

# 프로그래밍 언어 활용

1장 / 서버 프로그램 구현

2장 / 프로그래밍 언어 활용

3장 / 응용 SW 기초 기술 활용



# 1 장

## 서버 프로그램 구현

121 개발 환경 구축 **B** 등급

122 서버 개발 **C** 등급

123 보안 및 API **B** 등급

124 배치 프로그램 **B** 등급

125 패키지 소프트웨어 **C** 등급

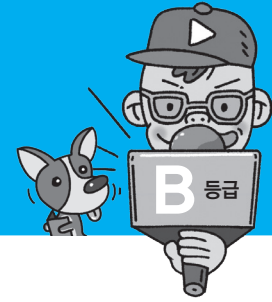


이 장에서 꼭 알아야 할 키워드 **Best 10**

1. 웹 서버 2. 웹 애플리케이션 서버 3. 개발 언어 선정 기준 4. 프레임워크 5. 소프트웨어 개발 보안  
6. API 7. 배치 프로그램 8. 스프링 배치 9. Quartz 10. 패키지 소프트웨어

# SECTION 121 LIVE

## 개발 환경 구축



### 전문가의 조언

소프트웨어 개발 시 구축해야 할 하드웨어 및 소프트웨어에는 어떤 것들이 있는지, 그리고 각 요소들의 개별적인 기능은 무엇인지 잘 파악해 두세요.

#### 정적 파일(Static File)

정적 파일은 인터넷 브라우저와 같은 클라이언트에서 별도의 처리 과정 없이 다운로드 하여 사용자에게 보여주는 파일로, HTML, CSS, 이미지 파일 등이 있습니다.

#### 동적 서비스(Dynamic Service)

동적 서비스는 사용자의 입력에 따라 다른 결과를 보여주는 서비스를 의미합니다. 쇼핑물을 예로 들면, 상품들을 인기순으로 정렬하기 위해 클릭을 했을 때 나오는 화면은 미리 만들어져 있는 페이지가 아닌 클릭한 순간 상품들을 정렬하여 페이지를 구성한 후 표시하는 동적인 화면입니다.

### 전문가의 조언

데이터베이스 서버는 데이터베이스와 DBMS가 갖는 특징과 동일한 기능을 가집니다. 데이터베이스와 DBMS에 대한 자세한 설명은 Section 035를 참조하세요.

## 1 개발 환경 구축의 개요

개발 환경 구축은 응용 소프트웨어 개발을 위해 개발 프로젝트를 이해하고 소프트웨어 및 하드웨어 장비를 구축하는 것을 의미한다.

- 개발 환경은 응용 소프트웨어가 운영될 환경과 유사한 구조로 구축한다.
- 개발 프로젝트의 분석 단계의 산출물을 바탕으로 개발에 필요한 하드웨어와 소프트웨어를 선정한다.
- 하드웨어와 소프트웨어의 성능, 편의성, 라이선스 등의 비즈니스 환경에 적합한 제품들을 최종적으로 결정하여 구축한다.

## 2 하드웨어 환경

하드웨어 환경은 사용자와의 인터페이스 역할을 하는 클라이언트(Client) 그리고 클라이언트와 통신하여 서비스를 제공하는 서버(Server)로 구성된다.

- 클라이언트에는 PC, 스마트폰 등이 있다.
- 서버는 사용 목적에 따라 웹 서버, 웹 애플리케이션 서버, 데이터베이스 서버, 파일 서버 등으로 나뉜다.
  - 웹 서버(Web Server) : 클라이언트로부터 직접 요청을 받아 처리하는 서버로, 저용량의 정적 파일\*들을 제공한다.
    - 예 Apache HTTP Server, Microsoft Internet Information Service, Google Web Server 등
  - 웹 애플리케이션 서버(WAS; Web Application Server) : 사용자에게 동적 서비스\*를 제공하기 위해 웹 서버로부터 요청을 받아 데이터 가공 작업을 수행하거나, 웹 서버와 데이터베이스 서버 또는 웹 서버와 파일 서버 사이에서 인터페이스 역할을 수행하는 서버이다.
    - 예 Apache Tomcat, IBM WebSphere, Oracle WebLogic 등
  - 데이터베이스 서버(DB Server) : 데이터베이스와 이를 관리하는 DBMS를 운영하는 서버이다.
    - 예 MySQL Server, Oracle Server, Microsoft SQL Server 등
  - 파일 서버(File Server) : 데이터베이스에 저장하기에는 비효율적이거나, 서비스 제공을 목적으로 유지하는 파일들을 저장하는 서버이다.
    - 예 AWS S3 등



