PATH

정보처리기능사 실기 기본서

Font : Tmon몬소리 Black / 네이버 나눔고딕

PATH

정보처리기능사 실기

1. 알고리즘
2. 프로그래밍
3. 데이터베이스
4. 운영체제
5. 네트워크
6. 애플리케이션

컴퓨터자격증은 PATH

안녕하세요 **컴퓨터자격증은PATH** 입니다.
저희는 여러 교재나 유튜브 영상들이 출제자의 의도가 아닌
수험생의 합격을 위한 도서와 영상을 제작하고 있는 전문강사진 입니다.
많은 분들이 궁금해 하시는 부분은 역시 **'어디부터? 어떻게 공부를 해야 하는 것인가?'** 인데
프로그래밍 부분을 우선적으로 공부한 후 데이터베이스 네트워크 애플리케이션 운영체제 순으로
학습하는 것을 추천해 드립니다.
현재 프로그래밍과 데이터베이스 부분에서 적으면 50%에서 많으면 60%까지 출제 되고 있습니다.
이 두 과목에서 많은 점수를 획득해야 합격에 조금 더 가까워집니다.
시험 문제에 대한 범위는 NCS 범위 안에서만 나온다는 정보 이외에 아무것도 없고
2020년 한 해 동안 시험을 준비 하면서 네트워크 애플리케이션 운영체제는
공개된 범위에도 없는 문제가 많았기 때문에 다른 과목보다 프로그래밍과 데이터베이스를
우선적으로 공부하는 게 좋을 것 같습니다.
저희 예상문제를 풀다가 모르는 부분에 대해서는 카톡이나 유튜브 영상 댓글로 남겨 주시면
답변해 드리겠습니다.

정보처리이외에도 컴퓨터활용능력, 전산회계, CAD, 그래픽 등
앞으로도 컴퓨터에 대한 새로운 경로를 안내해 드리겠습니다.

정보처리기능사 실기

PATH 01. 알고리즘

PATH 01 알고리즘

1. 알고리즘

- 문제 해결을 위한 일련의 절차나 방법을 공식화한 형태 또는 계산을 실행하기 위한 단계적 절차
- 프로그램 작성 과정 중 설계/계획 단계에 포함

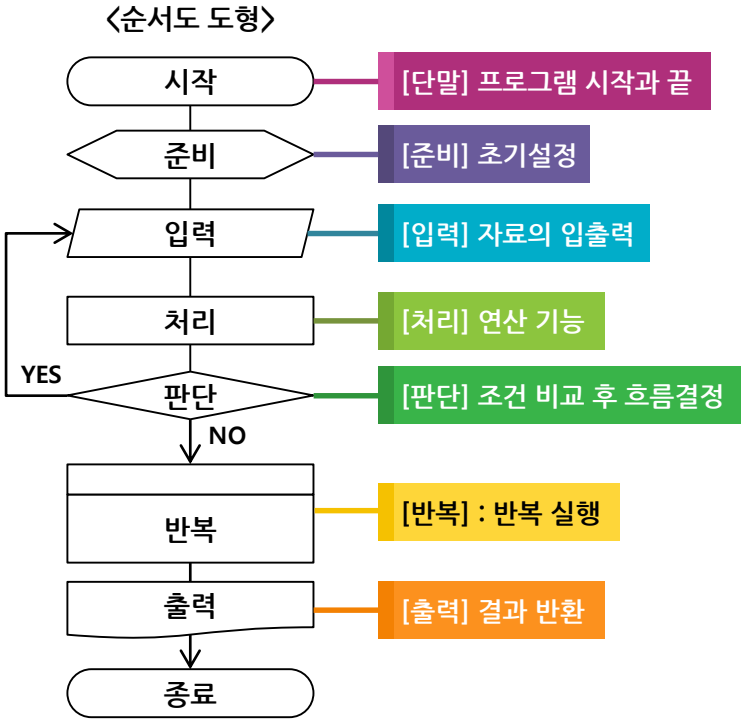
2. 순서도

- 약속된 기호를 통해 시각화된 알고리즘

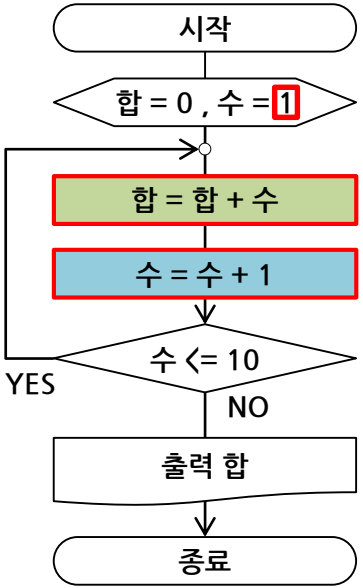
3. 응시 주의사항

- 주어진 변수, 연산자, 흐름을 이해하고 정답을 작성합니다.

※ 문제의 알고리즘은 실행 목적과 방법이 정해져 있기 때문에 개인적인 생각과 문제외적인 상황을 대입을 지양합니다.

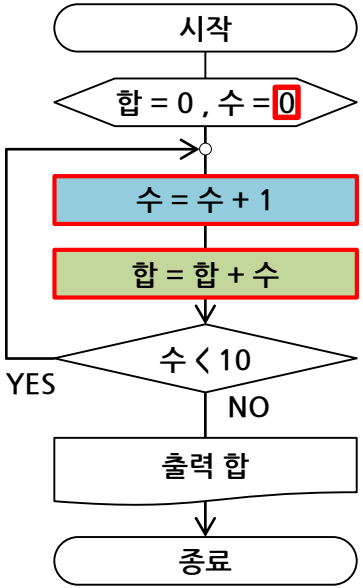


Q. 다음은 1부터 10까지의 합계를 구하는 알고리즘이다.



합		수		수<=10	
단계	값	단계	값	단계	값
①	0	②	1		
③	1	③	2	③	YES
④	3	④	3	④	YES
⑤	6	⑤	4	⑤	YES
⑥	10	⑥	5	⑥	YES
…		…		…	
⑪	45	⑪	10	⑪	YES
⑫	55	⑫	11	⑫	NO

최종 출력 값



수		합		수<=10	
단계	값	단계	값	단계	값
①	0	②	0		
③	1	③	1	③	YES
④	2	④	3	④	YES
⑤	3	⑤	6	⑤	YES
⑥	4	⑥	10	⑥	YES
…		…		…	
⑪	9	⑪	45	⑪	YES
⑫	10	⑫	55	⑫	NO

최종 출력 값

PATH 02 알고리즘 기본공식

1. 수의 증감
   - 수(i)의 기본값이 설정되어 있는 경우

   `i = i + 1` `i = i - 1`

   - 기본값이 설정되지 않은 수(i)의 감소

   `i = 고정값 - 변수`

2. 합계(sum) = 합계(sum) + 수(i)

   `sum = 0` → `sum = sum + i`

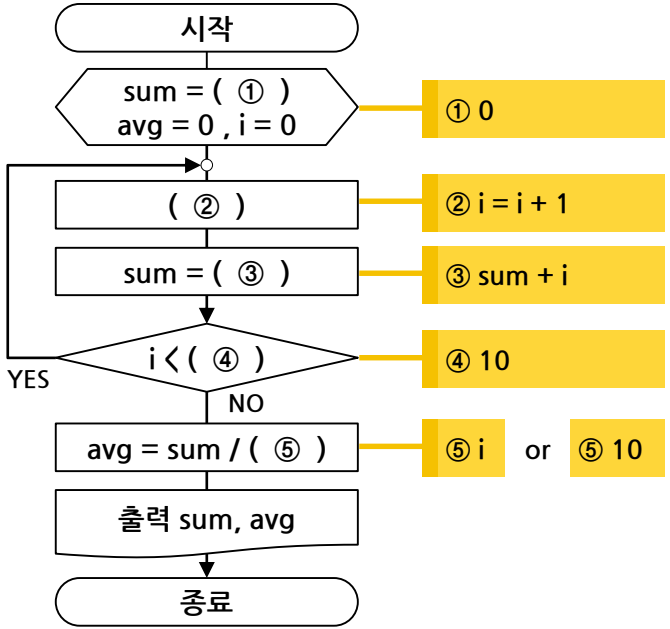
3. 평균(avg) = 합계(sum) ÷ 개수(cnt)

   `avg = sum / cnt`

4. 부호(s) 변경(+ ↔ -)

   `s = 1` or `s = -1` → `s = s * (-1)`

Q. 다음은 1부터 10까지의 합과 평균을 구하는 알고리즘이다.



PATH 02 알고리즘 기본공식

5. 몫과 나머지

- 몫 : 값 / 수 (결과값을 정수형으로 표현; int)

`5 / 2 = 2` `4 / 2 = 2`

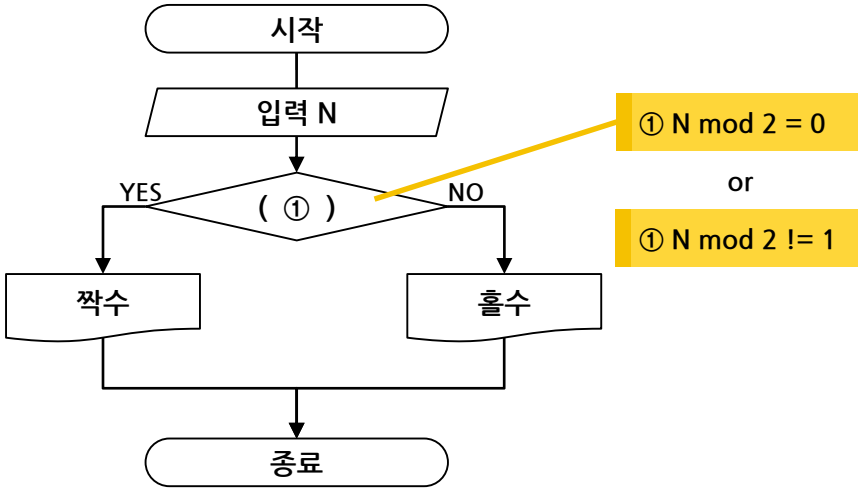
- 나머지 : 값 mod 수, 값 % 수 ▶프로그래밍에서는 %만 사용

`5 mod 2 = 1` `5 % 2 = 1`

5-1. 나머지(mod) 활용

- 값 mod 2 = 0 → 값은 짝수
- 값 mod 2 = 1 → 값은 홀수
- 값 mod 2 != 0 → 값은 홀수
- 값 mod 3 = 0 → 값은 3의 배수

Q. 다음은 입력된 N의 짝수/홀수 여부를 출력하는 알고리즘이다.



PATH 02 알고리즘 기본공식

6. 배열(수열)

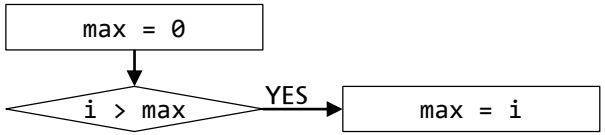
- 표현 : 배열명[배열길이]

ex> ARRAY[5]

