값입니다.

반복문 - while

1. while

- while 문은 조건이 먼저 나오고 조건이 참일때까지 반복을 하는 함수입니다.
- 조건식을 만족할 때까지 문장을 반복수행

<형식>

while(조건식)

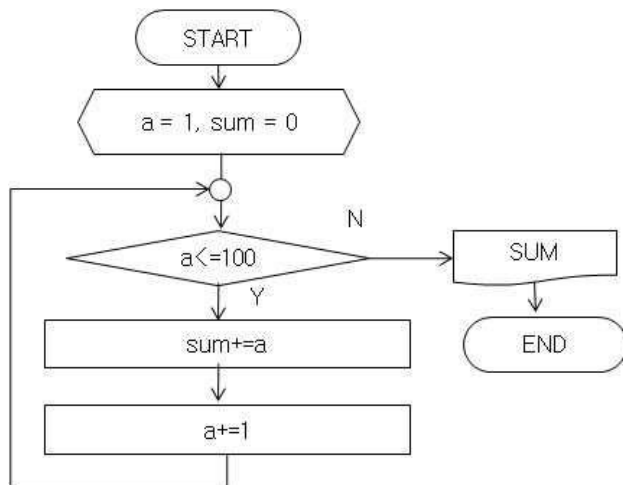
{ 반복 실행문장;

}

▶ 조건식이 참인 동안 반복 실행 문장을 실행함

<알고리즘>

1부터 100까지 합계를 출력하는 알고리즘.



<while 예제1>

```

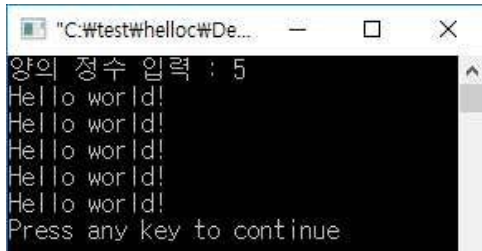
#include<stdio.h>
void main()
{
    int a=1, sum=0;
    while(a<=100){
        sum+=a;
        a++;
    }
    printf("1-----100까지 합=>%d\n",sum);
}
  
```

<출력형태> 괄호를 채우세요~

1-----100까지 합=>()

<while 예제2>

양의 정수를 하나 입력 받아서, 그 수만큼 “Hello world!”를 출력하는 프로그램



```

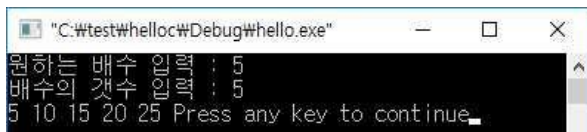
양의 정수 입력 : 5
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Press any key to continue
  
```

```

#include<stdio.h>
int main()
{
    int num;
    int i=0;
    printf("양의 정수 입력 : ");
    scanf("%d", &num);
    while(i<num)
    {
        printf("Hello world! \n");
        i++;
    }
    return 0;
}
  
```

<while 예제3>

구하려는 배수와 원하는 갯수를 입력 받은 다음, 그 수만큼 배수를 출력하는 프로그램



```

원하는 배수 입력 : 5
배수의 갯수 입력 : 5
5 10 15 20 25 Press any key to continue
  
```

```

#include<stdio.h>
int main(void)
{
    int num=0, cnt=0, i;
    printf("원하는 배수 입력 : ");
    scanf("%d", &i);
    printf("배수의 갯수 입력 : ");
    scanf("%d", &num);
    while(cnt++<num)
        printf("%d ", i*cnt);
    return 0;
}
  
```

<while 예제 4>

계속해서 정수를 입력 받고 합계를 구하고 0이 입력되면 그때까지 입력된 모든 정수의 합을 출력하고 프로그램을 종료시킨다.



```

정수 입력(0 to quit) : 5
정수 입력(0 to quit) : 3
정수 입력(0 to quit) : 2
정수 입력(0 to quit) : 7
정수 입력(0 to quit) : 1
정수 입력(0 to quit) : 0
입력값들의 합계는 18 입니다.Press any key to continue.
  
```

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

```
int main(void)
```

```
{
```

```
    int total=0;
```

```
    int num=1;
```

```
    while(num!=0)
```

```
    {
```

```
        printf("정수 입력(0 to quit) : ");
```

```
        scanf("%d", &num);
```

```
        total+=num;
```

```
    }
```

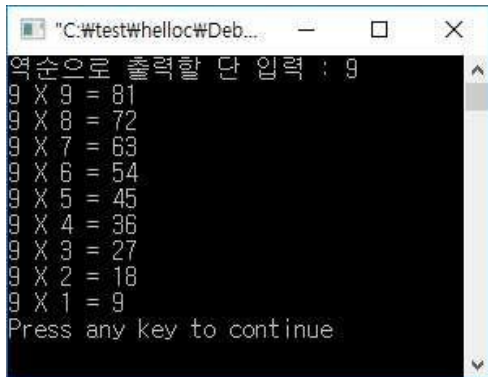
```
        printf("입력값들의 합계는 %d 입니다.",total);
```

```
    return 0;
```

```
}
```

<while 예제5>

프로그램 사용자로부터 입력 받은 숫자에 해당하는 구구단을 출력하되, 역순으로 출력하는 프로그램 소스를 분석하세요.



```

역순으로 출력할 단 입력 : 9
9 X 9 = 81
9 X 8 = 72
9 X 7 = 63
9 X 6 = 54
9 X 5 = 45
9 X 4 = 36
9 X 3 = 27
9 X 2 = 18
9 X 1 = 9
Press any key to continue

```

```

#include<stdio.h>
int main(void)
{
    int num;
    int i=9;
    printf("역순으로 출력할 단 입력 : ");
    scanf("%d", &num);

    while(i>0)
    {
        printf("%d X %d = %d\n", num, i, num*i);
        i--;
    }
    return 0;
}

```

중첩 반복문 - while

<중첩 while 예제1>

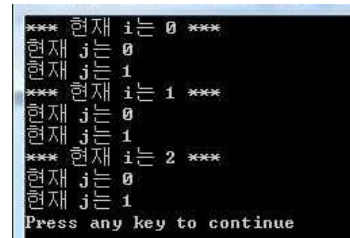
```

/* 중첩 while */
#include <stdio.h>
int main(void)
{
    int i=0, j=0;

    while(i<3)
    {
        printf("*** 현재 i는 %d *** \n",i);
        while(j<2)
        {
            printf("현재 j는 %d \n", j);
            j++;
        }

        i++;
        j=0;
    }
    return 0;
}

```

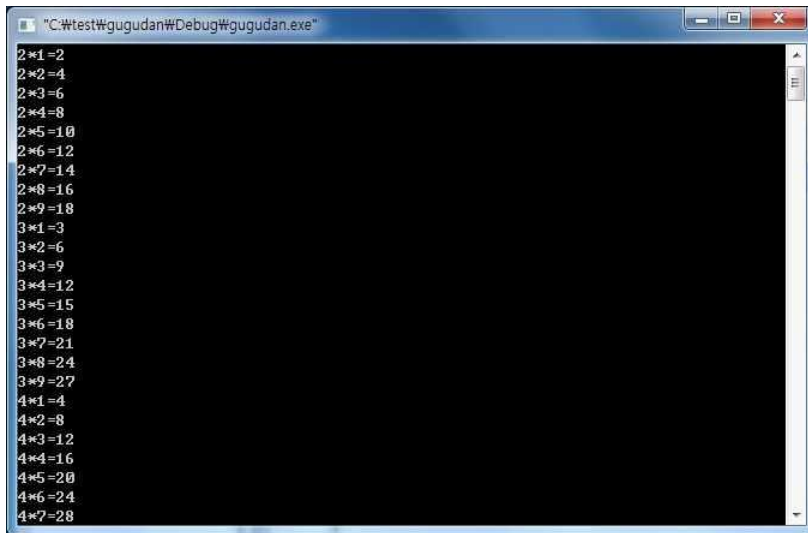


```

*** 현재 i는 0 ***
현재 j는 0
현재 j는 1
*** 현재 i는 1 ***
현재 j는 0
현재 j는 1
*** 현재 i는 2 ***
현재 j는 0
현재 j는 1
Press any key to continue

```

<중첩 while 예제2> 2단부터 9단까지 출력하는 프로그램을 분석하시오.



```

2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
3*1=3
3*2=6
3*3=9
3*4=12
3*5=15
3*6=18
3*7=21
3*8=24
3*9=27
4*1=4
4*2=8
4*3=12
4*4=16
4*5=20
4*6=24
4*7=28

```

```

#include <stdio.h>
int main(void)
{
    int i=2;
    int j=0;

    while(i<10)
    {
        j=1;
        while(j<10)
        {
            printf("%d*%d=%d Wn", i, j, i*j);
            j++;
        }
        i++;
    }
    return 0;
}

```

<중첩 while 예제3>

프로그램 사용자로부터 총 5개의 정수를 입력 받아서, 그 수의 합을 출력하는 프로그램을 작성하세요. 아래 조건을 반드시 참고해서 작성하세요.

<조건>

정수는 반드시 1이상이어야 한다. 만약 1 미만의 수가 입력되는 경우, 입력횟수에 포함하지 않고 재 입력을 받도록 하고, 1이상의 수가 입력되는 경우에만 입력횟수에 포함되도록 해야 한다. 그래서 결국 1이상의 정수 5개를 모두 입력 받을 수 있도록 프로그램을 완성할 것.