## 웹 서버(Web Server)의 기능

HTTP/HTTPS* 지원	브라우저로부터 요청을 받아 응답할 때 사용되는 프로토콜
통신 기록 (Communication Log)	처리한 요청들을 로그 파일로 기록하는 기능
정적 파일 관리 (Managing Static Files)	HTML, CSS, 이미지 등의 정적 파일들을 저장하고 관리하는 기능
대역폭 제한 (Bandwidth Throttling)	네트워크 트래픽의 포화를 방지하기 위해 응답 속도를 제한하는 기능
가상 호스팅 (Virtual Hosting)	하나의 서버로 여러 개의 도메인 이름을 연결하는 기능
인증 (Authentication)	사용자가 합법적인 사용자인지를 확인하는 기능

## 3 소프트웨어 환경

소프트웨어 환경은 클라이언트와 서버 운영을 위한 시스템 소프트웨어와 개발에 사용되는 개발 소프트웨어로 구성된다.

- 시스템 소프트웨어에는 운영체제(OS), 웹 서버 및 WAS 운용을 위한 서버 프로그램, DBMS 등이 있다.
- 개발 소프트웨어에는 요구사항 관리 도구, 설계/모델링 도구, 구현 도구, 빌드 도구, 테스트 도구, 형상 관리 도구 등이 있다.
  - 요구사항 관리 도구 : 요구사항의 수집과 분석, 추적 등을 편리하게 도와주는 소프트웨어
    - 예 JIRA, IBM DOORS, inteGREAT, Reqtify, Trello 등
  - 설계/모델링 도구 : UML(통합 모델링 언어)\*을 지원하며, 개발의 전 과정에서 설계 및 모델링을 도와주는 소프트웨어
    - 예 DB Designer, PlantUML, ArgoUML 등
  - 구현 도구 : 개발 언어를 통해 애플리케이션의 실제 구현을 지원하는 소프트웨어
    - 예 Eclipse, IntelliJ IDEA, Visual Studio, Netbeans, Node.js 등
  - 빌드 도구 : 구현 도구를 통해 작성된 소스의 빌드 및 배포, 라이브러리 관리를 지원하는 소프트웨어
    - 예 Ant, Gradle, Maven, Jenkins 등
  - 테스트 도구 : 모듈들이 요구사항에 적합하게 구현되었는지 테스트하는 소프트웨어
    - 예 CppUnit, JUnit, HttpUnit, NUnit, SpringTest 등

A

B

C

D

### HTTP/HTTPS (HyperText Transfer Protocol [Secure])

HTTP는 하이퍼텍스트 문서를 전송하기 위해 사용하는 프로토콜이고, HTTPS는 HTTP에 보안 모듈을 결합시킨 프로토콜입니다.

### UML(Unified Modeling Language)

UML은 시스템 분석, 설계, 구현 등 시스템 개발 과정에서 시스템 개발자와 고객 또는 개발자 상호 간의 의사소통이 원활하게 이루어지도록 표준화된 대표적인 객체지향 모델링 언어입니다. UML에 대한 자세한 내용은 Section 009를 참조하세요.

#### 형상 관리 도구

형상 관리를 다른 말로 버전 관리라고도 합니다. 자세한 내용은 Section 047을 참조하세요.