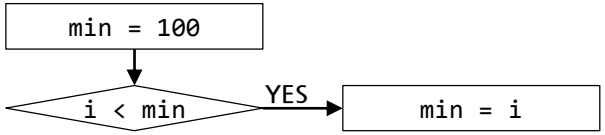
배열 색인의 기본값은 0 or 1
단, 프로그래밍에서는 0

색인(index)	0	1	2	3	4
ARRAY	수	수	수	수	수

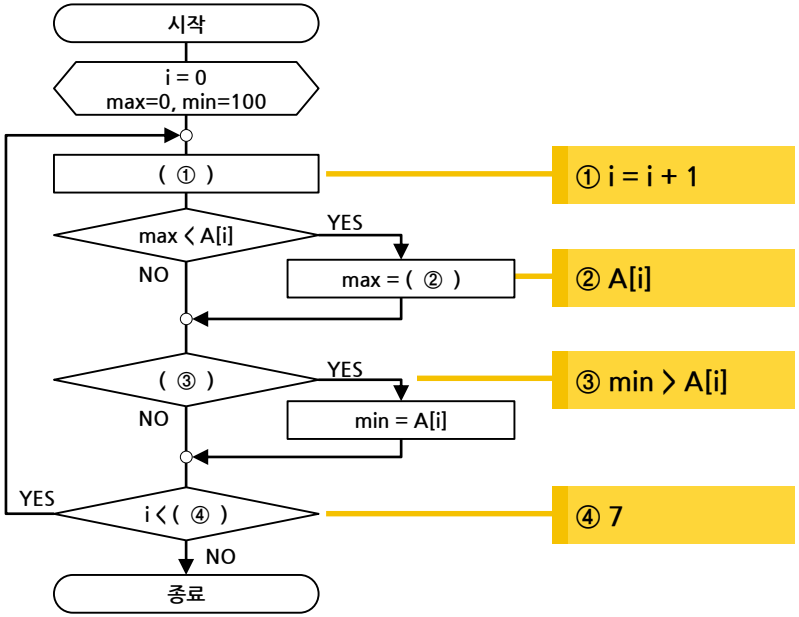
7-1. 최대값(max)



7-2. 최소값(min)



Q. 다음은 배열 A에 입력된 7개의 정수 중에서 최대값과 최소값을 구하는 알고리즘이다.



PATH 02 알고리즘 기본공식

8. 순위

- 순위의 초기값은 1
- 순위 선정시, 값을 비교하여 작을 경우 순위를 증가(+1)

ex> 배열을 이용하여 순위 구하기

`rank = 1` `+1` ▶ score(0):90점의 순위는 2위

`score(0) < score(1)`

score	90	85	60	95	50

FALSE

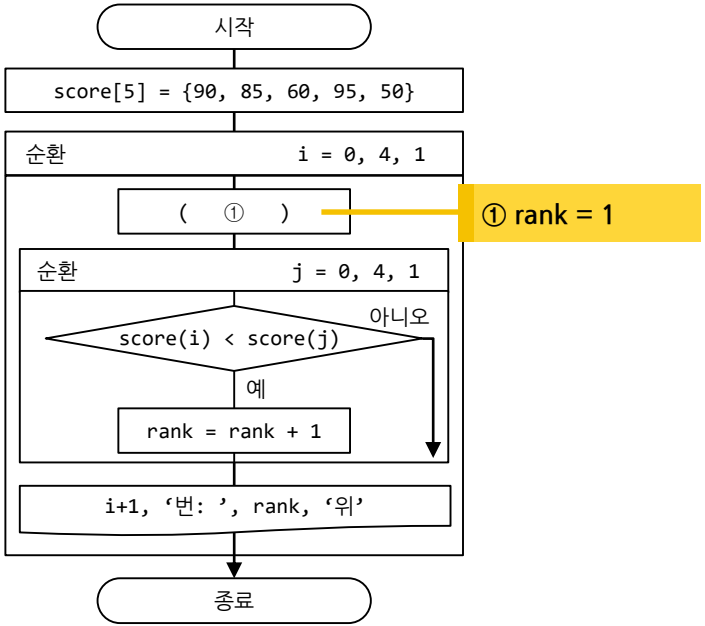
FALSE

TRUE

`rank = rank + 1`

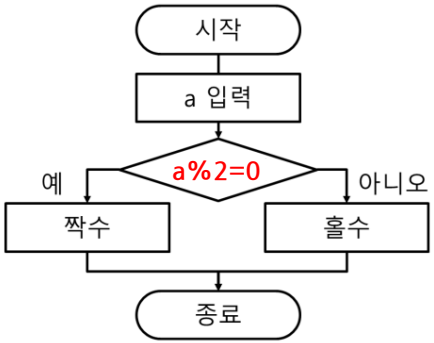
FALSE

Q. 배열로 입력 받은 점수의 순위를 출력하는 프로그램을 완성하시오.



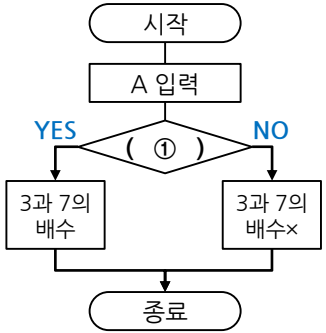
PATH 03 알고리즘 기출문제

[2020년 1회 기출문제]
정수를 입력 받아 짝수와 홀수를 출력하는 프로그램이 있다.
아래의 괄호 안에 적합한 표현을 작성하시오.




```
#include<stdio.h>
int main(void) {
   int a;
   scanf("%d", &a);
   if ( a%2==0 )
      printf("짝수");
   else
      printf("홀수");
}
```

[2020년 2회 기출문제]
정수를 입력 받아, 3과 7의 배수 여부를 확인하는 프로그램이 있다.
아래의 괄호 안에 적합한 표현을 작성하시오.



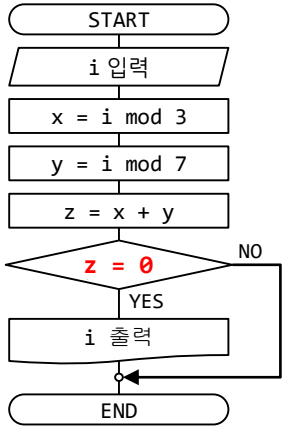
① a mod 3 + a mod 7 = 0
또는 a % 3 + a % 7 = 0

```
#include<stdio.h>
main() {
   int a;
   printf("정수를 입력하시오:");
   scanf("%d", &a);
   if(    ②    )
      printf("3과 7의 배수");
   else
      printf("3과 7의 배수×");
}
```

② a % 3 + a % 7 ==0 또는
a % 3 ==0 && a % 7 == 0

PATH 04 알고리즘 연습문제

[PATH 예상문제 1회]
정수를 입력 받아 3의 배수이면서 7의 배수인 수를 출력하는 프로그램이 있다. 아래의 괄호 안에 적합한 표현을 작성하시오.

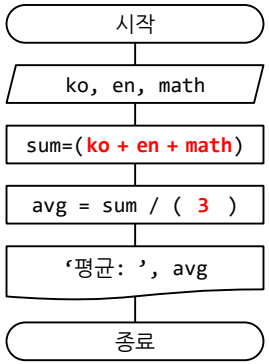



```
#include <stdio.h>
int main(void){
   int i,x,y;
   i = x = y = 0;
   scanf("%d",&i);

   x = i % 3;
   y = i % 7;
   z = x + y;

   if( z == 0 )
      printf("%d",i);
}
```

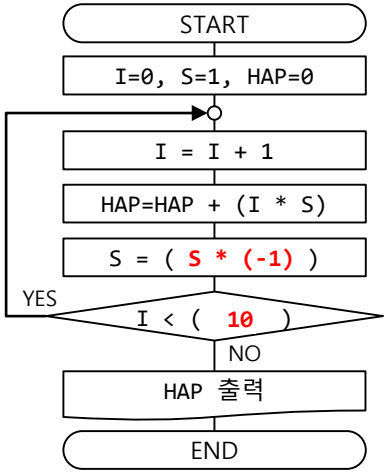
[PATH 예상문제 5회]
국어, 수학, 영어점수의 평균을 출력하는 프로그램이 있다. 아래의 괄호 안에 적합한 표현을 작성하시오.




```
import java.util.Scanner;
class function{
  public static void main (String[] args){
   Scanner input = new Scanner(System.in);
   int ko = input.nextInt();
   int en = input.nextInt();
   int math = input.nextInt();
   int sum, avg;
   sum = ( ko + en + math );
   avg = sum / (  3  ) ;
   System.out.println("평균:" + avg);  }
}
```

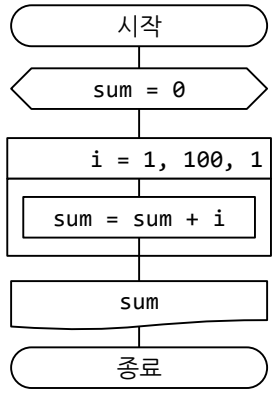
PATH 04 알고리즘 연습문제

[PATH 예상문제 2회]
다음은 1 - 2 + 3 … + 9 - 10 을 계산하는 순서도이다.
괄호 안에 적합한 표현을 작성하여 순서도를 완성하시오.



I		HAP		S:부호		I < 10	
단계	값	단계	값	단계	값	단계	값
①	0	①	0	①	1		
②	1	②	1	②	-1	②	Y
③	2	③	-1	③	1	③	Y
④	3	④	2	④	-1	④	Y
⑤	4	⑤	-2	⑤	1	⑤	Y
…		…		…		…	
⑩	9	⑩	5	⑩	-1	⑩	Y
⑪	10	⑪	-5	⑪	1	⑪	N

[2020년 2회 기출문제]
1부터 100까지의 합계를 계산하는 프로그램이 있다.
옆의 순서도를 참고하여, 괄호 안에 적합한 표현을 작성하시오.




```
class function{
 public static void main (String[] args){

    int sum=0;

    for( int i=1; i<=100; i++ ){
         sum = sum + i;
    }

    System.out.println(sum);
 }
}
```

정보처리기능사 실기

PATH 02. 프로그래밍

PATH 01 산술 연산자

1. 산술 연산자 종류

종류	역할
+	피연산자를 더한다. ex) 10 + 5 결과 = 15
-	피연산자를 뺀다. ex) 10 - 5 결과 = 5
*	피연산자를 곱한다. ex) 10 * 2 결과 = 20
/	피연산자를 나눈다. ex) 10 / 2 결과 = 5
%	피연산자의 나머지를 구한다. ex) 10 % 2 결과 = 0

2. 산술 연산자 우선 순위

- 프로그래밍 언어에서도 일반적으로 우리가 알고 있는 수학과 동일하게 연산자에 따른 우선순위가 결정됩니다.

우선 순위	1	2	3
연산자	(괄호)	* , /, %	+, -

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include<stdio.h>
int main(){
   int result1, result2;

   result1 = 10 + 15 % 4 - 20 % 9;
   result2 = 10 * 15 % 4 - 20 % 9 + 5;

   printf("결과 : %d %d", result1, result2);

   return 0;
}
```

A. 결과 : 11 5

PATH 02 관계연산자

1. 관계 연산자

- 두 가지의 피연산자를 비교하여,
  참(True)와 거짓(False)을 구분 하게 해주는 연산자입니다.

종류	예시
>, < (초과), (미만)	printf("%d", 10 > 3); // 결과 : 1(True)
>=, <= (이상), (이하)	printf("%d", 5 <= 5); // 결과 : 1(True)
== (같다)	printf("%d", 10 == 3); // 결과 : 0(False)
!= (다르다)	printf("%d", 5 != 3); // 결과 : 1(True)

▶Tip1 : 익숙하지 않은 **!= (다르다)**는 시험 전에 필수로 확인해주세요.
▶Tip2 : **!**는 **Not 연산자**로 사용되며, 논리값을 역전시켜 출력합니다.
ex> !(10 > 5) // 결과:0(False)

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include <stdio.h>
int main(void) {
    int num1 = 10; int num2 = 5;
    int num3 = 15; int num4 = 7;
    int result1, result2;

    result1 = num1 > num2;
    result2 = num1 < num2;

    printf("결과 : %d", result1 < result2);
    return 0;
}
```

A. 결과 : 0

PATH 03 논리연산자

1. 논리 연산자

- 다수의 조건식을 동시에 적용할 수 있는 연산자
- AND(&&)와 OR(||)을 사용

종류	의미	예시
&& (and)	두 가지의 조건이 모두 True일 경우, True 출력	10 < 5 && 5 < 10 // 결과 : 0
|| (or)	둘 중에 하나라도 조건이 True일 경우, True 출력	10 < 5 || 5 < 10 // 결과 : 1

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include <stdio.h>

int main(void) {
    int num1 = 21, num2 = 9;
    int num3 = 42, num4 = 12;

    printf("%d", num1 > num2 && num3 > num4);
    printf("%d", num1 > num2 && num3 < num4);

    printf("%d", num1 > num2 || num3 > num4);
    printf("%d", num1 > num2 || num3 < num4);
}
```

A. 결과 : 1011

PATH 04 C언어(삼항 연산자)

1. 삼항 연산자

- 조건에 부합할 경우, **True와 False에 해당하는 값을 출력**하는 구문을 의미합니다.

구문	예시

조건 ? True : False
// ? = 조건과 값 구분
// : = 값1과 값2 구분
	int a = 10; int b = 3;
	char result; // 출력 내용 : 단일문자
	result = num1 > num2 ? 'A' : 'B'
	printf("%c", result);
	// 'A' 출력

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include<stdio.h>
int main() {
   int num1, num2, num3, num4;
   int result;
   num1 = 7; num2 = 3;
   num3 = 4; num4 = 9;

   result = num1 < num2 && num3 < num4 ? num1 : num2;

   printf("%d", result);
   return 0;
}
```

A. 결과 : 3

PATH 05 조건문

1. if → else if → else문

- 조건에 대해 부합하게 되면 해당 실행문을 실행합니다.
- else의 경우, “그 외의”라는 뜻을 가지고 있으며, if와 else if로작성하지 않는 모든 조건에 대해 출력하게 됩니다.

