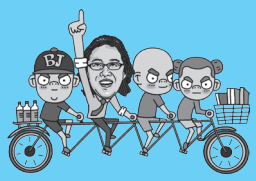
수험생의 마음으로 만든 책! 시나공 시리즈

2020 시나공

시험에 나오는 것만 공부한다!



2020년 2회 정보처리기사 실기 시험 100% 합격전략집

1일차 **2일차** 3일차 4일차 5일차 6일차 7일차 8일차 9일차 10일차
11일차 12일차 13일차 14일차 15일차 16일차 17일차 18일차 19일차 20일차

1장 프로그래밍 언어 활용

[핵심005] 연산자1

• 산술 연산자

연산자	의미
+	덧셈
–	뺄셈
*	곱셈
/	나눗셈
%	나머지
++	증가 연산자
––	감소 연산자

• 관계 연산자

연산자	의미
==	같다
!=	같지 않다
〉	크다
〉=	크거나 같다
〈	작다
〈=	작거나 같다

• 논리 연산자

연산자	의미	비고
!	not	부정
&&	and	모두 참이면 참
||	or	하나라도 참이면 참

※ 증가/감소 연산자는 변수의 앞(전치) 또는 변수의 뒤(후치)에 붙여 사용한다.

– **전치** : 변수 앞에 증감 연산자가 오는 형태로 먼저 변수의 값을 증감시킨 후 변수를 연산에 사용한다(++a, ––a).

– **후치** : 변수 뒤에 증감 연산자가 오는 형태로 먼저 변수를 연산에 사용한 후 변수의 값을 증감시킨다(a++, a––).

• 비트 연산자

연산자	의미	비고	연산자	의미	비고
&	and	모든 비트가 1일 때만 1	~	not	각 비트의 부정, 0이면 1, 1이면 0
^	xor	모든 비트가 같으면 0, 하나라도 다르면 1	〈〈	왼쪽 시프트	비트를 왼쪽으로 이동
|	or	모든 비트 중 한 비트라도 1이면 1	〉〉	오른쪽 시프트	비트를 오른쪽으로 이동

1. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include <stdio.h>
main() {
     int a = 4, b = 3, c = 5, d = 7;
     int r1, r2, r3, r4;
     r1 = 10 % a++;
     r2 = b > 3 && b > 2;
     r3 = c & d;
     r4 = d << 3;
     printf("%d, %d, %d, %d", r1, r2, r3, r4);
}
```

답 :

```
#include <stdio.h>
main() {
    int a = 4, b = 3, c = 5, d = 7;
    int r1, r2, r3, r4;
❶   r1 = 10 % a++;
❷   r2 = b > 3 && b > 2;
❸   r3 = c & d;
❹   r4 = d << 3;
❺   printf("%d, %d, %d, %d", r1, r2, r3, r4);
}
```

❶ r1 = 10 % a++;

- ㉠ : a의 초기값이 4이고 ㉠이 후치 증가 연산자이므로 연산에 사용되는 a는 4가 된다.
- ㉡ : 10 % 4의 결과인 2가 r1에 저장된다.

❷ r2 = b>3 && b>2;

- ㉠ : b는 3이므로 b > 3은 거짓(false)이다.
- ㉡ : b는 3이므로 b > 2는 참(false)이다.
- ㉢ : &&는 모두 참일 때만 참이므로 결과는 거짓(false)이다. 거짓은 0이다.

❸ r3 = c & d;

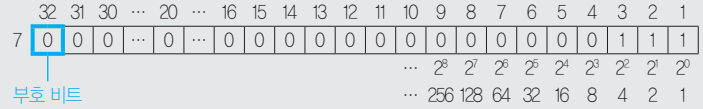
- &(비트 and)는 두 비트가 모두 1일 때만 1이 되는 비트 연산자이다.
- C 언어에서 정수형 변수는 4바이트(32비트)이므로 각 변수의 값을 4바이트 2진수로 변환한 다음 비트별로 연산한다.