<출력 화면>

```

"C:\test\helloworld\Debug\helloworld.exe"
0보다 큰 수를 입력(1번째) : 4
0보다 큰 수를 입력(2번째) : 0
0보다 큰 수를 입력(2번째) : 3
0보다 큰 수를 입력(3번째) : -1
0보다 큰 수를 입력(3번째) : 8
0보다 큰 수를 입력(4번째) : 5
0보다 큰 수를 입력(5번째) : 0
0보다 큰 수를 입력(5번째) : 7
총 합 : 27
Press any key to continue
  
```

```

#include<stdio.h>
int main(void)
{
    int sum = 0, num = 0, i=0;
    while(i<5)
    {
        while(num<=0)
        {
            printf("0보다 큰 수를 입력(%d번째) : ", i+1);
            scanf("%d", &num);
        }
        sum+=num;
        num=0; i++;
    }
    printf("총 합 : %d Wn", sum);
    return 0;
}
  
```

반복문 - do~while

★ do~while

- 문장을 먼저 실행한 후 조건 체크

<형식>

do

{ 실행문

장;

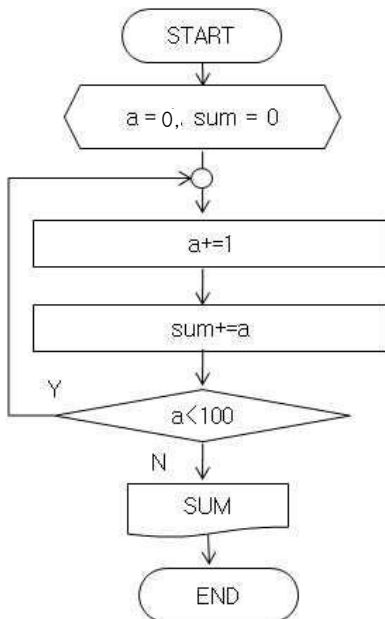
}

while(조건식);

▶ 실행→조건식→참인경우 반복, 거짓인 경우엔 종료

▶ 실행이 먼저 이루어지는 구조이므로 **최소한 한번은 문장을 수행함.**

<알고리즘> 1부터 100까지의 합계를 출력



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

```
void main()
```

```
{
```

```
    int a=0, sum=0;
```

```
    do {
```

```
        ++a;
```

```
        sum+=a;
```

```
    }
```

```
    while(a<100);
```

```
    printf("1----100까지 합=>%d\n",sum);
```

```
}
```

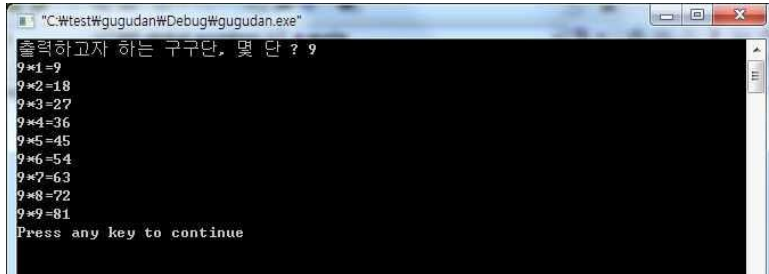
<출력결과>

```
1----100까지 합=>5050
Press any key to continue_
```


<do~while 문제-1> 원하는 단을 입력받아 구구단 출력하기

– 아래 <출력형태>를 참고하여 프로그램을 분석하시오.

<출력형태>



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

```
int main(void)
```

```
{
```

```
    int val;
```

```
    int i=1;
```

```
    printf("출력하고자 하는 구구단, 몇 단 ? ");
```

```
    scanf("%d", &val);
```

```
    do
```

```
    {
```

```
        printf("%d*%d=%d Wn", val, i, val*i);
```

```
        i++;
```

```
    }while(i<10);
```

```
    return 0;
```

```
}
```

<do~while 문제-2> 입력받는 숫자들의 합계를 출력하는 프로그램

- 입력받은 숫자의 합계를 출력하시오.
- 0을 입력받으면 그 전까지의 합계가 출력되도록 하시오.

<출력형태>



```

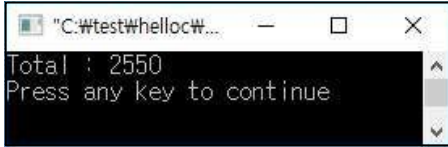
#include <stdio.h>
int main(void)
{
    int total=0;
    int val=0;

    do
    {
        printf("숫자 입력 (0 to quit) : ");
        scanf("%d", &val);
        total+=val;
    }while(val!=0);

    printf("Total : %d \n", total);
    return 0;
}
  
```

<do~while 문제-3>

0 이상 100 이하의 정수 중에서 짝수의 합을 출력하는 프로그램을 구현하되, do~while문 기반으로 구현해보자. 참고로 덧셈의 결과는 2550이 되어야 한다.

<출력형태>

```
#include<stdio.h>
int main(void)
{
    int total = 0, num = 2;

    do
    {
        total+=num;
        num=num+2;
    }while(num<=100);

    printf("Total : %d \n", total);
    return 0;
}
```

반복문 - for

2020년 1회, 2회차 출제됨 (for문, 다중for문)

1. for

- 한개 이상의 문장을 몇번이고 실행하는 프로그램 구조이며 "for loop"이라고 한다.

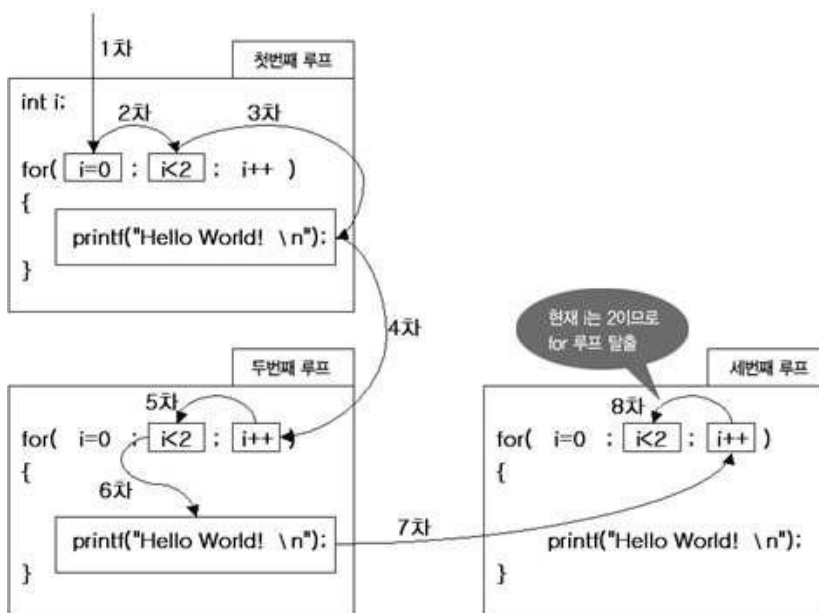
<형식>

for(초기값;조건값;증감값)

```
{
실행문장;
}
```

- ▶ 초기값 - 반복 변수를 초기화
- ▶ 조건값 - 반복 수행전에 조건을 검사
- ▶ 증감값 - 반복 실행한 후 한번씩 실행

<for문의 작동 순서>



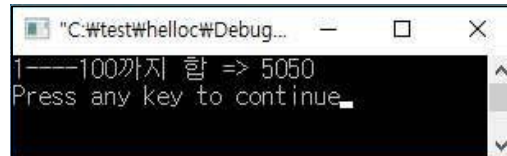
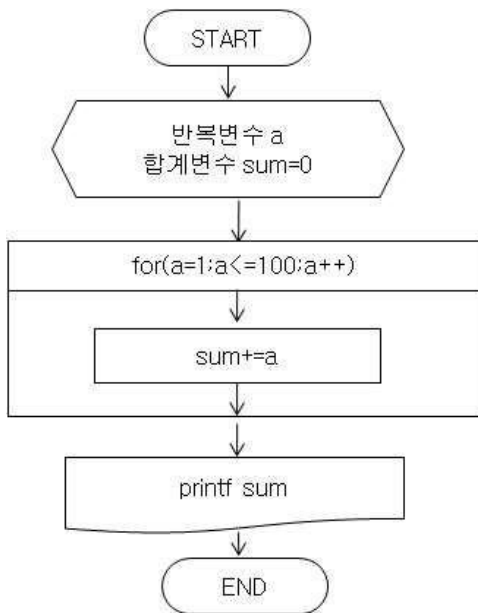
<for 예제 1> - 아래 프로그램을 분석하시오.

```
#include <stdio.h>
int main(void)
{
    int i;
    for(i=1;i<10;i++)
    {
        printf("현재 i는 %d 이다.\n", i);
    }
    return 0;
}
```



<for 예제2> - 2020년 2회차 자바 문제와 유사 (기출복원 문제 확인)

아래 알고리즘과 출력형태를 참고하여 1부터 100까지 합계를 구하는 프로그램을 for 문을 이용한 코드를 분석하세요.

<알고리즘>

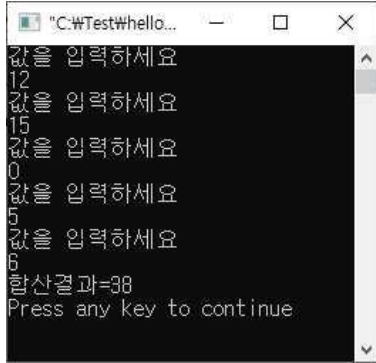
```

#include<stdio.h>
void main()
{
    int a, sum=0;
    for (a=1;a<=100;a++)
        sum+=a;    // a가 100이하 조건을 만족할 때까지 수행되는 문장
    printf("1----100까지 합=>%d\n",sum);
}
  
```

▶ for문은 초기값;조건;증가값 으로 구성되어 있어서 while 함수에 비해 프로그래밍이 간단해짐.