구문

```
if ( 조건 ){ // 중괄호의 경우 두 줄 이상 시에 작성한다.
    실행문; // if 구문은 중괄호가 없을 시 하단 1줄까지 인식
}
else if( 조건 ){ // else if가 아닌 if를 입력할 경우,
    실행문;     상단의 if와 동시에 조건 만족이 모두 출력
}
else{ // if와 else if에 해당하지 않는 모든 False 값을 가짐
    실행문;  그렇기에 조건 작성 X
}
```

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include <stdio.h>
void main() {
    int num = 3;

    if(num >= 5)
        printf("Hello");
    else if(num >=3)
        printf("PATH");
    else
        printf("Program Lang");
    return 0;
}
```

A. 결과 : PATH

PATH 06 C언어(switch case)

1. switch case문

- 변수의 값을 확인하여, case의 값과 일치하게 되면
  해당 실행문을 인식

구문	예시

switch(변수){
case 값 :
실행문;
break;
default :
실행문;
break;
}
	int num = 2;
	switch(num){
	case 1 : // case1 : num ≠ 1
	printf(“Hello”);
	break;
	default : // 나머지 조건
	printf(“PATH”);
	break; // PATH 출력 후 종료
	}

- case 조건은 **콜론(:)**으로 구분하며, **break**로 처리를 종료합니다.
  // break : 해당 구문을 탈출하기 위해 사용합니다.
- **default**는 If문의 **else**와 같은 역할을 합니다

Q. 다음 C 언어로 구현된 프로그램을 분석하여
그 실행 결과를 쓰시오.

```
#include<stdio.h>
int main(){
   int num;
   num = 1;

   switch(num){
      case '1':
         printf("%d", num + 2); break;
      case '2':
         printf("%d", num + 1); break;
      case '3':
         printf("%d", num + 3); break;
      default:
         printf("%d", num); break;
   }
}
```

A. 결과 : 1

PATH 07 증감연산자

1. 증감연산자 (값을 증가, 감소 시키는 연산자)

- 전위연산자(++a / --a) : **피연산자 증감 후**, 해당 라인 연산 수행

```
ex) int a = 5; int b = 5;
    printf("%d %d", ++a, --b) // 결과 : 6 4
    pinrtf("%d %d", a, b)     // 결과 : 6 4
```

- 후위연산자(a++ / a--) : **해당 라인 연산 수행** 후, 피연산자 증감

```
ex) int a = 5; int b = 5;
    printf("%d %d", a++, b--) // 결과 : 5 5
    pinrtf("%d %d", a, b)     // 결과 : 6 4
```

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include<stdio.h>
void main(){
   int x=7;
   int y=7;
   int result;
   result = ++x + y--;

   printf("%d %d %d",result, x, y);
}
```

A. 결과 : 15 8 6

PATH 08 반복문(for문)

1. For문 (횟수 반복)

- **지정한 횟수**만큼 반복적으로 수행하기 위한 구문입니다.
- 다른 반복문들도 횟수반복을 할 수 있지만, for문의 경우에는 조금 더 작성하기 쉽게 되어 있습니다.

구문	예시

for(기준값; 조건문; 증감문){
실행문;
}
	(1) for(int i = 1; i < 10; i++)
	printf("Hello PATH");
	}
	(2) int i; // 변수 선언
	for(i = 1; i < 10; i++){
	printf("Hello PATH");
	}

Q. 다음 C 언어로 구현된 프로그램을 분석하여, 출력 결과를 작성하시오.

```
#include<stdio.h>
int main(){
   int i, j;
   for(i=2; i<=4; i++){
       for(j=5; j<=7; j++){
       }
   }

   printf("%d × %d = %2d", j, i, i*j);
   return 0;
}
```

A. 결과 : 8 × 5 = 40

PATH 09 반복문(while문)

1. while문 (조건 반복)

- **조건에 부합하는 동안** 반복적으로 수행하기 위한 구문입니다.**무한 루프(반복)**를 표현하기 가장 간단
- 해당 구문은 **조건식만 작성**하기에 지정 횟수로 반복하기 위해 변수와 증감을 따로 작성해주어야 합니다.

구문	예시

while(조건문){
실행문; }
	(1) int i = 0 // 변수 선언
	while(i < 10){
	printf("Hello PATH");
	i++; } // 증감문
	// 증감문 작성을 안할 시, 무한 루프
	(2) while(1){ // 1 = TRUE, 무한루프
	printf("Hello PATH"); }

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include<stdio.h>
int main()
{
   int i = 0;
   int sum = 0;

   while( i <= 6 ){
      sum = sum + i;
      i = i + 2;
   }
   printf("i = %d \nsum = %d", i, sum);

   return 0;
}
```

A. 결과 : i = 8
sum = 12

PATH 10 반복문(do while문)

1. do ~ while문(조건 부합X, 한번 반복)

- 기본적인 형태는 while문과 같이 **조건에 부합하는 동안** 반복적작업을 수행하나 **조건에 부합하지 않더라도 한번은 반복**하게 됩니다.

구문	예시

do{
실행문;
}while(조건문);
	int i = 10; // 변수 선언
	
	do{
	printf("Hello ");
	i--; // 증감식
	}while(i > 0); // 조건
	// 결과 : Hello 10번 출력

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.(5점)

```
#include<stdio.h>

int main()
{
    int i = 5;
    do{
        printf("%2d", i);
        i--;
    }while(i < 5 && i > 0);
    return 0;
}
```

A. 결과 : 5 4 3 2 1

PATH 11 반복문 제어

1. 반복문 제어

- 반복문 내에서 반복된 흐름을 벗어나기 위해 사용합니다.
- 일반적으로는 특정 조건에서 빠져나가기 위해 조건문 구문과 같이 사용한다.

구문	의미
break (멈추다)	해당 반복문을 멈추고, **탈출**하는데 사용합니다.
continue (계속하다)	반복문을 유지하면서, 다음 반복으로 **건너뜁니다.**

각각 해당 구문에 ①번이나 ②번을 삽입 했을 때의 결과값 확인.

```
ex) for(int i = 5; i >= 0; i--){
        if ( i % 2 == 0 )
            ① break; ② continue;
        printf("%d", i);        }
// ①의 결과 : 5, 해당 조건에 만족할 경우, 반복문이 끝나게 된다.
// ②의 결과 : 531, 해당 조건에 만족할 경우, 해당 구문을 건너뛰고
              다음 반복문 구문을 실행한다.
```

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include<stdio.h>
int main(){
   int a, sum;
   a = sum = 0;

   while( a <= 10 ){
       a += 1;
       if(a%2 == 1)
          continue;
       sum += a;
   }

   printf("sum = %d", sum);

   return 0;
}
```

A. 결과 : sum = 30

PATH 12 C언어(이항 연산자)

1. 비트(BInary digiT) - 2진수

- 컴퓨터가 알고 있는 유일한 언어로서 데이터의 최소 단위를 뜻한다.

보통 우리가 쓰는 단위를 10진수라고 표현하며, 컴퓨터의 언어를

이진수(비트)로 표현한다.

ex) 1 (10)-> 0000 0001(2) 2 (10) -> 0000 0010(2)
3(10) -> 0000 0011(2), 10(10) -> 0000 1010(2)

2. 비트 연산자(이항 연산자)

- 10진수의 값을 2진수로 변환하여 논리 연산을 하는 연산자

종류	예시	의미

& (and)	1 int a = 5;
	2 int b = 2;
	3 printf("%d", a&b);
		a = 0101
		b = 0010 // 값이 모두 1이면 1 출력
		출력 0000**(2)** -> 0**(10)**

| (or)	1 int a = 5; int b = 2; 2 printf("%d", a|b);	a = 0101 b = 0010 -> 0111(2) -> 7(2) // 둘 중 하나라도 1이면 1 출력
^ (xor)	1 int a = 5; int b = 3; 2 printf("%d", a^b);	a = 0101 b = 0011 -> 0110(6) // 값이 서로 다르면 1 출력

Q. 다음 C언어 프로그래밍 예제 코드의 결과를 입력 하시오.

```
#include <stdio.h>
int main(void){
   int num1, num2, result1, result2;
   num1 = 38;
   num2 = 11;
   result1 = num1 & num2;
   result2 = num1 | num2;
   printf("result1 = %d ", result1);
   printf("result2 = %d", result2);
}
```

A. 결과 : result1 = 2 result = 47

PATH 13 C언어(시프트 연산자)

1. 시프트 연산자

- 10진수의 값을 2진수로 변환하여 비트의 위치를 이동시키는 연산자를 의미합니다.

종류	의미
<< (Left)	우측 값만큼 비트의 위치를 좌측으로 이동
>> (Right)	우측 값만큼 비트의 위치를 우측으로 이동

```
ex ) int num = 5;  // 0000 0101
     int result = num << 2; // 0001 0100(20)
     printf("%d", result); // 20 출력
```

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include<stdio.h>
main()
{
   int num1, num2, result;
   num1 = 20;
   num2 = 4;

   result = num1 >> 3 << num2;

   printf("%d", result);
}
```

A. 결과 : 32

PATH 14 C언어(보수 연산자)

1. 보수(NOT) 연산자

- 보수(Complement)란 비트(Bit)를 통해 표현할 수 있는 것이 양수로 한정되기에, 음수를 표현하기 위해 사용하며, C언어의 경우 2의 보수를 출력한다.

1의 보수 계산 : 0을 1로, 1을 0으로 반전

ex> 0000 1100(+12) → 1111 0011(-12)

// **비트의 가장 처음 값은 부호**의 역할 **(0 = 양수, 1 = 음수)**

2의 보수 계산 : 1의 보수를 보완하기 위해 기존의 값에 1을 더한 값

ex> 1111 0011(-12) + 1 → (-12+1) = -13

// TIP : 답을 편하게 구하기 위해서 정보처리기능사 용도로 개편하여 실제 계산 방법과는 다름을 공지드립니다.

종류	의미
~ 값	2의 보수(NOT) 연산 작업

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include<stdio.h>
int main()
{
    int num1, num2, result;
    num1 = num2 = 25;
    result = num1 + ~num2;

    printf("%d %d %d", num1, ~num2, result);
}
```

A. 결과 : 25 -26 -1

PATH 15 복합대입연산자

1. 복합 대입 연산자

- 산술 연산자와 대입 연산자(=)를 간결하게 사용하는 작업입니다.