- 형상 관리 도구\* : 산출물들을 버전별로 관리하여 품질 향상을 지원하는 소프트웨어

예) GIT, CVS, Subversion, Mercurial 등



#### 개발 언어의 선정 기준

개발 언어를 선정할 때는 다음과 같은 5가지 특성이 고려되어야 합니다.

적정성	개발하려는 소프트웨어의 목적에 적합해야 합니다.
효율성	코드의 작성 및 구현이 효율적이어야 합니다.
이식성	다양한 시스템 및 환경에 적용이 가능해야 합니다.
친밀성	개발 언어에 대한 개발자들의 이해도와 활용도가 높아야 합니다.
범용성	다른 개발 사례가 존재하고 여러 분야에서 활용되고 있어야 합니다.



#### 기출문제 따라잡기

Section 121

출제예상

1. 소프트웨어 개발을 위해 개발 환경을 구축하고자 할 때 고려해야 할 사항이 아닌 것은?

- ① 프로젝트 분석 단계에서 정리된 요구사항들을 고려하여 소프트웨어와 하드웨어 설비를 선정한다.
- ② 소프트웨어가 운영될 환경과 유사한 구조로 구축한다.
- ③ 하드웨어 환경은 서버와 네트워크로 구성된다.
- ④ 비즈니스 환경에 적합한 제품들을 선정하여 구축한다.

하드웨어 환경은 클라이언트(Client)와 서버(Server)로 구성됩니다.

출제예상

2. 개발 환경 구축 시 갖추어야 할 소프트웨어 환경에 대한 설명으로 잘못된 것은?

- ① 요구사항의 수집과 분석 및 추적을 위한 관리 도구를 구축해야 한다.
- ② HTML, CSS, 이미지 등을 처리할 웹 서버를 구축해야 한다.
- ③ 소프트웨어 개발을 위한 구현 도구를 구축해야 한다.
- ④ 산출물들을 버전별로 관리할 형상 관리 도구를 구축해야 한다.

소프트웨어 환경은 소프트웨어 개발 도구들로 구성되어 있습니다. 보기 중 도구와 관련 없는 내용을 찾아보세요.

출제예상

3. 웹 응용 소프트웨어 개발과 관련하여 사용자에게 동적 서비스를 제공하는 서버는?

- ① 웹 서버(Web Server)
- ② 웹 애플리케이션 서버(WAS)
- ③ 데이터베이스 서버(DB Server)
- ④ 파일 서버(File Server)

웹 서버는 정적 파일을, 웹 애플리케이션 서버는 동적 서비스를 처리한다는 것을 기억해 두세요.

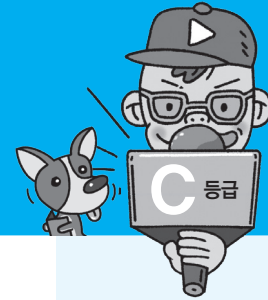
출제예상

4. 개발 환경 구축 시 고려해야 할 요소들에 대한 설명으로 잘못된 것은?

- ① 개발 환경은 크게 하드웨어 환경과 소프트웨어 환경으로 구분된다.
- ② 성능, 편의성, 개발자의 이해도 등 다양한 요소들을 고려하여 환경을 구축한다.
- ③ 시스템 소프트웨어와 개발 소프트웨어 등은 소프트웨어 환경에 속한다.
- ④ 하드웨어 환경에는 WAS, OS, DBMS 등이 속한다.

OS와 DBMS는 시스템 소프트웨어입니다.

▶ 정답 : 1. ③ 2. ② 3. ② 4. ④



## 1 서버 개발의 개요

서버 개발은 웹 애플리케이션의 로직을 구현할 서버 프로그램을 제작하여 웹 애플리케이션 서버(WAS)에 탑재하는 것을 의미한다.