```
 5 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
 7 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
 & 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
```

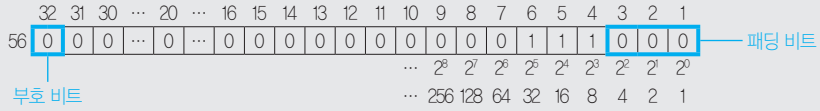
- 0000 0000 0000 0000 0000 0000 0000 0101은 10진수로 5이다.

❹ r4 = d << 3;

- <<는 왼쪽 시프트 연산자이므로, d에 저장된 값을 왼쪽으로 3비트 이동시킨 다음 그 값을 다시 d에 저장시킨다. 정수형 변수는 4바이트이므로 4바이트 2진수로 변환하여 계산하면 된다.
- 4바이트에 7을 2진수로 표현하면 다음과 같다.



- 부호를 제외한 전체 비트를 왼쪽으로 3비트 이동시킨다. 부호는 맨 왼쪽의 0이다. 양수이므로 빈 자리(패딩 비트)에는 0이 들어온다.



이것을 10진수로 변환하면 56(32+16+8)이다.

❺ r1, r2, r3, r4의 값을 화면에 출력한다.

정답 1, 2, 0, 5, 56

[핵심006] 연산자2

• 대입 연산자

연산자	예	의미
+=	a += 1	a = a + 1
-=	a -= 1	a = a - 1
*=	a *= 1	a = a * 1
/=	a /= 1	a = a / 1
%=	a %= 1	a = a % 1
〈〈=	a 〈〈= 1	a = a 〈〈 1
〉〉=	a 〉〉= 1	a = a 〉〉 1

• 조건 연산자

형식

조건 ? 수식1 : 수식2; '조건'의 수식이 참이면 '수식1'을, 거짓이면 '수식2'를 실행한다.

• 기타 연산자

연산자	의미
sizeof	자료형의 크기를 표시한다.
,(콤마)	• 콤마로 구분하여 한 줄에 두 개 이상의 수식을 작성하거나 변수를 정의한다. • 왼쪽에서 오른쪽으로 순서대로 수행되며, 순서 연산자라 부르기도 한다.
(자료형)	• 사용자가 자료형을 다른 자료형으로 변환할 때 사용하는 것으로, cast(캐스트) 연산자라고 부른다. • 변환할 자료형을 괄호 안에 넣어서 변환할 값이나 변수명 앞에 놓는다. 예 a = (int)1.3 + (int)1.4; 1.3을 정수형으로 변환한 값 1과 1.4를 정수형으로 변환한 값 1이 더해진 2가 a에 저장된다.

1. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include <stdio.h>
main() {
     int a = 2, b = 5, p = 10, q = 7;
     b -= a--;
     p %= a < b ? a++ : b++;
     q /= b % 3 ? a * b : b % a;
     printf("%d %d", p, q);
}
```

답 :

```
#include <stdio.h>
main() {
     int a = 2, b = 5, p = 10, q = 7;
❶    b -= a--;
❷    p %= a < b ? a++ : b++;
❸    q /= b % 3 ? a * b : b % a;
❹    printf("%d %d", p, q);
}
```

❶ 'b = b – a– –;'와 같다. a– –는 후치 연산이므로 연산에 사용되는 a는 2다. b에는 5 – 2의 결과인 3이 저장되고, a는 1이 된다.

❷ p %= a < b ? a++ : b++;
㉠ ㉡

- ㉠ : a는 1이고 b는 3이므로 조건(a < b)이 참이 되어 a++의 결과인 1이 사용된다.
- ㉡ : p = p % 1 = 10 % 1
- 10을 1로 나눈 나머지는 0이므로 p에는 0이 저장되고, 후치 연산으로 a는 2가 된다.

❸ q /= b % 3 ? a * b : b % a;
㉠ ㉡

- ㉠ : b는 3이므로 '3 % 3'은 0이 된다. 조건에서 0은 거짓과 같으므로 'b % a'의 결과인 1이 사용된다.
- ㉡ : q = q / 1 = 7

❹ p와 q의 값을 화면에 출력한다.

정답 1. 0 7

[핵심007] 연산자 우선순위

- 한 개의 수식에 여러 개의 연산자가 사용되면 기본적으로 다음 표의 순서대로 처리된다.
- 다음 표의 한 줄에 가로로 나열된 연산자는 우선순위가 같기 때문에 결합규칙에 따라 ←는 오른쪽에 있는 연산자부터, →는 왼쪽에 있는 연산자부터 차례로 계산된다.

대분류	중분류	연산자	결합규칙	우선 순위
단항 연산자	단항 연산자	!(논리 not) ~(비트 not) ++(증가) --(감소) sizeof(기타)	←	높음
이항 연산자	산술 연산자	* / %(나머지)	→	↑
		+ -		
	시프트 연산자	<< >>		
	관계 연산자	< <= >= >		
		==(같다) !=(같지 않다)		
	비트 연산자	&(비트 and) ^(비트 xor) |(비트 or)		
	논리 연산자	&&(논리 and) ||(논리 or)		
삼항 연산자	조건 연산자	? :	→	
대입 연산자	대입 연산자	= += -= *= /= %= <<= >>= 등	←	↓
순서 연산자	순서 연산자	,(콤마)	→	낮음

1. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include <stdio.h>
main() {
     int c = 3, d = 6, e = 3, x, y;
     x = c > 1 || d != 0;
     y = d <= 4 && e > 1;
     printf("%d, %d", x, y);
}
```

답 :