종류	예시	의미

+=	`int a = 5;`
	`a += 2;`
	`printf("%d", a); // 7`
		a += 2 → a = a + 2

-=	`a -= 2; // 3`	a -= 2 → a = a - 2
*=	`a *= 2; // 10`	a *= 2 → a = a * 2
/=	`a /= 2; // 2`	a /= 2 → a = a / 2
%=	`a %= 2; // 1`	a %= 2 → a = a % 2
&=	`a &= 2; // 0`	a &= 2 → a = a & 2
|=	`a |= 2; // 7`	a |= 2 → a = a | 2
^=	`a &= 2; // 7`	a ^= 2 → a = a ^ 2

Q. 다음 Java 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
class function{
   public static void main (String[] args) {

        int num = 13;

        num += 1;
        num -= 2;
        num *= 3;
        num /= 4;
        num %= 5;

        System.out.printf("%d", num);
    }
}
```

A. 결과 : 4

PATH 16 C언어(배열)

1. 배열(Array)

- 변수는 하나의 변수의 하나의 값을 가집니다. 필요 시에 새로 작성하게 됩니다.
- 배열이란 같은 자료형을 가진 값들을 하나의 변수에서 사용하는 집합과 같습니다.

Array [사용할 공간 선언] = { Value1, Value2...}
↳▶ 1부터 시작

의미

- 배열 선언 시 : 사용할 변수(공간)을 선언합니다.
▶ int num[2] = { 1, 2 }; // 2개의 공간 할당
// 중괄호를 통해 1과 2의 값을 지정

배열 출력 시 : 처음 시작점은 0을 기준으로 출력됩니다. ▶ int num[2] = { 1, 2 }; printf("%d", num[0]) → 1 출력 printf("%d", num[1]) → 2 출력

Q. 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
#include<stdio.h>

int main(){
   int temp;
   int num[5] = {1, 23, 42, 19, 50};

   for(int i = 0; i<5; i++)
      printf("%3d", num[i]);
}
```

A. 결과 : 1 23 42 19 50

PATH 17 재귀함수

1. 함수 (≒ JAVA : 메소드)

- 반복적인 코드의 사용을 방지하기 위하여, 하나의 항을 만들어서 필요할 때마다 해당 값을 호출하여 반환하는 역할을 수행합니다.

구문	예시

자료형 함수명 (매개변수)
{
실행문;
return 함수명;
}
 	
	int function(int num1,int num2){
	printf("%d %d", num1, num2);
	// 1 2 출력
	return function(num1, num2);
	// function 함수 재호출
	}
	int main(){
	function(1,2); // function 함수 호출
	}

Q. 다음 C 언어로 구현된 프로그램을 분석하여, 출력 결과를 작성하시오.

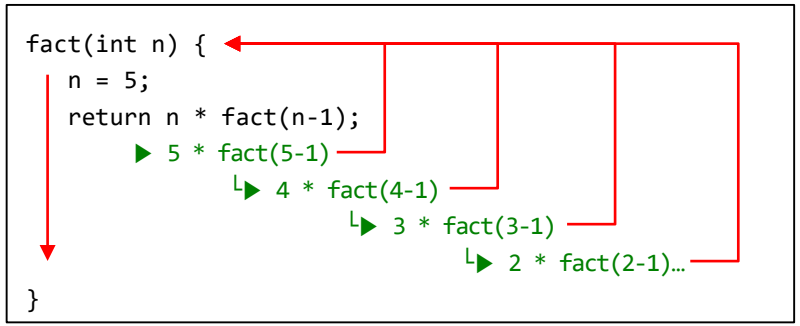
```
#include<stdio.h>
int hdr(int num){
   if(num <= 0)
      return;
   printf("%d ", num);
   hdr(num-1);
}
int main(){
   hdr(5);
   return 0;
}
```

A. 결과 : 5 4 3 2 1

PATH 18 Java(재귀 함수)

1. 재귀 함수

- 하나의 Method(함수)에서 **자기 자신을 다시 호출**하여 해당 구문을 다시 수행하는 함수입니다.



Q. 다음 Java 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오.

```
class function 1{
   public static void main (String[] args){
      int n = 5;
      System.out.println(fact(n));
   }
   public static int fact(int n){
      if ( n == 1 )
         return 1;
      return n * fact(n-1);
   }
}
```

A. 결과 : 120

PATH 19 C언어(헤더 파일)

1. 헤더(header)파일

- 헤더(head)파일이란 특정 기능을 가진 함수의 모음.
- 특정 값을 구하기 위해서는 해당 기능을 지원하는 함수를 가진 헤더파일을 사용해야 합니다.

2. 종류

헤더 파일	자주 사용하는 함수

#include<math.h>
(수학 관련)
	pow (값, 제곱) // 제곱
	sqrt (값) // 제곱근
	ceil (값) // 올림
	floor (값) // 내림
	fabs (값)
	// 부호 없는 정수

#include<stdlib.h>
(난수 및 메모리 할당)
	rand() // 1~ 약 33000 난수 출력
	srand(초기화 값)// 난수 초기화
	abs (값) // 부호 없는 정수

Q. 다음은 C언어로 구현된 25의 양의 제곱근 5를 출력하는 프로그램이다. 빈칸에 들어갈 라이브러리 함수를 작성하시오.

```
#include<stdio.h>
#include<math.h>
int main(void) {
   double x=25;
   printf("%g", (    ①    ));
}
```

A. 결과 : sqrt(x)

PATH 20 진수 입력 및 출력

1. 진수 변환 (2진수, 8진수, 10진수, 16진수)

- 프로그래밍에서는 기본적으로는 10진 형태로 숫자를 사용합니다.
- 다른 진수로 표현하기 위해서는 0b, 0, 0x와 같은 문자를 접두사로 표현해야 합니다.

ex) 2진수 = 0b1001
8진수 = 067
└ 파이썬 등의 언어에서는 0o 로 사용
16진수 = 0xA2

2. 진수 입,출력 (10진수, 8진수, 16진수)

- scanf 나 printf 에서 사용하는 진수 사용법으로는

ex) 10진수 = %d
8진수 = %o
16진수 = %x

Q. 다음 입력값을 토대로 각각의 진법으로 연산되어 출력되는 결과를 작성하시오.

```
#include <stdio.h>

int main(void) {

    int num = 0b11010;

    printf("10진수 : %d\n",num);

    printf("8진수 : %o\n",num);

    printf("16진수 : %x\n",num);

}
```

A. 10진수 : 26
8진수 : 32
16진수 : 1a

정보처리기능사 실기

PATH 03. 데이터베이스

PATH 01 데이터베이스 정의

데이터베이스 (Data Base) : 자료(Data) 의 모임

- 컴퓨터 시스템 내에서 다수의 사용자가 사용할 목적으로 체계화 시킨 데이터의 집합
- 특징은 실시간 접근성, 지속적인 변화, 동시 공유, 상호 참조, 데이터 논리적 독립성이 있다.

1) 데이터베이스 종류

종류	정의
MS SQL	Microsoft 사의 데이터 베이스 시스템(SQL Server)
ORACLE	리눅스(Linux) 서버 등에 자주 이용을 하며, 높은 구매 비용을 통한 사후 처리 시스템이 설비 되어 있는 중앙 집중 방식의 대용량 데이터베이스 시스템
My SQL	리눅스(Linux) 서버 등에 이용을 하며, 낮은 구매 비용과 무료 GUI 등을 지원

2) 데이터 베이스 설계

요구조건 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현

3) 데이터 베이스 특징

- 실시간 접근성(Real-time Accessibility)
- 계속적인 변화(Continuous Evolution)
- 동시 공유(Concurrent Sharing)
- 내용에 의한 참조(Content Reference)

스키마(Schema)

- 데이터베이스의 구조와 제약 조건에 관한 전반적인 명세

1) 스키마 종류

종류	정의
외부 스키마 (서브)	사용자 등의 개인적 입장에서 필요로 하는 데이터 베이스
개념 스키마 (논리)	사용자들이 필요로 하는 데이터를 기관이나 조직의 관점에서 정의한 데이터베이스
내부 스키마 (물리)	시스템 프로그래머나 설계자의 관점에서 정의하는 데이터베이스

PATH 02 데이터베이스 관리 시스템

DBMS(DataBase Management System)

관리자가 데이터베이스를 관리하는 시스템/프로그램

1. 파일시스템

- 파일에 이름을 부여하고 저장이나 검색을 위하여 논리적으로 그것들을 어디에 위치시켜야 하는지 등을 정의한 뒤 관리하는 데이터베이스 전 단계의 데이터 관리 방식

2. 망형 데이터베이스 관리 시스템 (NDBMS)

- 데이터의 구조를 네트워크 상의 망형 형태로 논리적으로 표현
- 트리구조나 계층형 데이터베이스 보다는 유연한 대응이 가능
- 설계가 복잡함

3. 계층형 데이터베이스 관리 시스템 (HDBMS)

- 상하 종속(부모 자식간의 형태)적인 관계로 계층화 하여 관리
- 데이터의 접근 속도가 빠르지만 구조에 유연한 대응이 어려움

4. 관계형 데이터베이스 관리 시스템 (RDBMS)

- 가장 보편화된 데이터베이스 관리 시스템
- 데이터를 저장하는 상하 관계로 표시하며 상관관계를 정리
- 오라클사에서 개발한 Oracle과 MS사에서 개발한 SQLserver (MsSQL), 썬 마이크로시스템에서 개발한 MySQL과 MySQL출신 개발자가 만든 Maria DB등이 있다.

5. NoSQL(non relational SQL)

- 전통적인 관계형 데이터베이스 보다 덜 제한적인 일관성 모델을 이용
- 빅 데이터와 실시간 웹 애플리케이션의 상업적 이용에 쓰임

DBMS 장단점

장점	단점
데이터 중복 최소화	데이터베이스 전문가 (DBA) 필요
데이터 공유(일관성 유지)	DBMS 구축 서버 필요 및 유지비
정합성, 부결성, 보안성 유지	데이터 백업과 복구 어려움
사용자 중심의 데이터 처리	시스템의 복잡성
데이터 표준화 적용 가능	대용량 디스크로 엑세스 집중 시 병목현상으로 과부하 발생
데이터 접근 용이	
데이터 저장 공간 공유로 인한 절약	대용량 데이터 처리 어려움

DBMS 주요 제품

Top 5 상용화 DBMS	Top 5 오픈소스 기반 DBMS
Oracle	MySQL
MS SQL Server	PostgreSQL
DB2	Mongo DB
Microsoft Access	Redis
Teradata	Elasticsearch

PATH 03 데이터베이스 용어

릴레이션 - 관계

- 데이터베이스 상에서 데이터를 표현하기 위해 체계화시킨 표를 의미

차수 : Degree

- 하나의 릴레이션 내에 들어 있는 속성의 수를 의미

기수 : Cardirality

- 하나의 릴레이션 내에 들어 있는 튜플의 수를 의미

속성 : Attribute

- 릴레이션 내의 하나의 열을 의미하며, 어떤 개체(Entity)를 표현하고 저장한 것을 말함, 흔히 칼럼(Column) 또는 필드(Filed)같은 의미로 사용

튜플 : Tuple

- 릴레이션 내의 하나의 행을 의미 하며 Record 또는 Row로 표현 하기도 함

1. 속성(Attribute)

1	2	3	4

2. 튜플(Tuple)

1			
2			
3			

트랜잭션 : Transaction

- 데이터베이스 내에서 하나의 작업 수행을 위한 연산들의 집합을 의미

ACID (트랜잭션의 안정성)
원자성(Atomicity), 일관성(Consistency), 독립성(Isolation), 지속성(Durability)

도메인 : Domain

- 릴레이션 내의 각각의 속성들이 가질 수 있는 값들의 집합을 의미

뷰(View)

- 하나 이상의 기본 테이블로부터 유도된 **가상의 테이블**을 입니다.
- 구조와 조작도 기본 테이블과 매우 유사합니다.