<for 예제3>

아래 출력형태처럼 키보드로부터 임의의 숫자 5개를 입력받아 그들의 합계를 계산하여 출력하는 프로그램을 for 문을 이용한 소스입니다.

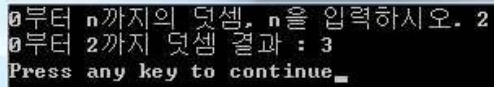
<출력형태>

```
#include<stdio.h>
void main()
{
    int i,a,sum=0;
    for (i=1;i<=5;i++)
    {
        printf("값을 입력하세요\n");
        scanf("%d",&a);
        sum+=a;
    }
    printf("합산결과=%d\n",sum);
}
```

for(i=0;i<5;i++) 로 해도 5번 반복됩니다.

<for 문제4> - 0부터 n까지의 덧셈 구하기

사용자로부터 하나의 정수를 입력받아 0부터 입력받은 수까지의 합계가 아래 <출력형태>와 같아지도록 코딩한 프로그램을 분석하세요.

<출력형태>


```
0부터 n까지의 덧셈, n을 입력하시오. 2
0부터 2까지 덧셈 결과 : 3
Press any key to continue_
```

```
#include <stdio.h>
int main(void)
{
    int total=0;
    int i, n;

    printf("0부터 n까지의 덧셈, n을 입력하시오. ");
    scanf("%d", &n);

    for(i=0;i<n+1;i++)
        total+=i;

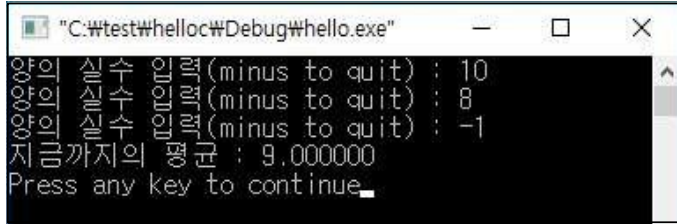
    printf("0부터 %d까지 덧셈 결과 : %d \n", n, total);
    return 0;
}
```

<for 실전문제1>-입력받은 숫자들의 평균을 구하시오.

아래 프로그램은 입력받은 숫자들의 평균을 구하는 C언어 소스입니다.

음의 실수를 입력하면 그 전까지(양의 실수들) 입력받은 수의 평균이 출력되도록 할 것.

프로그램이 정상적으로 동작하도록 (①) 을 채우세요.



```

C:\test\hello\Debug\hello.exe
양의 실수 입력(minus to quit) : 10
양의 실수 입력(minus to quit) : 8
양의 실수 입력(minus to quit) : -1
지금까지의 평균 : 9.000000
Press any key to continue_
  
```

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

```
int main(void)
```

```
{
```

```
    float total=0.0;
```

```
    float input=0.0;
```

```
    int count=0;
```

```
    for( ; input>=0.0; )
```

```
    {
```

```
        total+=input;
```

```
        printf("양의 실수 입력(minus to quit) : ");
```

```
        scanf("%f", &input);
```

```
        count++;
```

```
    }
```

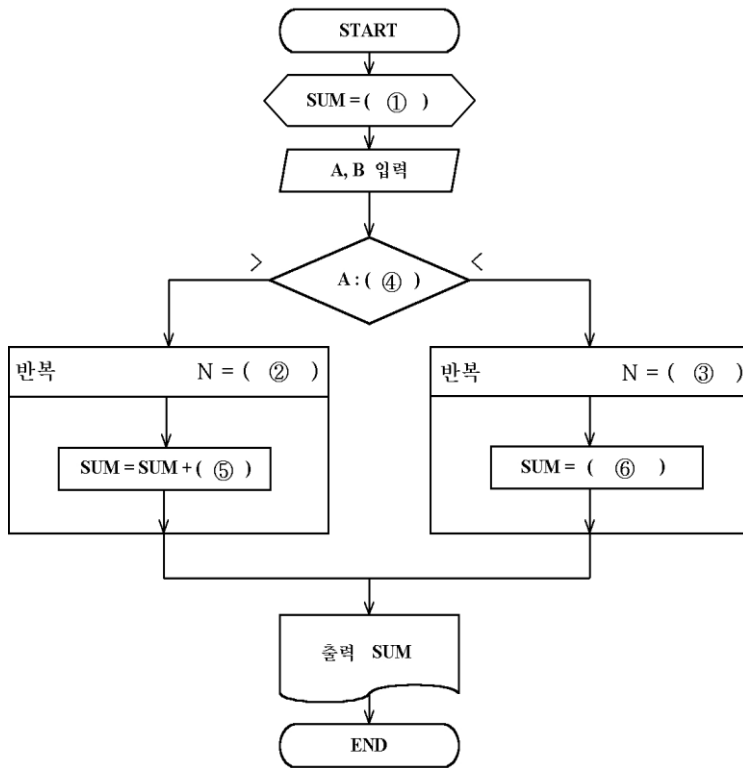
```
    printf("지금까지의 평균 : %f Wn", total/(     ①     ));
```

```
    return 0;
```

```
}
```


▶ for 실전문제2 - A와 B사이의 합계

- 시작과 끝을 5 10 또는 10 5를 입력해도 아래와 같은 결과를 얻도록 한다.



```

시작과 끝 입력 : 5 10
합계 : 45
Press any key to continue_
  
```

<실전 문제>

아래 프로그램이 정상적으로 동작하도록 (①)과 (②)를 채우세요.

start, end, tmp, result

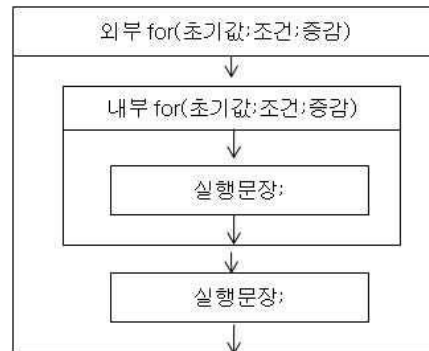
```

#include<stdio.h>
int main(void)
{
    int start, end, tmp;
    int result;
    printf("시작과 끝 입력 : ");
    scanf("%d %d", &start, &end);
    if (start > end)
    {
        ( ① ) = start;
        start = end;
        end = ( ① );
    }
    for(result=0; start<=end; start++)
        result+=( ② );
    printf("합계 : %d \n",result);
    return 0;
}
  
```

다중 for문

<형식>

```
for(초기값;조건값;증감)
{
    for(초기값;조건값;증감)
    {
        실행문장;
    }
    실행문장;
}
```



<다중 for문 예제1> - 2020년 2회차 기출문제 유사 (다중 for문)

```
#include <stdio.h>
int main(void)
{
    int i, j;
    for(i=0; i<3; i++)
    {
        printf("*** 현재 i는 %d *** \n", i);
        for(j=0; j<2; j++)
        {
            printf("현재 j는 %d \n", j);
        }
    }
    return 0;
}
```

<출력결과>

```
*** 현재 i는 0 ***
현재 j는 0
현재 j는 1
*** 현재 i는 1 ***
현재 j는 0
현재 j는 1
*** 현재 i는 2 ***
현재 j는 0
현재 j는 1
Press any key to continue_
```

<다중 for문 예제2>

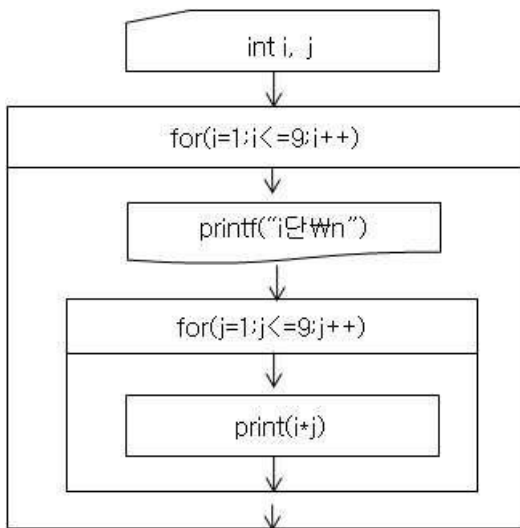
```
#include<stdio.h>
void main()
{
    int i, j;
    for (i=1;i<=2;i++)
    {
        for(j=1;j<=3;j++)
            printf("i=%d, j=%d \t", i, j);
        printf("\n");
    }
}
```

<출력결과>

```
i=1, j=1      i=1, j=2      i=1, j=3
i=2, j=1      i=2, j=2      i=2, j=3
Press any key to continue_
```

<다중 for문 예제3>

아래 순서도와 출력형태를 참고해서 이중 for문을 이용한 1단부터 9단까지 출력하는 구구단 프로그램을 분석하시오.