- 웹 애플리케이션 서버에 구현된 서버 프로그램은 웹 서버로부터 받은 요청을 처리하여 결과를 반환하는 역할을 수행한다.
- 서버 개발에 사용되는 프로그래밍 언어에는 Java, JavaScript, Python, PHP, Ruby 등이 있다.
- 각 프로그래밍 언어에는 해당 언어로 서버 프로그램을 개발할 수 있도록 지원하는 프레임워크\*가 있다.

## 2 서버 개발 프레임워크

서버 개발 프레임워크는 서버 프로그램 개발 시 다양한 네트워크 설정, 요청 및 응답 처리, 아키텍처 모델 구현 등을 손쉽게 처리할 수 있도록 클래스나 인터페이스를 제공하는 소프트웨어를 의미한다.

- 서버 개발 프레임워크에 따라 지원하는 프로그래밍 언어가 제한적이므로 선정할 수 있는 프레임워크도 제한적이다.
- 서버 개발 프레임워크의 대부분은 모델-뷰-컨트롤러(MVC)\* 패턴을 기반으로 개발되었다.
- 프레임워크의 종류는 다음과 같다.

프레임워크	기능
Spring	JAVA를 기반으로 만들어진 프레임워크로, 전자정부 표준 프레임워크의 기반 기술로 사용되고 있다.
Node.js	JavaScript를 기반으로 만들어진 프레임워크로, 비동기 입·출력 처리와 이벤트 위주의 높은 처리 성능을 갖고 있어 실시간으로 입·출력이 빈번한 애플리케이션에 적합하다.
Django	Python을 기반으로 만들어진 프레임워크로, 컴포넌트의 재사용과 플러그인화*를 강조하여 신속한 개발이 가능하도록 지원한다.
Codeigniter	PHP를 기반으로 만들어진 프레임워크로, 인터페이스가 간편하며 서버 자원을 적게 사용한다.
Ruby on Rails	Ruby를 기반으로 만들어진 프레임워크로, 테스트를 위한 웹 서버를 지원하며 데이터베이스 작업을 단순화, 자동화시켜 개발 코드의 길이가 짧아 신속한 개발이 가능하다.



### 전문가의 조언

서버 개발의 의미와 관련 프레임워크들을 숙지하세요. 서버 프로그램을 구현하는 과정은 1, 2과목에서 학습했던 모듈 구현 과정과 동일하다는 것을 염두에 두고 가볍게 읽어보세요.

### 프레임워크(Framework)

프레임워크는 사전적으로 '뼈대', '골조'를 의미하는데, 소프트웨어에서는 특정 기능을 수행하기 위해 필요한 클래스나 인터페이스 등을 모아둔 집합체를 말합니다.



### 전문가의 조언

서버 개발 프레임워크는 웹 프레임워크라고도 불리며, 이러한 프레임워크는 기능 구현을 위한 기본적인 형태를 지원할 뿐 필수적인 요소는 아닙니다. 하지만 존재 여부에 따라 생산성의 차이가 크다는 점을 기억해 두세요.

### 모델-뷰-컨트롤러(MVC)

모델-뷰-컨트롤러는 시스템을 세 부분으로 분리하여 서로 영향 받지 않고 개발할 수 있는 아키텍처 패턴을 의미합니다. 자세한 내용은 Section 020을 참조하세요.

### 플러그인화

플러그인화는 재사용과 비슷한 의미로 전원 플러그처럼 마음대로 꺾다 뺏다할 수 있다는 것을 의미합니다.



#### 전문가의 조언

1. 2과목에서 다루었던 모듈에 대해 복습한다는 생각으로 읽어보세요. 모듈에 대한 자세한 내용은 Section 023을, 공통 모듈에 대한 자세한 내용은 Section 024를 참조하세요.