관계대수

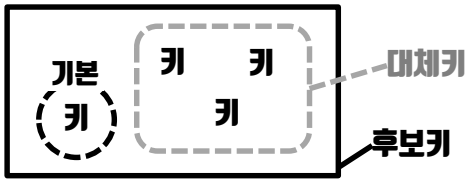
- 정보 유도 절차적 언어

관계해석

- 정보 유도 비절차적 언어

PATH 04 데이터베이스 용어

키(Key)

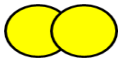
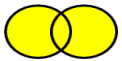
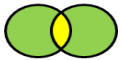
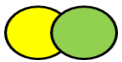


1. **후보키(Key) : 유일성 / 최소성 만족**
2. **기본키** : 후보키 중 선정(Not Null, 중복불가)
3. **슈퍼키 : 유일성 만족 / 최소성 불만족**
   - 후보 키가 없는 경우, 두 개 이상의 열을 연결하여 유일성을 만족시켜 후보 키가 되는 키
4. **대체키** : 기본 키로 선택되지 못한 후보 키
5. **외래키** : 다른 테이블의 행을 식별하는 키

이상(Anomaly)

- 데이터 중복성에 의해서, 릴레이션 조작 시 예기치 못한 곤란한 현상 또는 **데이터 불일치 현상**
- 종류 : **삽입이상 / 삭제이상 / 갱신이상**
- 이상을 해결하기 위해서는 정규화(Normalization)을 실행하여, 테이블 내의 데이터 중복을 제거합니다.

집합연산자

종류	형태	정의
UNION (합집합)		중복 행이 제거된 집합
UNION ALL (합집합)		중복 행이 제거되지 않은 집합
INTERSECTION (교집합)		두 쿼리 결과에 공통적으로 존재하는 집합
MINUS (차집합)		첫 쿼리에 있고 두번째 쿼리에는 없는 집합

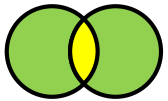
PATH 05 데이터베이스 용어

JOIN (조인)

- 두개이상의 테이블이나 데이터베이스를 연결하여 데이터를 검색하는 방법

1) INNER JOIN(내부 조인)

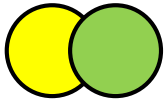
- 두 테이블에서 조인된 필드가 일치하는 행만을 표시



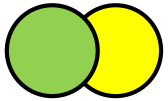
INNER JOIN

2) OUTER JOIN(외부조인)

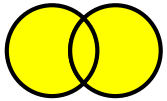
- 조인 조건에 만족하지 않는 튜플도 결과로 출력하기 위한 조인



LEFT
OUTER JOIN



RIGHT
OUTER JOIN



FULL
OUTER JOIN

데이터베이스 정규화

- 불필요한 데이터(Data Redundancy)를 제거하는 과정입니다.
- 논리적으로 데이터 저장합니다.

정규화 과정

비정규 릴레이션
↓
1NF
↓
2NF
↓
3NF
↓
BCNF
↓
4NF
↓
5NF

1. 도메인을 분해
2. 부분적 함수 종속 제거
3. 이행적 함수 종속 제거
4. 결정자 이면서 후보키가 아닌 것 제거
5. 함수 종속이 아닌 다치 종속 제거
6. 후보키를 통하지 않은 조인 종속성 제거

PATH 06 SQL

SQL - Structured Query Language

데이터베이스 시스템의 데이터를 관리하기 위해 설계된 특수 목적 프로그래밍 언어

DDL - Data Definition Language : 데이터 정의어

명령어	의미
CREATE	데이터베이스내의 릴레이션(테이블)을 생성하는 명령어 **ex) CREATE VIEW 뷰이름 AS SELECT 컬럼1, 컬럼2…FROM 테이블 WHERE 조건문** VIEW : 유도된 가상 테이블
ALTER	데이터베이스내의 릴레이션(테이블)의 칼럼을 변경하는 명령어 **ex) 1. ALTER TABLE 테이블명 ADD 컬럼명 데이터타입** ALTER ADD : ALTER문의 열 추가 **2. ALTER TABLE 테이블명 MODIFY 컬럼명 데이터타입** ALTER MODIFY : ALTER문의 타입 변경 **3. ALTER TABLE 테이블명 DROP 컬럼명** ALTER DROP : ALTER문의 열 삭제
DROP	데이터베이스내의 릴레이션(테이블)을 삭제하는 명령어 **ex) 1. DROP TABLE 테이블명 RESTRICT** RESTRICT : 참조 시 삭제 취소 **2. DROP TABLE 테이블명 CASCADE** CASCADE : 참조 삭제 **3. SELECT DISTINCT 컬럼명 FROM 테이블명** DISTINCT : 중복제거
TRUNCATE	릴레이션에서 모든 행을 삭제하는 명령어 **ex) 1. TRUNCATE TABLE 테이블명**

DML - Data Manipulation Language : 데이터 조작어

명령어	의미
SELECT	데이터베이스 내의 데이터를 조회 및 검색하는 명령어 **ex) 1. SELECT 컬럼명 From 테이블** WHERE : SQL 기본 조건문
INSERT	릴레이션 내에 데이터를 삽입하는 명령어 **ex) INSERT INTO 테이블명 VALUES 입력값1, 입력값2…**
UPDATE	릴레이션 내에 데이터를 수정하는 명령어 **ex) UPDATE 테이블명 FROM 컬럼명 = 수정값**
DELETE	릴레이션 내에 데이터를 삭제하는 명령어 **ex) DELETE [FROM] 테이블명**

DCL - Data Control Language : 데이터 제어어

명령어	의미
COMMIT	데이터베이스 내의 모든 작업의 완료를 의미하는 명령어
ROLLBACK	작업 중 문제가 발생할 시에, 처리과정의 변경 사항을 취소하는 명령어
GRANT	계정에 대한 권한을 부여하는 명령어 ex) GRANT - ON - TO
REVOKE	계정에 대한 권한을 삭제하는 명령어 ex) REVOKE - ON - FROM

PATH 07 SQL

SELECT 문의 문법순서와 실행 순서

SELECT	컬럼	GROUP BY	그룹화할 컬럼
FROM	테이블	HAVING	함수
WHERE	조건	ORDER BY	컬럼 정렬방법

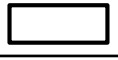
- 문법 순서(작성순서)

SELECT → FROM → WHERE → GROUP BY →
HAVING → ORDER BY

- 실행 순서

FROM → WHERE → GROUP BY → HAVING →
SELECT → ORDER BY

E-R Diagram : 개체(Entity)와 관계(Relation)를 도식화한 표현

기호	의미
	개체집합(Entity)
	속성 (Attribute)
	관계(Relation)
	연결(link)

연산 표기법

종류	설명	a + (b *c) 표현 예시
전위 (Prefix)	연산자를 먼저 쓰고, 피연산자를 뒤에 쓰는 표기법	+ a * b c
중위 (Infix)	실생활에 사용되는 방식과 동일하게 쓰는 표기법	a + (b *c)
후위 (Postfix)	피연산자를 먼저 쓰고, 연산자를 맨 뒤에 쓰는 표기법	a b c * +

정보처리기능사 실기

PATH 04. 운영체제

PATH 01 운영체제

운영체제 (OS)

- 하드웨어와 사용자 사이에서 하드웨어 및 소프트웨어의 자원 등을 관리를 도와주는 플랫폼

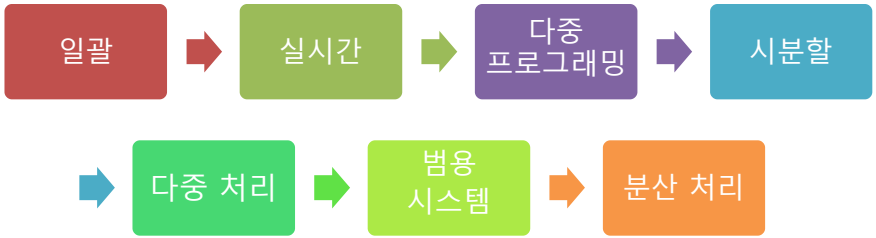
1) 운영체제 종류

종류	정의
DOS	문자 입력 인터페이스인 CLI (Command Line Interface) 의 대표적 운영체제
WIN	마이크로소프트가 개발한 선점형 멀티태스킹 운영체제 그래픽 사용자 인터페이스인 GUI (Graphic User Interface) 사용
UNIX	주로 서버에서 사용하는 시분할 시스템을 지원하는 다중 사용자, 멀티 태스킹의 운영체제
LINUX	유닉스 운영체제를 기반으로 만들어진 운영체제 기본적인 형태는 동일하나 오픈 소스 운영체제로 사용자가 소스 코드에 자유로운 접근이 가능하다.

2) 운영체제의 목적

처리 능력(Throughput)	일정한 단위 시간 내에 처리할수 있는 일의 양
반환 시간(Turn Around Time)	하나의 작업을 시작한 시간에서부터 결과를 얻을 때까지 걸린 시간
사용 가능도(Availability)	시스템을 이용하려 할 때 얼마나 빨리 사용할 수 있는 정도
신뢰도(Reliability)	시스템이 주어진 문제를 정확하게 해결할수 있는지의 정도

3) 운영체제의 발전 과정



PATH 02 운영체제 특징

운영체제 특징 및 용어

1) 윈도우

용어	의미
선점형 멀티태스킹	운영체제가 수행 중인 각 프로그램의 실행시간을 할당하고 실행중 문제가 발생하면 해당 프로그램을 강제 종료시키고, 모든 시스템 자원을 반환하는 멀티태스킹 운영방식
PNP (Plug & Play)	장치와 장치를 연결할 시에, 별도의 사용자 조작이 필요 없이 자동으로 장치를 인식하여 사용할 수 있는 것
핫 플러그	시스템 전원이 켜져 있는 상태에서 장치를 추가하는 것은 가능하나, 제거하는 것은 불가능
핫 스왑	핫 플러그와 다르게 전원이 켜져 있는 상태로 장치의 교체가 가능
가상화	단일 호스트에서 서로 다른 운영체제를 구동할 수 있게 하는 기능
하이퍼바이저	단일 호스트에서 다른 운영체제를 가상으로 구동 지원하는 플랫폼 대표적으로 VRWARE사의 VMWARE, ORCLE사의 Virtualbox 등
FAT 파일 시스템	파일 할당 테이블(File Allocation Table)을 의미하며, 운영체제를 운용하는데 디스크 공간을 할당
NTFS 파일 시스템	Windows가 현재 사용중인 압축 파일 시스템. 이전 버전의 FAT에 비해 신뢰성, 성능 개선 등 디스크 공간을 효율적으로 사용 가능
ReFS 파일 시스템	MS 에서 NTFS를 대체하기 위해 개발한 차세대 파일 시스템

2) UNIX 구성

- **커널 :** 운영체제의 핵심적인 구성요소 중 하나로서 프로세스 수행에 필요한 하드웨어의 성능 등 을 조정할 할 수 있도록 이어주는 서비스를 제공
- **쉘 :** 실행한 프로세스 등을 커널에게 전달할 수 있도록 명령을 번역해주는 명령 번역기
- **유틸리티 :** 운영체제에서 제공하는 것이 아닌 그 외의 실행 가능한 프로그램

3) UNIX 용어

용어	의미
시분할 시스템	사용자에게 컴퓨터의 자원을 시간에 따른 분할을 시스템으로 사용자와 컴퓨터간의 대화를 통해 작업을 처리하는 시스템
UFS 파일 시스템	유닉스 파일 시스템(Unix File System)으로 유닉스 및 유닉스 기반의 운영체제 등에서 쓰이는 디스크 기반의 파일 시스템
아이노드 (i-node)	정규 파일, 디렉토리 등의 파일 시스템을 보유한 유닉스 시스템 및 유닉스 계열의 운영체제에서 사용하는 자료 구조 시스템
소프트 링크 (심볼릭 링크)	유닉스 계열 운영체제에서 사용되는 기능이며, i-node를 이용하여 Windows의 바로가기와 동일하게, 링크 파일 삭제 시 원본을 유지시키는 링크 시스템
하드 링크	소프트 링크와는 다르게 원본과 동기화된 바로가기 기능이며, 링크 파일 삭제 시에 원본 파일도 삭제되는 특징이 있다.