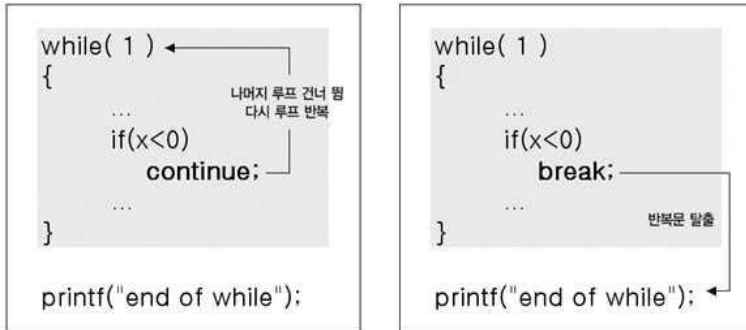
**<모범답안1>**

```

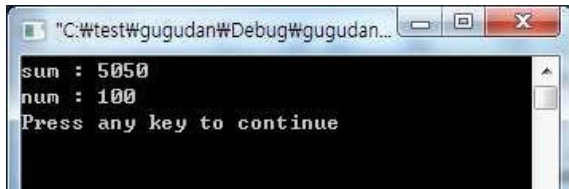
#include<stdio.h>
void main()
{
    int i, j;
    for (i=1; i<=9; i++)
    {
        printf("%d 단\\n", i);
        for(j=1; j<=9; j++)
            printf("%d * %d=%d\\n", i, j, i*j);
    }
}
  
```

break와 continue

- 1) break : 반복문을 빠져 나올 때 사용
- 2) continue : 다음 번 반복으로 넘어갈 때 사용



<출력형태>



<break 문제>

1부터 1씩 증가하는 합계를 구하되 합계가 5000 초과인 경우 반복문을 탈출하라.

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

```
int main(void)
```

```
{
```

```
    int sum=0;
```

```
    int num=0;
```

```
    while(1)
```

```
    {
```

```
        sum+=num;    //sum=sum+num
```

```
        if(sum>5000)
```

```
            break;
```

```
        num++;
```

```
    }
```

```
    printf("sum : %d \n", sum);
```

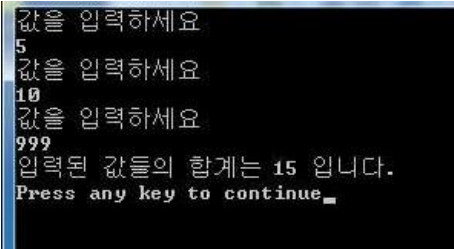
```
    printf("num : %d \n", num);
```

```
    return 0;
```

```
}
```

<break 실전문제>

– 999를 입력받기 전까지 입력받은 모든 숫자의 합계를 출력하는 프로그램

<출력형태>


```

값을 입력하세요
5
값을 입력하세요
10
값을 입력하세요
999
입력된 값들의 합계는 15 입니다.
Press any key to continue_
  
```

```

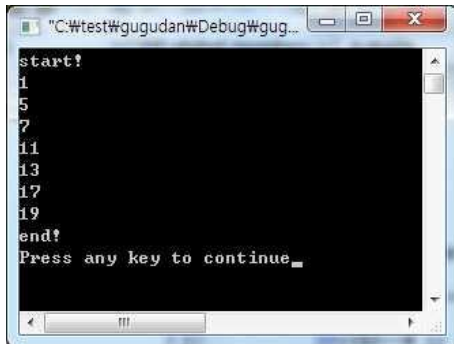
#include<stdio.h>
void main()
{
    int a=0, sum=0;
    for ( ; ; )
    {
        printf("값을 입력하세요\n");
        scanf("%d",&a);

        if(a==999)
            break;
        sum +=a;
    }
    printf("입력된 값들의 합계는 %d 입니다.\n",sum);
}
  
```

<continue 문제>

– 아래 프로그램은 0부터 19까지 2와 3의 배수가 아닌 수를 출력하는 프로그램입니다. 프로그램이 정상적으로 실행되도록 (___ ① ___)의 답안을 작성하시오.

<출력형태>



```
#include <stdio.h>
int main(void)
{
    int i;
    printf("start! \n");

    for(i=0; i<20; i++)
    {
        if(i%2==0 (___ ① ___) i%3==0)
            continue;
        printf("%d \n", i);
    }

    printf("end! \n");
    return 0;
}
```

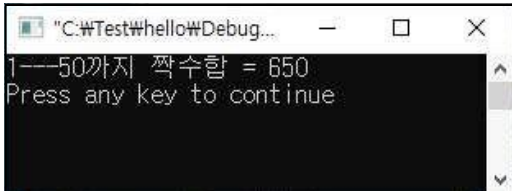
<continue 실전문제>

1부터 50까지 짝수의 합을 계산하여 출력하는 프로그램이다. 프로그램이 정상적으로 실행될 수 있도록 (①)를 채우시오

<변수>

i : 1부터 50까지 1증가하는 변수

sum : 짝수의 합계 변수



```
#include<stdio.h>
void main()
{
    int i, sum=0;
    for (i=1;i<=50;i++)
    {
        if(i%2 != ( ① ))
            continue;
        sum+=i;
    }
    printf("1---50까지 짝수합 = %d\n",sum);
}
```

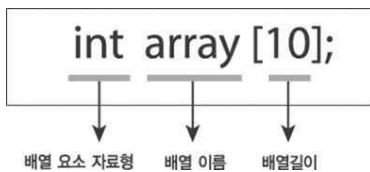
1차원 배열

▶ 배열이란 ?

- 같은 유형의 데이터를 여러 개 변수로 보관해야 할 경우 배열을 사용
- 여러 변수들을 하나의 이름으로 묶은 집합
- 첨자(또는 인덱스) : 각 값들을 구분할 수 있는 번호를 의미
- 배열요소 : 각각의 개별 값들을 일컬으며, 0부터 최고 인덱스까지 연속되어 대입

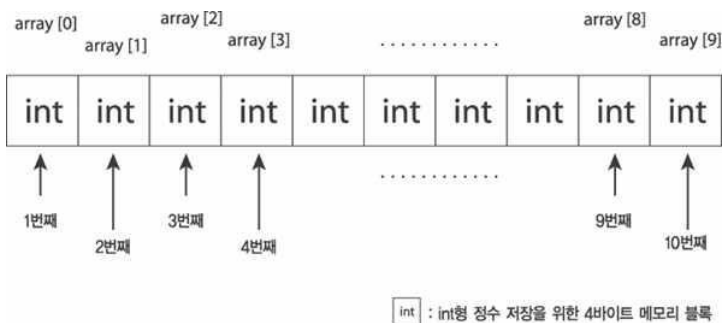
▶ 배열 선언에 있어서 필요한 것 세 가지

- 배열 길이 : 배열을 구성하는 변수의 개수 (반드시 상수를 사용)
- 배열 요소 자료형 : 배열을 구성하는 변수의 자료형
- 배열 이름 : 배열에 접근할 때 사용되는 이름



▶ 1차원 배열의 접근

- 배열 요소의 위치를 표현 : 인덱스(index)
- 인덱스는 0에서부터 시작



예) A반 5명의 시험성적

78	85	80	90	82
A[0]	A[1]	A[2]	A[3]	A[4]

<소스>

```
int a[5];
a[0]=78;
a[1]=85;
a[2]=80;
a[3]=90;
a[4]=82;
```


▶ 배열의 초기화 작업

1. 명시적 배열

– 배열 선언시 배열의 크기를 명시적으로 지정

예)

```
int sample[5];
sample[0]=10;
sample[1]=20;
```

...

<예>

```
#include <stdio.h>
int main(void)
{
    double total;
    double val[5];
    val[0]=1.01;
    val[1]=2.02;
    val[2]=3.03;
    val[3]=4.04;
    val[4]=5.05;
    total = val[0]+val[1]+val[2]+val[3]+val[4];
    printf("평균 : %lf \n", total/5);
    return 0 ;
}
```

2. 묵시적 배열

– 배열 선언시 배열의 크기를 생략하여 선언

예)

```
int sample[]={10,20,30,40,50};
```

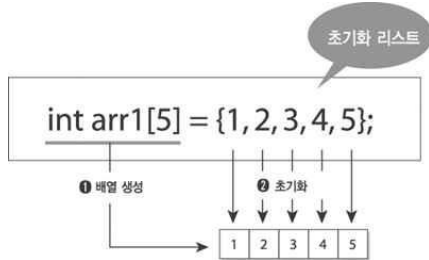
→ 배열크기를 지정하지 않고 초기값만 나열할 수 있음.

→ 묵시적 배열에서 초기값을 지정하지 않으면 error 임.

3. 선언과 동시에 초기화

```
int main(void)
{
    int arr1[5] = {1, 2, 3, 4, 5};
    int arr2[ ] = {1, 3, 5, 7, 9};
    int arr3[5] = {1, 2};
}
```

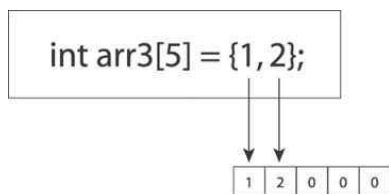
▶ `int arr1[5] = {1, 2, 3, 4, 5};`



▶ `int arr2[] = {1, 3, 5, 7, 9};`



▶ `int arr3[5] = {1, 2};`



<배열문제1>

A반 학생들의 점수를 이용하여 합계를 구한 후, 평균을 출력하는 프로그램을 완성하시오.
출력형태를 참고해서 완성할 것.

78	85	80	90	82
A[0]	A[1]	A[2]	A[3]	A[4]

<변수>

a[5] : a반 점수 배열, i : 인덱스변수, sum : 합계, abavg : a반 평균점수

```
#include<stdio.h>
void main()
{
    int a[5]={78,85,80,90,82};
    int i, sum=0, abavg;
    for(i=0;i (____①____) 5;i++)
    {
        sum+=a[i];
    }
    abavg=sum/5;
    printf("A반 학생들의 평균 점수는 %d점 입니다.\n",abavg);
}
```

<출력형태>

4. 문자열 배열

▶ 문자열 상수 : 문자열이면서 상수의 특징을 지닌다.