- **루틴(Routine)** : 기능을 가진 명령들의 모임
- **메인 루틴(Main Routine)** : 프로그램 실행의 큰 줄기가 되는 것
- **서브 루틴(Subroutine)** : 메인 루틴에 의해 필요할 때마다 호출되는 루틴
- **결합도(Coupling)** : 모듈 간에 상호 의존하는 정도 또는 두 모듈 사이의 연관 관계를 의미합니다.
- **응집도(Cohesion)** : 정보 은닉 개념을 확장한 것으로, 명령어나 호출문 등 모듈의 내부 요소들의 서로 관련되어 있는 정도, 즉 모듈이 독립적인 기능으로 정의되어 있는 정도를 의미합니다.

#### 재사용(Reuse)

재사용은 비용과 개발 시간을 절감하기 위해 이미 개발된 기능들을 파악하고 재구성하여 새로운 시스템 또는 기능 개발에 사용하기 적합하도록 최적화 시키는 작업입니다.

#### 다형성(Polymorphism)

다형성은 메시지에 의해 객체(클래스)가 연산을 수행하게 될 때 하나의 메시지에 대해 각각의 객체(클래스)가 가지고 있는 고유한 방법(특성)으로 응답할 수 있는 능력을 의미합니다. 다형성에 대한 자세한 내용은 Section 022를 참조하세요.

### 3 서버 프로그램 구현

서버 프로그램은 응용 소프트웨어와 동일하게 모듈 및 공통 모듈을 개발한 후, 모듈들을 통합하는 방식으로 구현된다.

- 모듈은 모듈화를 통해 분리된 시스템의 각 기능들로, 서브 루틴\*, 서브시스템, 소프트웨어 내의 프로그램, 작업 단위 등과 같은 의미로 사용된다.
- 모듈 개발 시 기능적 독립성을 고려하여 다른 모듈과의 과도한 상호작용을 배제함으로써 특정 모듈의 수정이 다른 모듈들에게 영향을 미치지 않아야 한다.
- 모듈의 독립성은 결합도(Coupling)\*와 응집도(Cohesion)\*에 의해 측정되며, 독립성을 높이려면 모듈의 결합도를 약하게 하고 응집도를 강하게 하며 모듈의 크기를 작게 만들어야 한다.
- 공통 모듈은 여러 프로그램에서 재사용(Reuse)\*할 수 있는 모듈을 의미하며, 자주 사용되는 계산식이나 매번 필요한 사용자 인증 같은 기능들이 공통 모듈로 구성될 수 있다.



#### 프레임워크의 특성

모듈화(Modularity)	프레임워크는 캡슐화를 통해 모듈화를 강화하고 설계 및 구현의 변경에 따른 영향을 최소화함으로써 소프트웨어의 품질을 향상시킵니다.
재사용성(Reusability)	프레임워크는 재사용 가능한 모듈들을 제공함으로써 개발자의 생산성을 향상시킵니다.
확장성(Extensibility)	프레임워크는 다형성(Polymorphism)*을 통한 인터페이스 확장이 가능하여 다양한 형태와 기능을 가진 애플리케이션 개발이 가능합니다.
제어의 역흐름(Inversion of Control)	개발자가 관리하고 통제해야 하는 객체들의 제어를 프레임워크에 넘김으로써 생산성을 향상시킵니다.





## 기출문제 따라잡기

Section 122

출제예상

## 1. 서버 개발 프레임워크에 대한 설명으로 옳지 않은 것은?

- ① 서버 프로그램 개발 시 사용할 수 있는 다양한 클래스 및 인터페이스의 집합체를 의미한다.
- ② 서버 개발 프레임워크가 없어도 생산성에 큰 영향은 없다.
- ③ 네트워크 설정, 요청 및 응답 처리, 아키텍처 모델 등 다양한 모듈을 제공한다.
- ④ 주로 모델-뷰-컨트롤러(MVC) 패턴을 기반으로 개발되었다.

서버 개발 프레임워크는 서버 개발을 도와주는 소프트웨어인데 생산성에 영향을 주지 않을까요?

출제예상

## 2. 서버 개발에 사용되는 언어와 프레임워크의 연결이 잘못 짝지어진 것은?

- ① JavaScript - Spring
- ② Python - Django
- ③ PHP - Codeigniter
- ④ Ruby - Rails

스프링(Spring) 프레임워크하면 JAVA라는 것 잊지마세요.