PATH 03 운영체제 단축키 및 명령어

1) 윈도우 단축키

+ E	윈도우 탐색기 실행	+ D	바탕화면 표시
+ R	윈도우 실행창 실행	+ L	사용자 전환 / 윈도우 잠금
+ F	특정 파일 및 폴더 등 찾기	+ A	윈도우 알림 센터
+ M	열려 있는 창 최소화	+ Pause	시스템 구성 요소(정보) 확인
+ S	윈도우 검색 창 실행	+ X	윈도우 시스템 관리 메뉴
+ X	윈도우 설정 실행	+ I	윈도우 제어판
+ P	프로젝트 창 실행 (다른 화면에 표시 항목을 선택 듀얼모니터나 프로젝터 설정)		
+ Shift + M	최소화된 창을 복원		
+ Shift + S	캡쳐도구 실행 (영역 캡쳐)		
+ Tab	테스크 바 (Task bar)실행 , 실행중인 모든 앱 타임라인 보기		
+ Ctrl + D	가상 데스크톱을 추가		
+ Ctrl + F4	현재 사용 중인 가상 데스크톱 닫기		
+ Shift + ← , → (왼쪽 화살표키, 오른쪽 화살표키)	실행 중인 창을 다른 모니터로 이동		

2) DOS & UNIX 명령어

기능	윈도우 / MS-DOS	리눅스
경로 변경	cd	cd
목록 출력	dir	ls
파일 복사	copy	cp
구조 복사	xcopy	cp
디렉토리 생성	mkdir	mkdir
하위파일 삭제	del	rm
권한 설정	attrib	chmod
화면 표시	type	cat
파일 복사	copy	cp
목적지 까지 경로	tracert	traceroute
프로세스 종료		kill
실행중 프로세스 표시		ps
디렉토리 경로 표시		pwd
네트워크 상태 점검		ping
접속해 있는 사용자 표시		who

PATH 04 프로세스

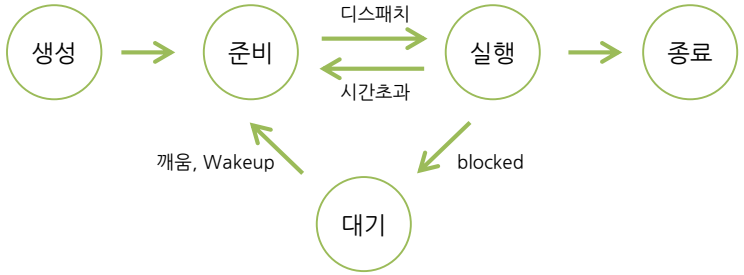
프로세스(Process)

- 프로그램은 파일 형태로 저장하여 관리되다가 실행을 시키면 동작을 하게 된다.
  이때 실행 중인 프로그램을 프로세스(Process)라고 하며, 작업(Job) 또는 태스크(Task)라고도 한다.

1) 프로세스 상태

생성 (Create)	실행 (Running)	준비 (Ready)	대기 (Waiting)	종료 (Terminated)

2) 프로세스 상태 전이



1. 디스패치(Dispatch Process) : *ready → running*
- 준비 상태에서 실행 상태로 바뀌는 것
2. 보류(Block Process) : *running → blocked*
- 프로세스 실행 중 시간이 다 되기도 전에 입출력 동작을 해야 할 경우 프로세스가 CPU를 반납하고 보류 상태로 들어가는 것
3. 깨움(Wakeup Process) : *blocked → ready*
- 보류 상태 이후 다시 준비 상태로 넘어가는 과정
4. 시간제한 (Timeout Process) : *running → ready*
- 클락 인터럽트를 통해 프로세스가 일정 시간만 점유할 수 있게 하는 것.

PATH 05 스케줄링 및 가상화

2) 프로세스 스케줄링

- 프로세스를 효율적으로 실행될 수 있도록 여러 자원들 사이의 우선순위를 관리하는 작업
- 선점형 : 이미 할당되어 실행중인 프로세스라도 강제로 빼앗아 선택하여 사용할 수 있음
- 비선점형 : 이미 실행중인 프로세스를 강제로 빼앗아 사용할 수 없음

	종류	정의
선점형	RR (라운드 로빈)	순서대로 시간단위로 CPU를 할당하는 방식, 자원 사용에 제한된 시간이 있고 할당된 시간이 지나면 자원을 반납
선점형	SRT Shortest Remainning Time First	SJF의 기법의 장점을 최대화 하여 실행시간이 가장 짧은 시간을 우선적으로 처리하는 방식
선점형	다단계 큐	프로세스를 그룹으로 분류할 수 있을 경우 그룹에 따라 각기 다른 준비상태 의 큐를 사용하는 기법
선점형	다단계 피드백 큐	다단계 큐를 보완하여 프로세스 큐들 간의 이동이 허용될 수 있도록 개선한 기법
비선점형	FIFO = FCFS	선입선출 방식(먼저 들어오면 먼저 나가는 방식) 우선순위에 상관없이 먼저 도착한 프로세스 부터 처리
비선점형	SJF Shortest Job First	CPU점유 시간이 가장 짧은 프로세스를 먼저 할당하는 처리
비선점형	HRN	SJF기법을 보완하여 대기 시간과 실행시간을 이용하여 처리 (대기시간 + 실행시간) / 실행시간
비선점형	우선순위	준비상태의 프로세스마다 우선순위를 부여하여 가장 높은 프로세스 부터 할당하는 기법

가상화(Virtualization)

- 물리적 하드웨어 등을 사용자로부터 은폐하여, 대체품으로 논리적인 리소스를 제공하거나 하나의 물리적인 리소스를 여러 개로 보이게 하는 기술이다.

1) 가상화 종류

구분	정의
호스트OS 형	물리적 하드웨어 위에 OS를 설치하여 가상화 소프트웨어와 가상머신을 이용하는 방식 ex) 기존 OS 위에 가상화 OS를 설치
하이퍼바이저 형	HOST OS를 필요하지 않는 방식 직접 물리 하드웨어를 움직여 독립한 호스트 같이 행동 ex) 가상화를 위한 독립적 OS를 직접 설치
컨테이너 형	컨테이너라는 가상화 소프트웨어를 이용하여 사용하는 방식 ex) 미리 구축해 놓은 가상화 OS(컨테이너) 를 설치

PATH 06 클라우드 및 리눅스 디렉토리 구조

클라우드 (Cloud Computing)

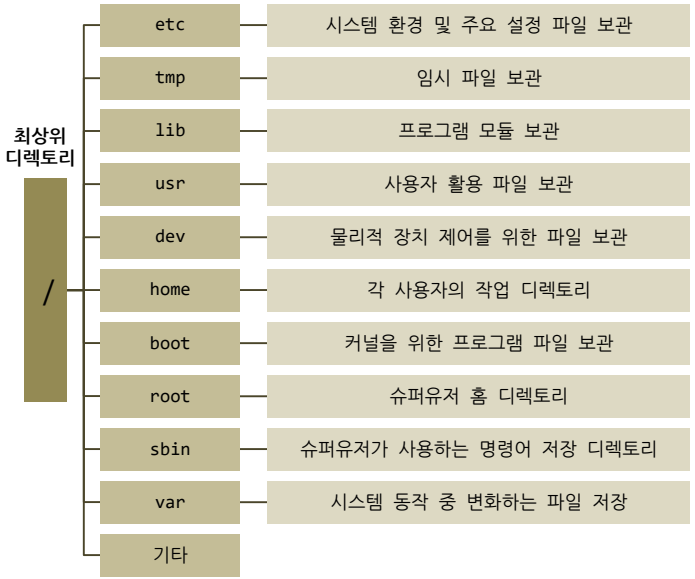
- 인터넷 기반에서 구동되는 컴퓨팅 기술, 정보를 자신의 컴퓨터가 아닌 클라우드(인터넷)에 연결된 다른 컴퓨터로 처리하는 기술

1) 클라우드 서비스 종류

구분	정의
IaaS 인프라형	사용자가 관리할 수 있는 범위가 가장 넓은 클라우드 컴퓨팅 서버 OS, 미들웨어, 런타임 , 데이터 등 직접 구성하고 관리할 수 있다.
PaaS 플랫폼형	미리 구축된 플랫폼을 이용하는 클라우드 컴퓨팅 관리상의 자유도가 낮지만 개발 자체의 집중할 수 있는 장점이 있다.
SaaS 서비스형	소프트웨어 및 관련 데이터는 중앙 호스팅이 되어 사용자는 제공해 주는 소프트웨어를 서비스 받아 사용

리눅스 디렉토리

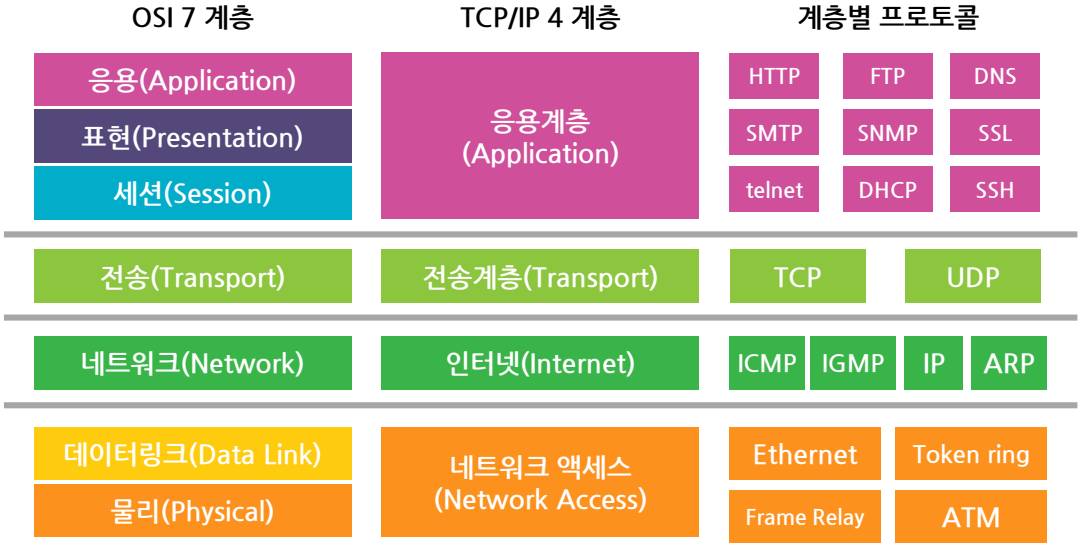
- 리눅스 디렉토리란 윈도우의 폴더 와 같은 개념 기본적으로 윈도우에서는 ₩ 또는 역슬래시를 사용하고 리눅스에서는 / (슬래시)를 사용