(예) `printf("Hello World! \n");`

▶ 문자변수와 문자열 변수

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

```
int main(void)
```

```
{
```

```
    char str1[4]={'G','o','o','d'};
```

```
    char str2[]={ "morning"};
```

```
    printf("문자배열의 크기 : %d바이트\n", sizeof(str1));
```

```
    printf("문자배열의 크기 : %d바이트\n", sizeof(str2));
```

```
    return 0;
```

```
}
```



▶ 문자열의 특징

- 문자열은 널(null)문자를 끝에 지닌다.

- 널(null) 문자 : '\0'(아스키 코드 값으로 0)

(예)

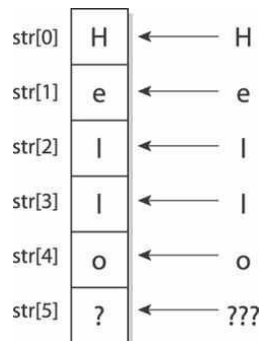
```
int main(void)
```

```
{
```

```
    char str[6]="Hello";
```

```
    printf("Hello");
```

```
    . . . . .
```



▶ 널(null) 문자를 지녀야 하는 이유

- 문자열의 끝을 표현하기 위해서

- 쓰레기 값과 실제 문자열의 경계를 나타내기 위해

- printf 함수는 널 문자를 통해서 출력의 범위를 결정 짓는다.

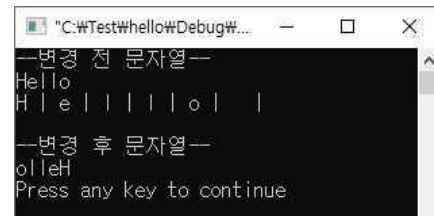
<문자열 배열문제1>

```
#include <stdio.h>
int main(void)
{
    char str1[5]="Good";
    char str2[]="morning";
    printf("%s Wn", str1);
    printf("%s %s Wn ", str1, str2);
    return 0;
}
```

**<문자열 배열문제2> 입력된 문자를 거꾸로 교체하는 프로그램**

```
#include <stdio.h>
int main(void)
{
    int i;
    char ch;
    char str[6]="Hello";
    printf("--변경 전 문자열-- Wn");
    printf("%s Wn", str);

    for(i=0; i<6; i++)
        printf("%c | ", str[i]);
    /* 문자열 변경 */
    for(i=0; i<3; i++)
    {
        ch=str[4-i];
        str[4-i]=str[i];
        str[i]=ch;
    }
    printf("Wn Wn--변경 후 문자열-- Wn");
    printf("%s Wn", str);
    return 0;
}
```




▶ 1차원배열 실전문제

아래는 5개의 정수를 배열에 입력 받고, 입력이 끝나면 최대값, 최소값, 총 합을 계산하는 C언어 소스입니다. 빈칸에 알맞은 답안을 채우시오.

※ 변수를 대상으로 & 연산자를 붙여주듯이, 배열변수에도 & 연산자를 붙여야 한다.

```
#include<stdio.h>
int main(void)
{
    int arr[5];
    int max, min, sum, i;
    for(i=0; i<5; i++)
    {
        printf("입력 : ");
        scanf("%d", ( ① ));
    }
    max=min=sum=arr[0];
    for(i=1; ( ② ); i++)
    {
        sum += arr[i];
        if(max < ( ③ ))
            max = arr[i];
        if(min > ( ③ ))
            min = arr[i];
    }
    printf("최대값 : %d \n", max);
    printf("최소값 : %d \n", min);
    printf("총 합 : %d \n", sum);
    return 0;
}
```

<출력형태>



```
"C:\wtest\helloc\De..."
입력 : 80
입력 : 90
입력 : 70
입력 : 65
입력 : 77
최대값 : 90
최소값 : 65
총 합 : 382
Press any key to continue.
```

2차원 배열

▶ 2차원 배열의 구조

- 행과 열의 구조로 자료를 표현하고자 하는 경우 사용하는 것으로 인덱스가 2개 정의됨.

<형식>

자료형 배열식별자[행][열];

예)

int a[3][3];

a[0][0]	a[0][1]	a[0][2]
a[1][0]	a[1][1]	a[1][2]
a[2][0]	a[2][1]	a[2][2]

예) 배열의 초기화 작업

1	2	3
4	5	6
7	8	9

<소스>

int a[3][3]={ {1,2,3}, {4,5,6}, {7,8,9} };

<2차원배열 문제1>



```

1 2 3
4 5 6
7 8 9
Press any key to continue

```

```

#include<stdio.h>
void main()
{
    int a[3][3]={1,2,3},{4,5,6},{7,8,9}};
    int i, j;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
            printf("Wt %d",a[i][j]);
        printf("Wn");
    }
}

```

<2차원배열 문제2>



```

1번 학생 총점 : 30
2번 학생 총점 : 90
3번 학생 총점 : 150
Press any key to continue

```

```

#include<stdio.h>
void main()
{
    int s[3][2]={10,20},{40,50},{70,80}};
    int i, j, subtotal;
    for(i=0;i<3;i++)
    {
        subtotal = 0;
        for(j=0;j<2;j++)
        {
            ( ① );
        }
        printf("%d번 학생 총점 : %dWn", ( ② ), subtotal);
    }
}

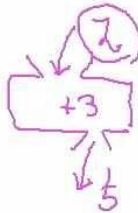
```


함수

사용자 함수 문제 : 2020년 1회, 2회차 다수 출제 (자바 포함)

return값이 있는 경우와 void인 경우에 대한 설명은 C언어 보강에 있습니다.

1. main 함수 다시 보기 : 함수의 기본 형태



2. 함수를 정의하는 이유

- 모듈화에 의한 프로그램의 질 향상이 가능
- 유지 보수 및 확장의 용이성
- 문제 해결의 용이성 : "Divide and Conquer!" (일을 나누고 전체를 정복하라 !)

3. 사용자 함수 예시

```

a      b      c
int   Add   (int i, int j)
{
    int result = i+j;
    d return result;
}
  
```

- a 반환 형
- b 함수 이름
- c 매개 변수
- d 값의 반환

<함수 문제1>

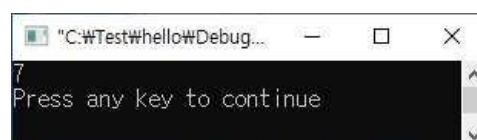
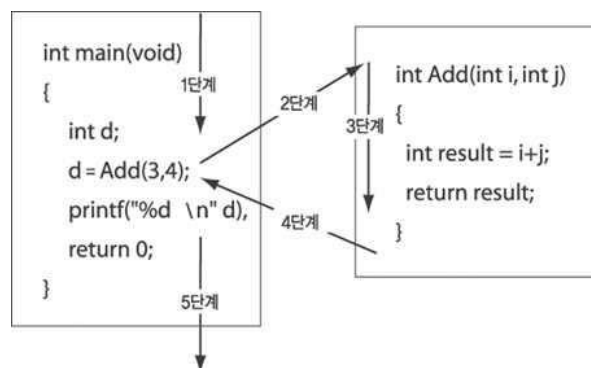
```
#include <stdio.h>
```

```
int Add(int i, int j)
```

```
{
    int result = i + j;
    return result;
}
```

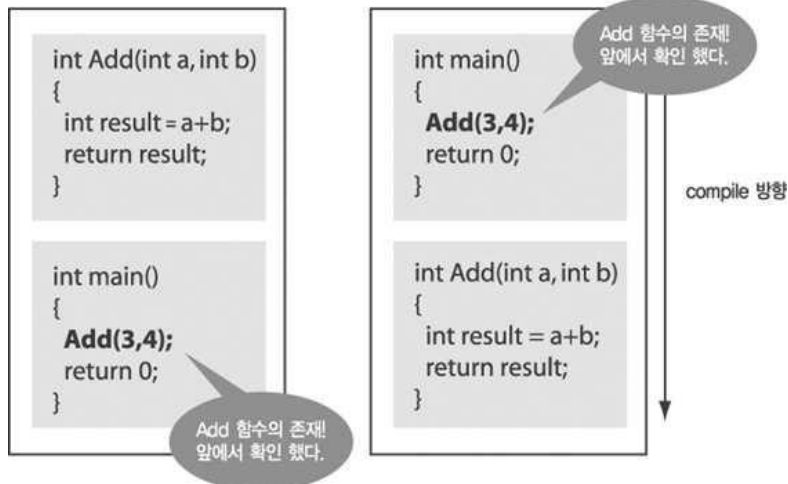
```
int main(void)
```

```
{
    int d;
    d = Add(3, 4);
    printf("%d Wn", d);
    return 0;
}
```



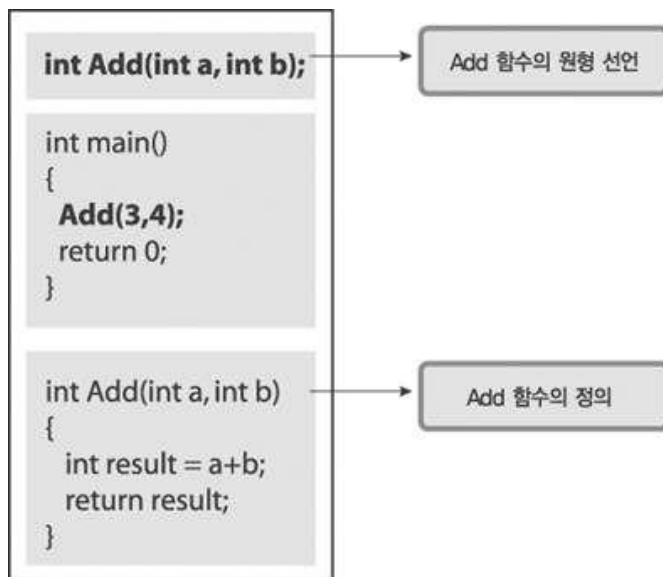
4. 함수 선언의 필요성

- 컴파일러의 특성상, 함수는 호출되기 전에 정의되어야 한다.