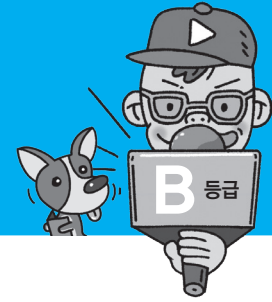
출제예상

## 3. 다음 중 프레임워크의 특성에 해당하지 않는 것은?

- ① 모듈화(Modularity)
- ② 캡슐화(Capsulation)
- ③ 확장성(Extensibility)
- ④ 제어의 역흐름(Inversion of control)

방금 공부한 내용인데요 기억나지 않는다면 왼쪽에 있는 표를 다시 한 번 공부하세요.

▶ 정답 : 1. ② 2. ① 3. ②



## 전문가의 조언

서버 프로그램 구현 과정에서 고려해야 하는 보안 요소와 API에 대해서 공부합니다. 보안에 대한 자세한 내용은 5과목에서 다루어지므로 여기서는 어떤 내용들이 있는지 가볍게 확인하고 넘어가면 됩니다. API는 1과목에서 많이 사용되었던 용어입니다. 무엇이었는지 복습하는 마음으로 읽어보세요.

기밀성(Confidentiality), 무결성(Integrity), 가용성(Availability)은 보안의 3대 요소로 불립니다. 자세한 내용은 Section 179를 참조하세요.

## 소프트웨어 개발 보안 가이드

소프트웨어 개발 보안 가이드는 안전한 소프트웨어 개발을 위해 정부에서 제작하여 배포하고 있는 지침으로, 한국인터넷진흥원 사이트(kisa.or.kr)에서 다운받을 수 있습니다.



## 전문가의 조언

보안 점검 항목들은 Section 180부터 187까지 각각의 섹션으로 자세하게 설명하고 있습니다. 여기서는 각 항목들이 무엇을 의미하는지만 간단히 읽어보세요.

## 1 소프트웨어 개발 보안의 개요

소프트웨어 개발 보안은 소프트웨어 개발 과정에서 발생할 수 있는 보안 취약점을 최소화하여 보안 위협으로부터 안전한 소프트웨어를 개발하기 위한 일련의 보안 활동을 의미한다.

- 소프트웨어 개발 보안은 데이터의 기밀성\*, 무결성\*, 가용성\*을 유지하는 것을 목표로 한다.
- 정부에서 제공하는 소프트웨어 개발 보안 가이드\*를 참고하여 소프트웨어 개발 과정에서 점검해야 할 보안 항목들을 점검한다.

## 2 소프트웨어 개발 보안 점검 항목

소프트웨어 개발 보안 점검 항목은 소프트웨어 개발의 각 단계에서 점검되어야 할 보안 항목들을 말한다.