```
#include <stdio.h>
main() {
    int c = 3, d = 6, e = 3, x, y;
❶   x = c > 1 || d != 0;
❷   y = d <= 4 && e > 1;
❸   printf("%d, %d", x, y);
}
```

❶ 관계 연산자와 논리 연산자가 있으면 관계 연산자를 먼저 수행한다. 'c 〉 1'은 참이고 'd 는 0'은 참이므로 1(참) || 1(참) = 1(참)이 되어 x에 1이 저장된다.
❷ 'd 〈= 4'는 거짓이고 'e 〉 1'은 참이므로 0(거짓) && 1(참) = 0(거짓)이 되어 y에 0이 저장된다.
❸ x와 y의 값을 출력한다.

2. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include <stdio.h>
main() {
    int a = 3, b = 4, c = 5, d = 6, p, q;
    p = a * b + c >= d && d / a - b != 0;
    q = d % b + ++a * c-- || c - --a >= 10;
    printf("%d", p && q);
}
```

답 :

```
#include <stdio.h>
main() {
    int a = 3, b = 4, c = 5, d = 6, p, q;
❶   p = a * b + c >= d && d / a - b != 0;
❷   q = d % b + ++a * c-- || c - --a >= 10;
❸   printf("%d", p && q);
}
```

❶ p = a * b + c 〉= d && d / a − b != 0;
①(12) ②(2)
③(17) ④(−2)
⑤(1) ⑥(1)
⑦(1)

❷ q = d % b + ++a * c-- || c − --a 〉= 10;
④(2) ③(3) ②(5) ①(3)
⑤(15)
⑥(17) ⑦(2)
⑧(0)
⑨(1)

※ ❷의 ①은 --a에 의해 처음에는 2를 갖지만 ③의 전치 증가 연산이 적용되어 계산에 사용될 때는 3이 된다.
❸ p와 q를 &&(논리 and) 연산한 결과 1을 출력한다.

정답 1. 1, 0 2. 1

[핵심008] if문

단순 if문

- if문은 조건에 따라서 실행할 문장을 달리하는 제어문이며, 단순 if문은 조건이 한 개 일 때 사용하는 제어문이다.
- 조건이 참일 때만 실행할 문장을 지정할 수도 있고, 참과 거짓에 대해 각각 다른 실행문을 지정할 수도 있다.
- 형식1 : 조건이 참일 때만 실행한다.

  – 조건이 참일 때 실행할 문장이 하나인 경우

```
if(조건)                  if는 조건 판단문에 사용되는 예약어이므로 그대로 적는다.
                          조건은 참(1) 또는 거짓(0)이 결과로 나올 수 있는 수식을 ( ) 안에 입력한다.
    실행할 문장;          조건이 참일 경우 실행할 문장을 적는다.
```

  – 조건이 참일 때 실행할 문장이 두 문장 이상인 경우

```
{
    실행할 문장1;         { } 사이에 조건이 참일 경우 실행할 문장을 적는다.
    실행할 문장2;
        ⋮
}
```

- 형식2 : 조건이 참일 때와 거짓일 때 실행할 문장이 다르다.

```
if(조건)
    실행할 문장1;         조건이 참일 경우 실행할 문장을 적는다. 참일 경우 실행할 문장이 두 문장 이상이면 { }를 입력하고 그 사이에 문장을 적는다.
else
    실행할 문장2;         조건이 거짓일 경우 실행할 문장을 적는다. 두 문장 이상인 경우 { }를 입력하고 그 사이에 문장을 적는다.
```

다중 if문

- 조건이 여러 개 일 때 사용하는 제어문이다.
- 형식1

```
if(조건1)
    실행할 문장1;         조건1이 참일 경우 실행할 문장을 적는다.
else if(조건2)
    실행할 문장2;         조건2가 참일 경우 실행할 문장을 적는다.
else if(조건3)
    실행할 문장3;         조건3이 참일 경우 실행할 문장을 적는다.
        ⋮
else
    실행할 문장4;         앞의 조건이 모두 거짓일 경우 실행할 문장을 적는다.
```

• 형식2 : if문 안에 if문이 포함된다.

```
if(조건1)
{                          조건1이 참일 경우 실행할 문장의 시작점이다.
   if(조건2)
      실행할 문장1;        조건2가 참일 경우 실행할 문장을 적는다.
   else
      실행할 문장2;        조건2가 거짓일 경우 실행할 문장을 적는다.
}
else
   실행할 문장3;           조건1이 거짓일 경우 실행할 문장을 적는다.
```

1. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include <stdio.h>
main() {
    int a = 15, b = 22;
    if (a % 2 == 0)
        if (b % 2 == 0)
            printf("A");
        else
            printf("B");
    else if (b % 2 == 0)
        printf("C");
    else
        printf("D");
}
```

답 :