- 위 그림의 왼쪽처럼 Add함수를 먼저 만들고, main함수를 뒤에 오도록 해야한다.
- 왜냐하면 컴파일러는 위에서 부터 아래방향으로 컴파일을 하기 때문에, main 함수를 먼저 작성하면 ADD함수의 존재를 모르기(컴파일 전이라) 때문에 error 가 발생되기 때문이다.

▶ 해결방법 (main 함수를 상단에 오도록 하는 방법)



<함수 문제2>

```
#include<stdio.h>
int Add(int a, int b);
int Input(void);
void Result_Print(int val);
void Intro(void);
```

```
int main()
```

```
{
    int a, b;
    int result;
    Intro();                // 시작을 알림.
    a=Input();              // 값을 입력 받음
    b=Input();              // 값을 입력 받음
    result = Add(a, b);     //덧셈을 수행
    Result_Print(result);   //결과를 출력
    return 0;
}
```

```
int Add(int i, int j)
```

```
{
    return i+j;
}
```

```
int Input(void)
```

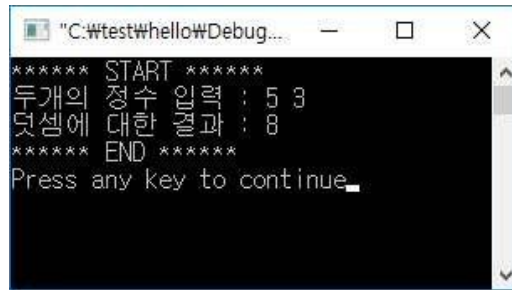
```
{
    int input; scanf("%d",
    &input); return input;
}
```

```
void Result_Print(int val)
```

```
{
    printf("덧셈에 대한 결과 : %d \n", val);
    printf("***** END ***** \n");
}
```

```
void Intro(void)
```

```
{
    printf("***** START ***** \n");
    printf("두개의 정수 입력 : ");
}
```



<함수 문제3>

```
/* 둘중에 큰수 출력하기 */
#include <stdio.h>
int Large_Num(int a, int b);

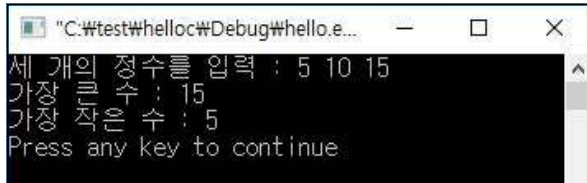
int main(void)
{
    printf("3과 4중에서 큰 수는 %d 이다 \n", Large_Num(3,4));
    printf("10과 4중에서 큰 수는 %d 이다 \n", Large_Num(10,4));

    return 0;
}

int Large_Num(int a, int b)
{
    if(a>b)
        return a;
    else
        return b;
}
```

<함수 문제 4>

세개의 정수를 인자로 전달받아서 그 중 가장 큰 수를 반환하는 함수와 가장 작은 수를 반환하는 함수를 정의해보자. 그리고 이 함수들을 호출하는 main 함수도 작성해보자



```

"C:\test\helloc\Debug\hello.e...
세 개의 정수를 입력 : 5 10 15
가장 큰 수 : 15
가장 작은 수 : 5
Press any key to continue
  
```

```

#include<stdio.h>

int maxnum(int n1, int n2, int n3)
{
    if(n1>n2)
        return (n1>n3) ? n1 : n3;
    else
        return (n2>n3) ? n2 : n3;
}

int minnum(int n1, int n2, int n3)
{
    if(n1<n2)
        return (n1<n3) ? n1 : n3;
    else
        return (n2<n3) ? n2 : n3;
}

int main(void)
{
    int num1, num2, num3;
    printf("세 개의 정수를 입력 : ");
    scanf("%d %d %d", &num1, &num2, &num3);
    printf("가장 큰 수 : %d \n", maxnum(num1, num2, num3));
    printf("가장 작은 수 : %d \n", minnum(num1, num2, num3));
    return 0;
}
  
```

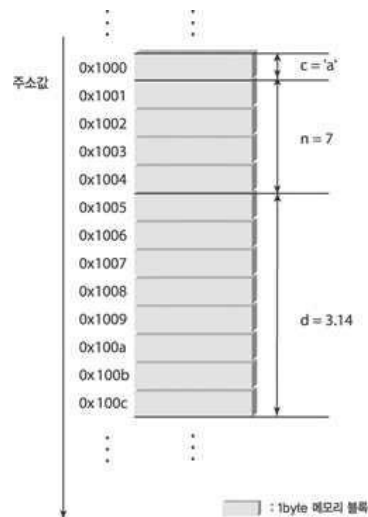
포인터의 이해 - 기초만 연습합니다.

▶ 포인터와 포인터 변수

- 메모리의 주소 값을 저장하기 위한 변수
- "포인터"를 흔히 "포인터 변수"라 한다.
- 주소 값과 포인터는 다른 것이다.

<변수 예제, 자료형 크기>

```
int main(void)
{
    char c='a';    //1바이트(한블록)
    int n=7;       //4바이트
    double d=3.14; //8바이트
    . . . . .
```



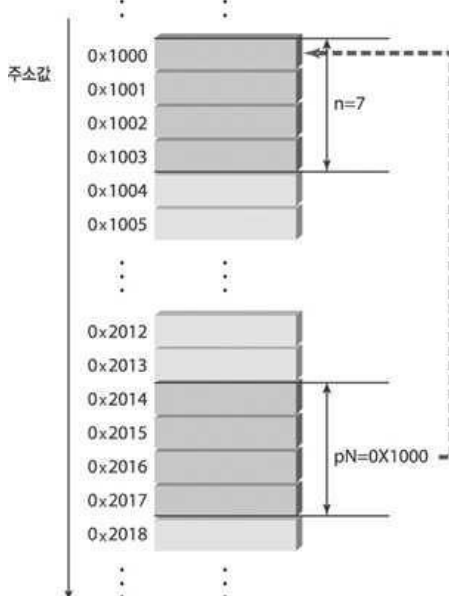
※ char c 의 주소는 ? 0x1000

※ int n 의 주소는 ? 0x1001 ~ 0x1004 이지만 시작위치인 0x1001 라고 해야 함.

※ double d 의 주소는 ? 0x1005

▶ 그림을 통한 포인터의 이해

- 컴퓨터의 주소 체계에 따라 크기가 결정
- 32비트 시스템 기반 : 4 바이트



※ pn은 n의 주소를 가리키고 있다. (pn이 포인터가 된다.)

▶ 포인터의 타입과 선언

- 포인터 = 주소값 + 자료형

- 포인터 선언 시 사용되는 연산자 : *
- A형 포인터(*A) : A형 변수의 주소 값을 저장

(포인터변수 예제-1)

```
int main(void)
{
    int *a;      // a라는 이름의 int형 포인터(a는 int형 변수 혹은 상수를 가리키는 포인터)
    char *b;     // b라는 이름의 char형 포인터(b는 char형 변수 혹은 상수를 가리키는 포인터)
    double *c;  // c라는 이름의 double형 포인터(c는 double형 변수 혹은 상수를 가리키는 포인터)
    . . . . .
```

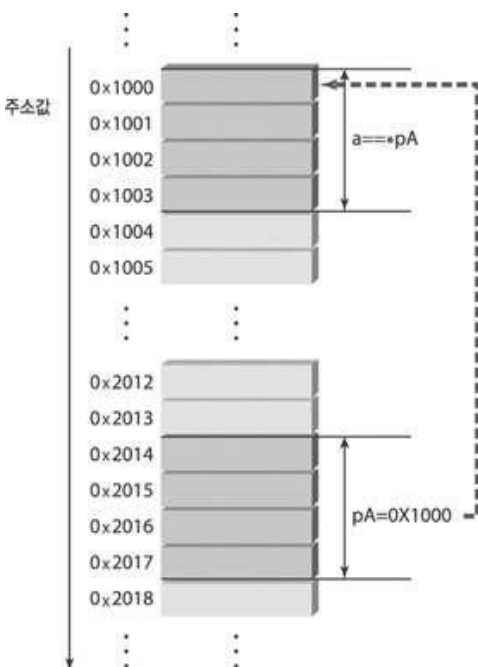
※ 포인터변수 선언 : `int* a; = int * a; = int *a;` → 3가지 방법이 모두 가능

▶ 주소 관련 연산자(뒷장 실습예제 참조)

- & 연산자(번지 연산자, 주소 연산자) : 변수의 주소 값 반환
- * 연산자 : 포인터가 가리키는 메모리 참조

(포인터변수 예제-2)

```
int main(void)
{
    int a=2005;    // a변수에 2005를 집어넣는다. a의 주소가 0x30 이라고 가정하자.
    int *pA=&a;    // pa포인터변수에 a의 주소 0x30을 저장한다.
    printf("%d\n", a); //직접 접근방식으로 a의 값 2005를 출력한다.
    printf("%d", *pA); // 간접 접근방식으로 pa라는 포인터가 가리키는 변수의 값을 출력한다.
    . . . . .
```



<포인터 실습예제>

```

#include <stdio.h>
int main(void)
{
    int a=2005;
    int *pA=&a;

    printf("pA : %d \n", pA); //a변수의 주소 출력
    printf("&a : %d \n", &a); //a변수의 주소 출력

    (*pA)++; //a++와 같은 의미를 지닌다.

    printf("a   : %d \n", a);
    printf("*pA : %d \n", *pA);

    return 0;
}