세션 통제	<ul style="list-style-type: none"> <li>• 세션은 서버와 클라이언트의 연결을 말하며, 세션 통제는 세션의 연결과 연결로 인해 발생하는 정보를 관리하는 것을 의미한다.</li> <li>• 보안 약점에는 불충분한 세션 관리, 잘못된 세션에 의한 정보 노출 등이 있다.</li> </ul>
입력 데이터 검증 및 표현	<ul style="list-style-type: none"> <li>• 입력 데이터에 대한 유효성 검증체계를 갖추고, 검증 실패 시 이를 처리할 수 있도록 코딩하는 것을 의미한다.</li> <li>• 보안 약점에는 SQL 삽입, 경로 조작 및 자원 삽입, 크로스사이트 스크립트(XSS) 등이 있다.</li> </ul>
보안 기능	<ul style="list-style-type: none"> <li>• 인증, 접근제어, 기밀성, 암호화 등의 기능을 의미한다.</li> <li>• 보안 약점에는 적절한 인증 없는 중요기능 허용, 부적절한 인가 등이 있다.</li> </ul>
시간 및 상태	<ul style="list-style-type: none"> <li>• 동시 수행을 지원하는 병렬 처리 시스템이나 다수의 프로세스가 동작하는 환경에서 시간과 실행 상태를 관리하여 시스템이 원활히 동작되도록 코딩하는 것을 의미한다.</li> <li>• 보안 약점에는 검사 시점과 사용 시점(TOCTOU) 경쟁조건, 종료되지 않는 반복문 또는 재귀함수 등이 있다.</li> </ul>
에러처리	<ul style="list-style-type: none"> <li>• 소프트웨어 실행 중 발생할 수 있는 오류들을 사전에 정의하여 에러로 인해 발생할 수 있는 문제들을 예방하는 것을 의미한다.</li> <li>• 보안 약점에는 오류 메시지를 통한 정보 노출, 오류 상황 대응 부재 등이 있다.</li> </ul>
코드 오류	<ul style="list-style-type: none"> <li>• 개발자들이 코딩 중 실수하기 쉬운 형(Type) 변환, 자원의 반환 등을 고려하며 코딩하는 것을 의미한다.</li> <li>• 보안 약점에는 널 포인터 역참조, 부적절한 자원 해제 등이 있다.</li> </ul>
캡슐화	<ul style="list-style-type: none"> <li>• 데이터(속성)와 데이터를 처리하는 함수를 하나의 객체로 묶어 코딩하는 것을 의미한다.</li> <li>• 보안 약점에는 잘못된 세션에 의한 데이터 정보 노출, 제거되지 않고 남은 디버그 코드 등이 있다.</li> </ul>

## API 응용

- API를 잘못 사용하거나 보안에 취약한 API를 사용하지 않도록 고려하여 코딩하는 것을 의미한다.
- 보안 약점에는 DNS lookup에 의존한 보안결정, 취약한 API 사용이 있다.

### 3 API(Application Programming Interface)

API는 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스를 의미한다.

- API는 프로그래밍 언어에서 특정한 작업을 수행하기 위해 사용되거나, 운영체제의 파일 제어, 화상 처리, 문자 제어 등의 기능을 활용하기 위해 사용된다.
- API는 개발에 필요한 여러 도구를 제공하기 때문에 이를 이용하면 원하는 기능을 쉽고 효율적으로 구현할 수 있다.
- API의 종류에는 Windows API, 단일 유닉스 규격(SUS), Java API, 웹 API 등이 있으며, 누구나 무료로 사용할 수 있게 공개된 API를 Open API라고 한다.



#### 기출문제 따라잡기

#### Section 123

출제예상

#### 1. 소프트웨어 개발 보안 점검 항목에 대한 설명으로 잘못된 것은?

- ① 세션 통제는 세션의 연결과 연결로 인해 발생하는 정보를 관리하는 것을 의미한다.
- ② 보안 기능은 입력 데이터에 대한 유효성 검증체계를 갖추고, 검증 실패 시 이를 처리할 수 있도록 코딩하는 것을 의미한다.
- ③ 에러처리는 소프트웨어 실행 중 발생할 수 있는 오류들을 사전에 정의하여 예방하는 것을 의미한다.
- ④ 캡슐화는 데이터와 함수를 하나의 객체로 묶어 코딩하는 것을 의미한다.

소프트웨어 개발 보안 점검 항목들의 특징을 구분할 수 있어야 합니다.

출제예상

#### 2. 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 함으로써 효율적인 소프트웨어 구현을 도와주는 인터페이스는?

- ① IDE(Integrated Development Environment)
- ② 통신 프로토콜(Communication Protocol)
- ③ API(Application Programming Interface)
- ④ USB(Universal Serial Bus)

응용 프로그램(Application Program)의 개발을 도와 주는 인터페이스(Interface)는 무엇일까요?