```
#include <stdio.h>
main() {
    int a = 15, b = 22;
❶   if (a % 2 == 0)
❷       if (b % 2 == 0)
❸           printf("A");
❹       else
❺           printf("B");
❻   else if (b % 2 == 0)
❼       printf("C");
❽   else
❾       printf("D");
} ❿
```

❶ a를 2로 나눈 나머지가 0이면 ❷번을 실행하고, 아니면 ❻번으로 이동한다.
❷ b를 2로 나눈 나머지가 0이면 ❸번을 실행하고, 아니면 ❹번으로 이동한다.
❸ "A"를 출력하고, ❿번으로 이동하여 프로그램을 종료한다.
❹ ❷번의 조건식이 거짓일 경우 ❺번을 실행한다.
❺ "B"를 출력하고, ❿번으로 이동하여 프로그램을 종료한다.
❻ b를 2로 나눈 나머지가 0이면 ❼번을 실행하고, 아니면 ❽번으로 이동한다.
❼ "C"를 출력하고, ❿번으로 이동하여 프로그램을 종료한다.
❽ ❻번의 조건식이 거짓일 경우 ❾번을 실행한다.
❾ "D"를 출력하고, ❿번으로 이동하여 프로그램을 종료한다.

정답 1. C

[핵심009] switch문

- 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리할 수 있는 제어문이다.
- 형식

```
switch(수식) ❶
{ ❷
case 레이블1: ❸
    실행할 문장1;
    break;
case 레이블2: ❹
    실행할 문장2;
    break;
      :
default:
    실행할 문장3;
} ❺
```

- switch는 switch문에 사용되는 예약어로 그대로 입력한다.
- 수식 : '레이블1' ~ '레이블n'의 값 중 하나를 도출하는 변수나 수식을 입력한다.

❷~❺번이 switch문의 범위이다. '{'로 시작해서 '}'로 끝난다. 반드시 입력해야 한다.

- case는 switch문에서 레이블을 지정하기 위한 예약어로 그대로 입력해야 한다.
- 레이블1 : ❶번 식의 결과가 될 만한 값 중 하나를 입력한다. 결과가 '레이블1'과 일치하면 이곳으로 찾아온다. 식의 결과가 5종류로 나타나면 case문이 5번 나와야 한다.

❶번 식의 결과가 ❸번의 '레이블1'과 일치할 때 실행할 문장이다.

switch문을 탈출하여 ❺번으로 간다.

❶번의 식의 결과가 '레이블2'와 일치하면 찾아오는 곳이다.

❶번의 식의 결과가 ❹번의 '레이블2'와 일치할 때 실행할 문장이다.

switch문을 탈출하여 ❺번으로 간다.

❶번의 식의 결과가 '레이블1' ~ '레이블n'에 해당하지 않는 경우 찾아오는 곳이다.

- case문의 레이블에는 한 개의 상수만 지정할 수 있으며, int, char, enum형의 상수만 가능하다.
- case문의 레이블에는 변수를 지정할 수 없다.
- break문은 생략이 가능하지만 break문이 생략되면 수식과 레이블이 일치할 때 실행할 문장부터 break문 또는 switch문이 종료될 때까지 모든 문장이 실행된다.

2020년 1회 기사 실기

1. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include <stdio.h>
main() {
     int c = 1;
     switch (3) {
     case 1: c += 3;
     case 2: c++;
     case 3: c = 0;
     case 4: c += 3;
     case 5: c -= 10;
     default: c--;
     }
     printf("%d", c);
}
```

답 :

```
#include <stdio.h>
main() {
❶    int c = 1;
❷    switch (3) {
     case 1: c += 3;
     case 2: c++;
❸    case 3: c = 0;
❹    case 4: c += 3;
❺    case 5: c -= 10;
❻    default: c--;
     }
❼    printf("%d", c);
}
```

모든 case문에 break문이 생략되었으므로, switch문의 인수와 일치하는 'case 3' 문장부터 switch문이 종료될 때까지 모든 문장이 실행된다.

❶ 정수형 변수 c를 선언하고 1로 초기화한다. → c = 1
❷ 3에 해당하는 숫자를 찾아간다. 'case 3' 문장으로 이동한다.
❸ c의 값을 0으로 치환한다. → c = 0
❹ 'c = c + 3'과 동일하다. c의 값에 3을 더한다. → c = 3
❺ 'c = c – 10'과 동일하다. c의 값에서 10을 뺀다. → c = –7
❻ 'c = c – 1'과 동일하다. c의 값에서 1을 뺀다. → c = –8
❼ c의 값을 출력한다.

정답 1. –8