```

<결과화면>

▶ 포인터에 다양한 타입이 존재하는 이유

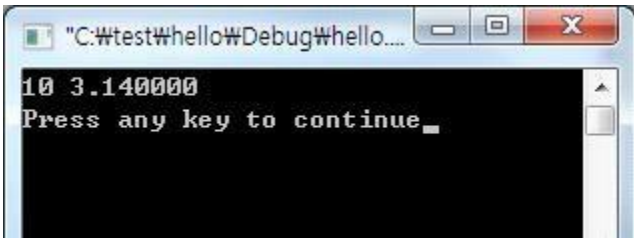
- 포인터 타입은 참조할 메모리의 크기 정보를 제공

(예제)

```
#include <stdio.h>

int main(void)
{
    int a=10;
    int *pA = &a;
    double e=3.14;
    double *pE=&e;

    printf("%d %f", *pA, *pE);
    return 0;
}
```



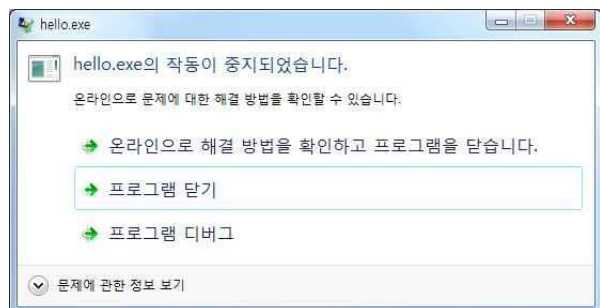
▶ 잘못된 포인터의 사용 (예제를 실행시키면 아래 경고창이 실행됨)

(예제 1)

```
int main(void)
{
    int *pA;    // pA는 쓰레기 값으로 초기화 됨
    *pA=10;
    return 0;
}
```

(예제 2)

```
int main(void)
{
    int* pA=100; // 100이 어딘 줄 알고???
    *pA=10;
    return 0;
}
```



Java 언어 기초

<자바 프로그램 구조>

객체지향형 프로그래밍 언어로서 클래스 안에 속성과 메소드를 만들어서 실행

1. 객체지향 프로그램 기법

- 현실세계의 모든 대상, 개체(entity)를 하나의 객체(object)로 만들어 이 개념을 이용하여 현실세계를 표현 및 모델링하며, 객체와 객체들이 모여 프로그램을 구성
- 객체지향 프로그램 기법의 주요 구성요소 - 객체, 클래스, 메시지 등

가. 객체(object)

- 속성과 이를 처리하기 위한 동작(연산, 메소드)을 결합시킨 실체
- 행위에 대한 특징을 나타내며, 객체는 식별성을 갖는다.
- 각 객체를 구분하기 위한 이름을 갖는다.
- 속성(attribute) : 객체가 정의된 연산을 의미(명사)
- 메소드(method) : 객체에 정의된 연산을 의미(동사)

나. 클래스(class)

- 공통된 속성의 객체들을 하나의 집합으로 묶은 단위
- 인스턴스(instance) : 하나의 클래스에 속한 각각의 객체를 의미
- 슈퍼클래스(super class) : 특정 클래스의 상위 클래스를 의미(부모클래스)



1. 기본형식

```
class hello           // 클래스 이름 정의
{
    public static void main(String[] args)           // C언어의 main 함수
    {
        System.out.println("Hello Java!");           // 화면상에 출력한 후, 줄바꿈
        System.out.println("안녕 자바!");
    }
}
```

2. if 문

```
if (조건식) {
    참일때 실행할 문장;
}
```

```
if (조건식) {
    참일때 실행할 문장;
}
else {
    거짓일때 실행할 문장
}
```

```
if (조건식1)
{ 참일때 실행할
문장;
} else if (조건식2)
{ 참일때 실행할 문장;
} else {
위 조건이 모두 거짓일때 실행할 문장;
}
```

<if 문 예제>

```

class sample_if
{
    public static void main(String[] args)
    {
        int a, b;
        a=25;
        if {(a%=5) == 0)
        {
            System.out.println("a의 값은 0입니다.");
        }
        else
        {
            System.out.println("a의 값은 0이 아닙니다.");
        }
    }
}

```

<출력 결과>

a의 값은 0입니다.

3. switch문**<형식>**

```

switch(변수) {
    case 값1:
        실행문;
        break;           // switch 문을 벗어남
    case 값2:
        실행문;
        break;
    default:
        실행문;           // case 에 해당되는 값이 없을때 실행
}

```

<switch 문 예제>

```
class sample_switch
{
    public static void main(String[] args)
    {
        char bt;
        bt='A';
        switch(bt) {
            case 'A':
                System.out.println("혈액형 A형");
                break;
            case 'B':
                System.out.println("혈액형 B형");
                break;
            case 'O':
                System.out.println("혈액형 O형");
                break;
            default:
                System.out.println("혈액형 AB형");
        }
    }
}
```

<출력 결과>

혈액형 A형

4. for문

<형식>

```
for(초기값;최종값;증가/감소값) {
    반복작업영역;
}
```

<for문 예제>

```
import java.util.Scanner;           // C언어의 scanf, 입력받기 전에 먼저 선언함
class sample_forTest
{
    public static void main(String[] args)
    {
        String str;                 // 문자열 변수 str 선언
        int i, sum=0;
        System.out.println("최종값을 입력하세요");
        Scanner scan=new Scanner(System.in);
        // 키보드로 입력(System.in) 받은 내용을 새로운(new) 내용을 scan 이름의 객체에 담아라
        str=scan.next();           // scan에 입력받은 내용을 문자열로 변환하여 str에 넣어라
        for(i=1;i<=Integer.parseInt(str);i++)
        // Integer.parseInt(str) 이란 문자열 변수 str에 담긴 내용을 정수형 숫자로 변환하라는 뜻
            sum+=i;
        System.out.printf("반복수행결과==>%d", sum);    // printf를 사용할 수 있다.
    }
}
```

<결과>

최종값을 입력하세요

5

반복수행결과==>15

5. while문

<형식>

```
while(조건식) {
    반복작업영역;
}
```

<while문 예제>

```
class sample_whiletest
{
    public static void main(String[] args)
    {
        boolean a=true;
        int cnt=0;
        while(a)
        {
            cnt+=1;
            System.out.println(cnt + "회 반복");
            if (cnt==10)
            {
                break;
            }
        }
        System.out.println("반복수행종료");
    }
}
```

<출력결과>

```
1회 반복
2회 반복
3회 반복
4회 반복
5회 반복
6회 반복
7회 반복
8회 반복
9회 반복
10회 반복
반복수행종료
```

6. do~while문

<형식>

```
do {
    반복작업영역;
} while(조건식);
```

<do~while문 예제>

```
import java.util.Scanner;
public class dowhilestudy {
    public static void main(String[] args)
    {
        int num;
        do
        {
            System.out.println("값 입력");
            Scanner i=new Scanner(System.in);
            num=i.nextInt();
            System.out.println("50보다 큰 값" + num);
        }while(num>50);
    }
}
```

<결과>

```
값 입력
101
50보다 큰 값101
값 입력
57
50보다 큰 값57
값 입력
11
50보다 큰 값11
```


<모의문제>

1. 아래 java 프로그램을 분석하여 프로그램 실행결과를 쓰세요.

```
public class javastudy {  
    public static void main(String[] args)  
    {  
        int i, sum=1;  
        for(i=1;i<=10;i+=2,sum+=i)  
            System.out.printf("합산결과=%d\n",sum);  
    }  
}
```

답 : _____

<정답>

합산결과=1

합산결과=4

합산결과=9

합산결과=16

합산결과=25

Java 배열과 문자열처리

1. 자바에서 배열생성.

- 배열은 객체로 인식하므로 new 연산자에 의해서 객체를 생성
- 배열을 생성하면서 메모리를 할당받을 때는 배열의 크기를 지정

2. 배열 선언

- 배열이름 : 연속된 값들을 참조하기 위한 값
- 배열 자체가 객체이며 배열선언과 동시에 배열명(참조변수)에 메모리 할당

<형식>

데이터타입[] 배열명 = new 데이터타입[참자갯수]

데이터타입 배열명[] = new 데이터타입[참자갯수]

<사용 예>

```
int[] arr = new int[5];
```

// 첨자가 5개인 배열을 arr 이름의 정수형 배열로 생성하라.

3. 배열의 초기화 작업 예

예1)

```
int arr[] = new int[] {10,20,30,40,50};
```

int arr[] = {10,20,30,40,50}; // C언어와 같은 방식으로 사용 가능

예2)

```
String str_arr[]={"유재석","박명수","정준하","정형돈"};
```

// 배열 선언과 동시에 초기값 할당 가능

예3)

```
String str_arr[]=new String[4]
```

```
    str_arr[0]="유재석";
```

```
    str_arr[1]="박명수";
```

```
    str_arr[2]="정준하";
```

```
    str_arr[3]="정형돈";
```

// 배열 선언과 할당은 따로따로 설정

<배열 예제>

```

class sample_array
{
    public static void main(String[] args)
    {
        int[] arr={10,20,30,40,50};
        int i;
        for(i=0;i<arr.length;i++)    // length는 arr 배열의 크기 5를 계산
            System.out.println(i + "번째 값==>" + arr[i]);
    }
}

```

<결과>

```

0번째 값==>10
1번째 값==>20
2번째 값==>30
3번째 값==>40
4번째 값==>50

```

4. 문자열 관련 함수**★ charAt(인수값)**

– 문자열에서 해당 인수번째 문자를 가져옴

★ indexOf(찾는문자)

– 문자열에서 찾고자하는 문자의 위치를 가져옴

★ Substring(시작값,마지막)

– 문자열에서 시작값 위치 문자부터 마지막값 바로 앞에 문자까지 가져옴

```

public class javastudy {
    public static void main(String[] args)
    {
        String str="wow wonderful!!!";
        System.out.println(str.charAt(5));    // o
        System.out.println(str.indexOf('d'));  // 7
        System.out.println(str.substring(4,7)); // won
    }
}

```

<모의문제>

```
public class javastudy
{
    public static void main(String[] args)
    {
        int i;
        String str="handaman";
        char a[] =str.toCharArray();    // toCharArray()는 문자열을 문자배열로 변환
        for(i=0;i<a.length;i++)
        {
            if(a[i]=='m')
                break;
        }
        System.out.printf("결과=%d\n",i);
    }
}
```

<결과>

결과=5

C언어 보강 – 1 라이브러리

★ 라이브러리에 대한 문제가 2020년 1회차 시험에서 출제되었습니다. 라이브러리에 해당하는 모든 함수를 알 필요는 없습니다. 라이브러리별 함수 종류만 보셔도 됩니다.