정보처리기능사 실기

PATH 05. 네트워크

PATH 01 네트워크 계층별 정의



OSI 7 계층 특징

계층	정의
응용	파일 전송, 원격 접속, 메일 서비스 등 응용 서비스를 담당하며 여러 가지 서비스(인터페이스)를 제공하는 계층
표현	송신자 측에서 수신자 측에 맞는 형태로 데이터를 변환(번역)하고, 수신측 에서는 다음 계층에 맞는 형태로 변환
세션	응용 프로그램 간의 통신에 대한 제어 구조를 제공하기 위해 응용프로그램 간의 접속을 연결, 유지, 종료 시켜주는 역할을 수행하는 계층
전송	프로토콜 (TCP, UDP) 과 관련된 계층으로 오류제어 및 흐름제어 등을 담당하며, 두 시스템 간을 연결하여 신뢰성 있는 데이터 전송
네트워크	논리적 주소(IP 주소)를 이용 최적의 경로를 선택하며, 데이터가 전송될 수신측 주소를 확인하여 일치하면 다음 계층으로 전송
데이터링크	오류와 흐름을 제거하여 신뢰성 있는 데이터를 물리적 링크를 통해 프레임 단위로 데이터를 신뢰성 있게 전송하는 계층 물리적 주소(MAC 주소)를 관리
물리	실제 장비들을 연결하기 위한 기계적, 전기적, 기능적, 절차적 특성을 정의

PATH 02 프로토콜

프로토콜

- 타 기종의 기기들이 서로 원활히 통신할 수 있도록 표준화시켜 놓은 통신 규약
- 기본요소로는 구문(Syntax), 의미(Semantics), 시간(Timing)이 존재

HTTP	웹(인터넷) 상에서 데이터를 주고 받을 수 있는 프로토콜
FTP	네트워크를 통해 컴퓨터들간의 파일을 교환하기 위한 프로토콜 ※ Anonymous FTP는 익명성을 보장합니다.
Telnet	원격 통신에 이용되는 프로토콜의 하나로서, 데이터의 전송 시에 암호화 작업이 존재하지 않아 보안성이 낮은 프로토콜 (포트번호 : 23)
SSH	원격 통신에 이용되는 프로토콜의 하나로서, 데이터의 전송 시에 암호화 기법을 통해 보안성이 높은 프로토콜 (포트번호 : 22)
rlogin	Unix와 같은 시스템에서 같은 네트워크 상에서 사용되는 원격 통신 프로토콜 포트번호 : 513
SOAP	HTTP, HTTPS 등을 통해 이기종(Heterogeneous)과 정보통신을 XML 기반의 데이터를 컴퓨터 네트워크 상에서 교환하는 프로토콜
TCP	데이터를 주고 받는 과정에서 데이터를 검수하는 작업을 통해 서로 확인하는 과정을 거쳐 정상적으로 데이터 누락 등을 확인할 수 있는 신뢰성 있는 데이터 전송 프로토콜
UDP	데이터를 주고 받는 과정에서 확인하는 과정을 거치지 않아 신뢰성이 떨어지는 프로토콜로서, 확인하는 과정을 거치지 않기에, TCP보다 데이터 전송이 빠름
DHCP	각종 TCP/IP 프로토콜 및 IP 주소 등을 자동적으로 클라이언트가 제공 및 사용할 수 있도록 해주는 프로토콜 (유동 IP 설정)
ARP	Address Resuloution Protocol의 약자로서 IP(논리적 주소)에 대응되는 이더넷카드의 MAC(물리적 주소)를 검색하여 변환해주는 프로토콜입니다.
RARP	ARP와 다르게 MAC(물리적주소)를 IP(논리적 주소)로 변환해주는 프로토콜입니다.
SMTP	전자 우편 송신 프로토콜로서, 사용자의 전자 우편을 받아서 타사용자의 계정으로전송해주는 프로토콜
POP3	전자 우편 수신 프로토콜로서, 전송 받은 전자 우편을 저장하고 있다가, 사용자에게보내주는 역할을 하는 프로토콜

PATH 03 네트워크장비 및 프로세스

허브(Hub)
다수의 장치들을 하나로 연결하는 장치이며, 연결된 장치끼리 상호 간의 통신을 할 수 있는 장비, 더미허브와 스위치허브가 있다.

더미허브: 데이터를 단순히 연결, 성형구조
스위치허브 : 데이터의 유무 및 흐름을 제어 하는 지능형 허브

리피터 (Repeater)
허브 등으로 연결하여 장거리로 통신할 경우 감쇠된 신호를 수신하여 증폭한 후, 다음 구간으로 재전송

브리지 (Bridge)
소프트웨어적인 방법을 통해 프레임(데이터)를 목적지 주소를 기준으로 1:1로 연결하여 전송

스위치 (Switch)
브리지의 확장된 개념으로 기본 기능은 브리지와 동일하지만 하드웨어적 방식을 사용하여 ,전송 중 패킷의 충돌이 일어나지 않도록 목적지 주소를 기준으로 해당 포트들과 1:n으로 연결

라우터 (Router)
네트워크 계층에서 서로 다른 구조의 망을 연결 가능하여, LAN과 MAN, WAN 등을 연결하는데 사용하며 데이터 전송의 최적 경로를 선택할 수 있다.

게이트웨이 (Gateway)
다른 종류의 네트워크 등을 상호 접속하여 정보를 주고 받을 수 있는 장치

네트워크 핵심 알고리즘

1. 패킷 교환 : 작은 블록인 패킷을 이용하여 데이터를 교환하는 방식

X.25	전기 통신 국제기구인 ITU-T에서 관리 감독하는 프로토콜 대용량의 데이터를 다수의 패킷으로 분리하여 송신하며, 수신측에서 다수의 패킷을 결합하여 원래의 데이터로 복원한다.
프레임 릴레이	ISDN을 사용하기 위한 프로토콜로 OSI 1~2계층에서만 동작하며 전용선보다 저렴하고 유연한 대역폭을 할당한다
ATM	비동기 전송모드라고 하는 광대역 전송에 쓰이고 동기화를 맞추지 않아 보낼 데이터가 없는 사용자의 슬롯은 다른 사람이 사용할 수 있도록 하여 네트워크 상의 효율성을 높였다. OSI 7계층과는 다른 고유한 참조모델을 가짐

2. 서킷 교환 : 네트워크 리소스를 독점하여 사용하도록 하는 교환방식

3. 라우팅 알고리즘 : 목적지까지의 최적 경로를 산출하여 교환하는 방식

거리 벡터 알고리즘	라우터간의 최단 경로를 찾고 그 최적의 경로를 찾을수 없을 경우 다른 경로를 찾는 알고리즘
링크 상태 알고리즘	라우터간의 모든 경로를 파악한 뒤 대체 경로를 사전에 마련해 두는 알고리즘

※. 라우팅 프로토콜의 종류

RIP	최초의 라우팅 프로토콜 거리 벡터 알고리즘 사용 30초주기	OSPF	링크 상태 알고리즘 기반으로 최단 경로를 찾는 프로토콜
IGRP	RIP를 개선하기 위해 생성 네트워크 상태를 고려하여 라우팅	BGP	규모가 큰 네트워크 연결 대형 사업자간의 라우팅

PATH 04 네트워크장비 및 IP주소

IP 주소

구분	IPv4	IPv6
주소 길이	32bit	**128bit**
표시 방법	8비트씩 4부분 (10진수)	16비트씩 8부분 (16진수)
주소 개수	약 43억 개	약 43억X43억X43억X43억 개
주소 할당	A,B,C 등 클래스 단위의 비순차적 할당	네트워크 규모 및 단말기 수에 따른 순차적 할당
품질 제어	지원 수단 없음	등급별, 서비스별로 패킷 구분
보안 기능	Ipsec 프로토콜 별도 설치	확장 기능에서 기본으로 제공
PnP	지원 수단 없음	지원 수단 있음

네트워크 클래스(Classful network)

네트워크 단말의 증가로 가용 가능한 IPv4의 주소가 부족해졌고,

이에 **사용 목적에 따라 IP의 대역대를 나누어** 각 규모에 따라 관리하기 쉽게 표현한 것

Class	IP주소의 첫째 옥탯	사용 목적	이론적 IP 주소 범위
A class	0xxx xxxx	대륙안통신	0.0.0.0~ 127.255.255.255
B class	10xx xxxx	국가안통신	128.0.0.0 ~ 191.255.255.255
C class	110x xxxx	기업안통신	192.0.0.0 ~ 223.255.255.255
D class	1110 xxxx	그룹통신	224.0.0.0 ~ 239.255.255.255
E class	1111 xxxx	연구용통신	240.0.0.0 ~ 255.255.255.255

정보처리기능사 실기

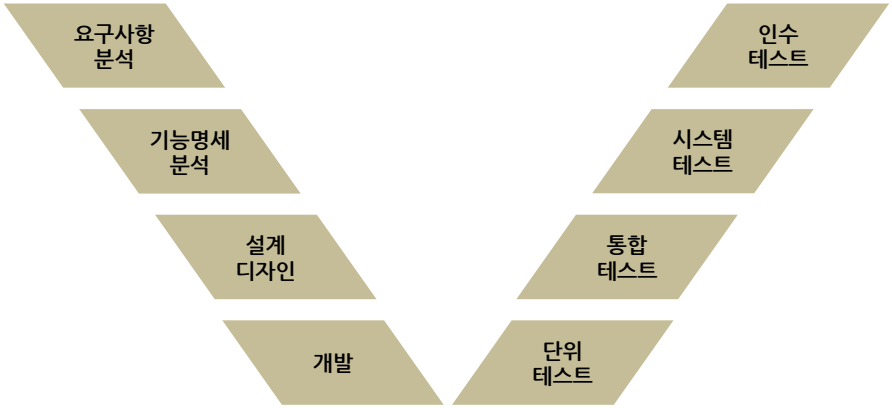
PATH 06. 애플리케이션

PATH 01 애플리케이션 테스트

테스트

- 개발된 응용 애플리케이션이나 시스템이 사용자가 요구하는 기능과 성능, 사용성, 안전성 등을 확인하고, 노출되지 않은 숨어있는 결함을 찾아내는 활동

테스트 모델



소프트웨어 생명 주기의 V-모델

테스트의 7가지 원칙

1) 테스트는 결함이 존재함을 밝히는 활동이다.
2) 완전한 테스트는 불가능
3) 초기 테스트
4) 결함 집중
5) 살충제 패러독스
6) 테스트는 정황의 의존적이다.
7) 오류-부재의 궤변

살충제 패러독스 (Pesticide Paradox)

- 동일한 테스트 케이스에 의한 반복적 테스트로 새로운 버그를 찾지 못하는 내성 현상
- 애플리케이션 테스트에서도 동일한 테스트에 대한 비정상적인 결함 검수가 이루어질 수 있어서 이러한 현상을 방지하기 위해서는 다양한 방법으로 테스트 하는 것이 필요

파레토 법칙(80/20법칙)

- 빌프레도 파테로가 '이탈리아 인구의 20%가 이탈리아 전체 부의 80%를 가지고 있다'는 발언에서 시작한 법칙
- 발견된 80%의 오류는 20% 모듈에서 발견되므로, 20%의 모듈을 집중적으로 테스트

소프트웨어 아키텍처(Software Architecture)

- 소프트웨어의 골격이 되는 기본 구조
- 구성요소(Component) 간의 관계를 표현하는 시스템 구조/구조체
- 설계 기본원리 : 모듈화 / 추상화 / 단계적 분해 / 정보은닉

PATH 02 애플리케이션 테스트

프로젝트 수행 단계에 따른 분류

1. 인수 테스트
- 개발된 제품에 대해 운영 여부를 결정하는 테스트
- 사용자와 이해관계자 등이 실제 업무 적용 전에 수행
- 사용자인수 / 운영상 인수 / 계약인수 / 규정인수 테스트
  알파테스트 / 베타 테스트