1. 라이브러리

– 효율적인 프로그램 개발을 위해 필요한 프로그램을 모아 놓은 집합체로

2. 라이브러리 구성

구성	설명
도움말	라이브러리를 사용할 수 있도록 하는 도움말 문서
설치 파일	라이브러리를 적용하기 위해 제공되는 설치 파일
샘플 코드	라이브러리를 이해하고 손쉽게 적용하기 위해 제공하는 샘플 소스 코드

3. 라이브러리의 종류

종류	변환 형식
표준 라이브러리	– 프로그래밍 언어가 기본적으로 가지고 있는 라이브러리로, 모듈이나 패키지 형태이다.
외부 라이브러리	– 표준 라이브러리와 달리 별도의 파일을 설치 – 외부 라이브러리는 누구나 개발하여 설치할 수 있으며, 인터넷 등을 이용하여 공유할 수도 있음

4. C언어 라이브러리 종류

종류	함수 종류
<stdlib.h>	자료형 변환, 난수 발생, 메모리 할당에 사용되는 기능들을 제공 rand(), srand(), atoi(), atof(), malloc(), free(), abort(), exit()
<stdio.h>	데이터 입출력에 사용되는 기능들을 제공 printf(), scanf(), gets(), puts()
<math.h>	수학 함수들을 제공 sin(), cos(), tan(), sqrt(), abs(), log()
<string.h>	문자열 처리에 사용되는 기능들을 제공 strlen(), strcpy(), strcat(), strcmp()
<time.h>	시간처리에 사용되는 기능들을 제공 clock(), difftime()

증가/감소 연산자

★ 2020년 1회차 시험에서 출제되었습니다. 직접 풀어보시고 강의를 듣고 확인하시기 바랍니다.

<증가/감소 연산자 문제>

아래 C언어 소스를 분석하여 출력 결과를 적으시오.

```
#include <stdio.h>
main()
{
    int a, b, c;
    a = 5 % 3;
    a--;
    b = (a++) + 3;
    printf("%d, %d \n", a, b);           결과 : (          )
    c = (++a) + 3;
    printf("%d, %d, %d\n", a, b, c);   결과 : (          )
}
```

비트 연산자

★ 비트연산자는 2020년 1회차 시험에서 출제되었습니다.

연산자	의미	비고
&	and	두개의 비트가 모두 1일 때만 1이고 나머지는 0
^	xor	두개의 비트가 서로 같으면 0, 서로 다르면 1 (가령이)
	or	두개의 비트가 모두 0일 때만 0이고 나머지는 1
~	not	0이면 1, 1이면 0
<<	왼쪽 시프트	비트를 왼쪽으로 이동
>>	오른쪽 시프트	비트를 오른쪽으로 이동

<비트 연산자 문제>

아래 C언어 소스를 분석하여 출력 결과를 적으시오.

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

```
main()
```

 $\{$

```
int a = 5, b = 7, c, d, e, f;
```

```
c = a & b;
```

```
d = a | b;
```

$$e = a \wedge b;$$
$$f = \sim b;$$

```
a = a >> 1;
```

```
b = b << 3;
```

```
printf("%d, %d, %d, %d, %d, %d\n", a, b, c, d, e, f);
```

}

결과 : ()

<약수 문제>

아래 프로그램은 1부터 9까지 10의 약수를 출력하는 프로그램이다. 괄호를 채우시오.

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

```
void main()
```

$$\{$$

```
int n = 10, i = 1;
```

```
while( i < n ) {
```

```
if ( ( ① ) == 0 ) {
```

```
printf("%d ", i);
```

}

$$(\quad \textcircled{2} \quad)++;$$

}

}

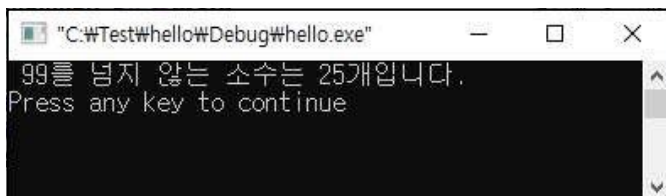
결과 : ()

<소수 문제>

아래 프로그램은 2부터 99까지 소수의 개수를 구하는 프로그램이다. 괄호를 채우시오.

```
#include <stdio.h>
int sosoo(int number)
{
    int i;
    for(i = 2; i < number; i++)
        if ( ( ① ) )
            return 0;
    return 1;
}

int main()
{
    int number = 99, cnt = 0, i;
    for(i = 2; i < number; i++)
        cnt = cnt + sosoo(i);
    printf(" %d를 넘지 않는 소수는 %d개입니다.\n", number, cnt);
    return 0;
}
결과 : ( ① )
```



자바 보강 1 – 라이브러리

1. 자바의 표준 라이브러리

– Java는 라이브러리를 패키지에 포함하여 제공하고, 각 패키지에는 Java 응용 프로그램 개발에 필요한 메소드들이 클래스로 정리되어 있다.

※ 매소드란 C언어에서의 함수를 말함.

※ C언어에서 라이브러리는 include 명령을 이용하지만 자바에서 패키지를 부를때는 import 명령을 사용함.

2. 자바 패키지 종류

종류	함수 종류
java.lang	<ul style="list-style-type: none"> – 기본적으로 필요한 인터페이스, 자료형, 예외 처리 등에 관련된 기능을 제공 – import 문 없이도 사용 가능 – 클래스 : String, System, Process, Runtime, Math, Error 등
java.util	<ul style="list-style-type: none"> – 날짜처리, 난수 발생, 복잡한 문자열 처리 등에 관련된 기능을 제공 – 클래스 : Date, Calender, Random, StringTokenizer 등
java.io	<ul style="list-style-type: none"> – 파일 입/출력과 관련된 기능 및 프로토콜을 제공 – 클래스 : InputStream, OutputStream, Reader, Writer 등
java.net	<ul style="list-style-type: none"> – 네트워크와 관련된 기능을 제공 – 클래스 : Socket, URL, InetAddress 등
java.awt	<ul style="list-style-type: none"> – 사용자 인터페이스와 관련된 기능을 제공한다. – 클래스 : Frame, Panel, Dialog, Button, Checkbox 등

자바 클래스 문제

※ 자바 클래스는 C언어에서 사용자정의 함수 사용법과 동일합니다.

<문제 1>

Java에서 날짜 처리, 난수 발생, 복잡한 문자열 처리 등에 관련된 기능을 제공하는 표준 라이브러리 로, Date, Calender, Random, StringTokenizer 등의 다양한 클래스가 포함되어 있는 패키지의 이름 을 쓰시오.

(답 :)

<문제 2>

아래 JAVA 소스를 분석하여 출력결과를 숫자로 쓰시오.

[소스]

```
public class class problem
{
    public static int a = 1;

    public static void main(String[]
        args){ increase();
        increase();
    }

    static void increase()
    {
        int b = 2;
        System.out.printf("%d, %d\n", ++a, ++b);
    }
}
```

(첫번째 출력결과 :

(두번째 출력결과 :

<문제 3>

아래 JAVA 소스를 분석하여 출력결과를 숫자로 쓰시오.

[소스]

```
public class class problem {
    public static void main(String[]
        args){ int a, b, c;
        a = 10;
        b = 20;
        c = prnt(a, b);
        System.out.printf("a=%d, b=%d, c=%d\\n", a, b, c);
    }

    static int prnt(int x, int y)
    {
        int z;
        if (x == y) z = x + y;
        else z = x - y;
        return(z);
    }
}
```

(출력결과 :)

<문제 4>

아래 JAVA 소스를 분석하여 실행 결과를 숫자로 쓰시오.

[소스]

```
public class Twocheck {
    public static void main(String args[]){
        int [], exint = { 2, 4, 2, 47, 6, 4, 7, 2, 3, 4, 5 };

        int value = 0;

        for( int i=0; i < exint.length; i++ )
            { if (exint[i] == 2 ) {
                value++;
            }
        }
        System.out.println(value);
    }
}
```

(출력결과 :)

C언어 보강 - 1 답

<증가/감소 연산자 문제>

결과 : (2, 4)

결과 : (3, 4, 6)

<비트 연산자 문제>

결과 : (2, 56, 5, 7, 2, -8)

<약수 문제>

(① $n\%i$, ② i)

<소수 문제>

(① $\text{number \% } i == 0$)

자바 보강 - 1 답

<문제1>

(`java.util`)

<문제2>

(2, 3)

(3, 3)

<문제3>

($a=10$, $b=20$, $c=-10$)

<문제4>

(3)