2. 단위 테스트
- 테스트 가능한 단위(컴포넌트,모듈 등) 내의 결함 검색 및 기능 검증
- 구조기반 / 명세기반 테스트

3. 통합 테스트
- 모듈 사이의 인터페이스, 통합된 컴포넌트 간의 상호 작용 테스트
- 빅뱅 / 상향식 / 하향식 / 샌드위치 테스트 등

4. 시스템 테스트
- 통합된 단위 시스템의 기능의 시스템에서의 정상 수행 여부 테스트
- 기능적 요구사항과 비기능적 요구사항을 검사
- 성능 및 장애 테스트

테스트 기법에 따른 분류

1) 블랙박스 테스트
- 사용자 관점(내부 구조나 작동원리를 모르는 상태)으로
  명세(요구사항과 결과물의 일치)기반의 테스트 방법입니다.
- 종류
  - 균등분할(동치분해)
  - 한계값(경계값)테스트
  - 원인효과그래프테스트
  - 비교테스트

2) 화이트박스 테스트
- 개발자 관점으로 내부구조와 동작을 테스트합니다.
- 종류
  - 기초경로테스트
  - 제어흐름테스트
  - 조건테스트
  - 루프테스트
  - 데이터흐름테스트
  - 분기테스트

PATH 03 애플리케이션 테스트

통합 테스트 수행 방법

구분	스텁	드라이버
시기	상위 모듈은 있지만 하위 모듈이 없을 때, 하위 모듈로의 테스트를 진행	상위 모듈 없이 하위 모듈만 있을 때, 상위 모듈로의 테스트를 진행
방식	하향식 테스트	상향식 테스트
특징	작성이 쉬운 시험용 모듈	존재 하지 않는 상위 모듈간의 인터페이스 역할

테스트 케이스

- 특정 요구사항 준수 여부를 확인하기 위해 개발된 입력 값, 실행 조건, 예상된 결과의 집합, 미리 설계하면 오류를 방지할 수 있고 테스트 수행에 필요한 인력, 시간 등의 낭비를 축소

*** 테스트 케이스 작성 절차**

1. 계획 검토 및 참조 문서 수집
2. 내부 검토 및 우선순위 결정
3. 요구사항 정의
4. 테스트 설계와 방법 결정
5. 테스트 케이스 정의
6. 테스트 케이스 타당성 확인 및 유지보수
7. 테스트 수행

PATH 04 애플리케이션 테스트

테스트 커버리지

- 주어진 테스트 케이스에 의해 수행되는 소프트웨어의 테스트 범위를 측정하는 테스트 품질 측정 기준이며, 테스트의 정확성과 신뢰성을 향상시키는 역할을 수행한다.

구분	의미
기능 기반	전체 기능을 모수로 설정하고 실제 수행된 기능의 수를 측정하는 방법, 100% 달성을 목표로 하며, 화면 수를 모수로 사용할 수도 있다.
라인	전체 소스 코드의 Line 수를 모수로 측정하는 방법 단위테스트에서는 라인 커버리지를 척도로 삼기도 한다.
구문	코드 구조 내의 모든 구문에 대해 한 번 이상 수행하는 테스트 커버리지
조건	결정 포인트 내의 모든 개별 조건식에 대해 수행하는 테스트 커버리지
결정	모든 분기문에 대해 수행하는 테스트 커버리지
변형 조건/결정	조건과 결정을 복합적으로 고려한 측정 방법으로 다른 개별적인 조건식의 결과에 상관없이 독립적으로 전체 조건식의 결과에 영향을 주는 테스트 커버리지

테스트 목적에 따른 테스트 종류

구분	의미
회복 (Recovery)	시스템에 실패를 유도하고 시스템의 정상적 복귀 여부를 테스트하는 기법
안전 (Security)	불법 소프트웨어가 접근하여 파괴하지 못하도록 소스 코드 내의 보안 결함을 미리 점검하는 기법
강도 (Stress)	과다 정보량을 입력하여 과부하 시에도 정상적으로 작동하는지를 검증하는 기법
성능 (Performance)	사용자의 실행에 응답하는 시간, 시간 내에 처리하는 업무량, 요구에 반응하는 속도 등을 측정하는 기법
구조 (Structure)	내부 논리 흐름에 따라 테스트 케이스를 작성하고 결함을 발견하는 기법
회귀 (Regression)	변경 또는 수정된 코드에 새로운 결함 여부를 평가하는 기법
병행 (Parallel)	변경된 시스템과 기존 시스템에 동일한 자료를 입력후 결과를 비교하는 기법

PATH 05 애플리케이션 테스트

소프트웨어 형상관리(Software Configuration Management)

- 소프트웨어의 변경 과정, 처리 상태를 기록 및 보고하며, 부합하는 해당 사항에 대하여 추적, 통제하여 관리하여 품질 향상 및 안전성을 높이는 효과를 얻을 수 있습니다.

도구	설명
CVS	가장 오래된 형상관리 도구 중 하나로 서버는 단순한 명령 구조를 가진 장점이 있지만 텍스트 기반의 코드만 지원하는 단점이 있다.
SVN	CVS의 단점을 보완해 현재 가장 대중화된 도구 중 하나로 다양한 GUI 도구가 존재하고 압축을 통해 서버의 공간을 절약하는 장점을 가지고 있다.
Git	리눅스 커널 개발을 위해 만든 형상 관리 시스템이다 CVS와 SVN의 단점을 모두 보완하는 장점은 있으나 중앙집중형이 아닌 분산형 방식으로 스스로 저장공간이 필요하며 개념이 다르므로 개발자에게 학습할 시간이 필요하다.

유스케이스(Use case)

- 시스템의 동작을 사용자의 입장에서 표현한 시나리오
- 시스템에 관련한 요구사항을 알아내는 과정

테스트 오라클

- 테스트의 결과가 참인지 거짓인지를 판단하기 위해서 사전에 정의된 참 값을 대입하여 비교하는 소프트웨어 테스트 기법 및 활동

a. 참 오라클(True Oracle)

- 모든 입력 값의 기대 결과를 생성해서 발생된 오류를 모두 검출
- 전수 테스트 가능

b. 샘플링 오라클(Sampling Oracle)

- 특정한 입력 값들에 대해서만 기대하는 결과를 제공
- 전수테스트가 불가할 경우 사용

c. 휴리스틱 오라클(Heuristic Oracle)

- 샘플링 오라클에 휴리스틱(추정값) 입력값을 추가
- 확률/직관에 의한 예상결과

d. 일관성 검사 오라클(Consistent Oracle)

- 애플리케이션 변경 시, 수행 전과 후의 값의 동일 여부 검증

PATH 06 결함

결함

- 프로그램과 명세서 간의 차이, 업무 내용 불일치
- 개발자가 설계한 의도와 다른 동작 및 결과를 발생시키는 것

1. 에러(Error)
- 소프트웨어 개발 / 유지 보수 중에 발생한 부정확한 결과
- 개발자 실수, 개발 명세서의 잘못된 이해, 서브루틴 기능 오해 등

2. 오류(Fault)
- 프로그램 코드 상에 존재하는 것
- 정상/비정상적인 프로그램 버전간의 차이로 인하여 발생

3. 실패(Failure)
- 정상/비정상적인 프로그램 버전간의 실행 결과의 차이
- 실제 실행 결과를 개발 명세서의 예상 결과와 비교함으로써 발견

4. 결함(Defect)
- 버그, 에러, 오류, 실패, 프로그램 실행에 대한 문제점, 프로그램 개선 사항 등의 전체를 포괄하는 용어이다.

결함 관리 프로세스 7과정

구분	의미
결함 관리 계획	전체에 대한 계획을 수립하는 단계
결함 기록	발견된 결함을 관리 DB에 등록
결함 검토	주요 내용을 검토하고 수정할 내용을 개발자에게 전달
결함 수정	전달 받은 결함을 수정
결함 재확인	결함을 수정후 다시 테스트 수행
결함 상태 추적 및 모니터링 활동	관리 DB를 이용 대시보드 또는 게시판의 형태로 서비스제공
최종 분석 및 보고서 작성	결함에 대한 정보와 이해관계자들의 의견이 반영된 보고서 작성하고 결함 관리 종료

PATH 07 결함

결함의 상태 및 추적 프로세스

상태	내용
Fixed	개발자가 결함을 수정한 상태
Assigned	분석 및 수정을 위해 결함이 개발자에게 할당된 상태
Open	결함이 보고되었지만 아직 분석되지 않은 상태
Closed	결함 수정여부를 확인하고, 회귀 테스트 시 결함이 발견되지 않은 상태. 수정된 결과가 만족스럽지 않을 경우, 결함의 상태를 'Open' 으로 변경
Deferred	우선순위가 낮게 분류되었기 때문에 수정을 연기한 상태
Clarified	보고된 결함이 프로젝트 팀에 의해 비결함으로 판단된 상태

결함 심각도(Defect Severity)

- 애플리케이션에 발생한 결함이 전체 시스템에 미치는 영향의 척도

High	▪ 프로세스를 더 이상 진행할 수 없도록 만드는 결함 ▪ 핵심 요구사항 미구현, 장시간 시스템 응답 지연, 시스템 다운 등
Medium	▪ 시스템 흐름에 영향을 미치는 결함 ▪ 부정확한 기능, 부정확한 업무 프로세스, 데이터 베이스 에러, 보안 관련 오류 등
Low	▪ 시스템 흐름에는 영향을 미치지 않는 결함이지만. 상황에 맞지 않는 결과나 화면구성에 대한 결함 ▪ 부정확한 GUI, 에러 메시지 미출력, 화면상 표시 오류

PATH 08 결함

단위 테스트 기법

1. xUnit

- 다양한 코드중심의 테스트 프레임워크
- 소프트웨어의 함수, 클래스 등 서로 다른 구성 단위를 검사
- Junit(JAVA), CppUnit(C++), Nunit(.NET)

2. Mock 테스트

- 특정 기능 또는 모듈에 대한 응답 결과를 미리 정의해 놓고 테스트

유형	내용
Dummy	객체의 전달에만 사용되고 실제로는 사용되지 않음 주로 매개 변수 목록을 채우는 데 쓰임
Fake	실제로 동작하도록 구현되지만, 보통 빠른 구현을 위해 실제 환경과는 다르게 구현할 수 있다.
Stubs	테스트를 위해 미리 준비한 응답만을 제공하는것 그 외의 상황에 대해서는 정상적인 작동을 하지 못하는 것 스텁은 호출에 대한 정보를 기록하는 경우도 있다.
Mocks	스펙을 통해 정의된 응답을 받고 다른 응답을 받을 경우 예외를 발생하도록 구현되어 있으며, 응답에 대한 확인을 수행하는 역할

프로그램 코드 검토 기법

소프트웨어 인스펙션

- 코드 인스펙션 외에도 설계 및 설계 산출물 까지 포괄하여 소프트웨어 인스펙션으로 부르기도 한다.
- 개발소스 코드를 분석해 표준에 위배되거나, 규칙에 맞지 않게 구현된 부분을 수정하는 작업
- 자동화된 도구를 사용하여 개발 및 상용 소프트웨어의 결함을 발견 및 수정하게 되며, 해당 프로그램의 품질을 향상시키기 위해 사용된다.

워크 스루(walk-through)

- 코드의 품질을 평가하고, 개선하기 위한 목적으로 수행되는 검토 기법으로 검토회의
- 프로그램이나 설계서의 오류를 검토 과정에서 발견하기 위하여 시행하는 회의 오류의 조기 검출을 목적으로 하며 발견된 오류는 문서화시키고 미리 제공된 자료들을 정해진 절차에 따라 평가

인스펙션과 워크스루의 차이점

- 인스펙션은 워크스루와 달리 체크리스트를 기반으로 검토하며 발견된 모든 결함을 제거해야 한다는 특징이 있다. 완성도가 기준 이상일 때 수행함으로써 모든 결함을 없애는 데 주요 목적이 있다.