출제예상

#### 3. 다음 지문의 ( ) 안에 들어갈 용어로 알맞은 것은?

소프트웨어 개발 과정에서 발생할 수 있는 보안 취약점을 최소화하여 보안 위협으로부터 안전한 소프트웨어를 개발하기 위한 일련의 보안 활동은 데이터의 기밀성, 무결성, 가용성을 유지하는 것을 목표로 한다. 이를 위해 정부에서 제공하는 ( )를 참고하여 보안 항목들을 점검하여야 한다.

- ① 소프트웨어 개발 보안 가이드
- ② CERT 구축/운영 안내서
- ③ 보안인증제도 안내서
- ④ 암호기술 구현 안내서

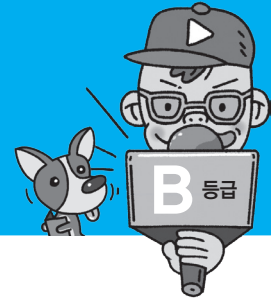
소프트웨어 개발 과정에서 발생할 수 있는 보안 위협을 방지하도록 도와주는 것은 무엇일까요?

▶ 정답 : 1. ② 2. ③ 3. ①



#### 전문가의 조언

API는 리모컨과 비교하면 이해가 쉽습니다. 리모컨으로 TV의 채널을 바꾸기 위해서는 채널 변경 버튼을 눌러야 하고, 소리를 조절하기 위해서는 음량 조절 버튼을 눌러야 하는 등 제조사가 미리 정해둔 방법을 이용해야 합니다. 여기서 TV를 조작하기 위해 제조사가 미리 정해둔 방법이 바로 API에 해당합니다.



## 전문의가의 조언

배치 프로그램이 무엇인지 개념을 명확히 하고 수행 주기 및 필수 요소들을 기억해 두세요. 그리고 나서 배치 프로그램을 구현하는 스케줄러의 종류와 각각의 기능적인 특징에는 어떤 것들이 있는지 알아두세요.

## 전문의가의 조언

배치 프로그램은 주기마다 수행되는 백업 작업, 외부의 데이터베이스로부터 최신 자료를 갱신하는 작업 등 대용량의 데이터가 주기적으로 교환되는 업무에 주로 사용됩니다. 게다가 배치 프로그램은 대용량의 데이터가 오가는 만큼 자원을 많이 차지하므로 업무에 방해되지 않도록 주로 야간이나 새벽에 수행되도록 설정합니다.

## 전문의가의 조언

배치 스케줄러 없이 배치 프로그램을 만들 수는 있으나, 코드를 직접 작성하는 것과 안정성 있는 외부 코드를 가져와 쓰는 것에는 큰 차이가 있습니다. 특히 배치 프로그램의 필수 요소들을 구현하기 위해서는 상당량의 코드를 작성해야 하는데, 이런 부분을 생략할 수 있게 해주니 큰 도움이 되죠.

### 스프링 프레임워크(Spring Framework)

스프링 프레임워크는 웹 서버 개발을 위한 다양한 API를 제공하는 오픈 소스의 경량형 애플리케이션 프레임워크로, 전자정부 표준 프레임워크의 기반 기술로도 사용되고 있습니다.

## 1 배치 프로그램(Batch Program)의 개요

배치 프로그램은 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하는 것을 의미한다.

- 배치 프로그램이 자동으로 수행되는 주기에 따라 정기 배치, 이벤트성 배치, On-Demand 배치로 구분된다.

정기 배치	일, 주, 월과 같이 정해진 기간에 정기적으로 수행된다.
이벤트성 배치	특정 조건을 설정해두고 조건이 충족될 때만 수행된다.
On-Demand 배치	사용자 요청 시 수행된다.

- 배치 프로그램이 갖추어야 하는 필수 요소는 다음과 같다.

대용량 데이터	대량의 데이터를 가져오거나, 전달하거나, 계산하는 등의 처리가 가능해야 한다.
자동화	심각한 오류가 발생하는 상황을 제외하고는 사용자의 개입 없이 수행되어야 한다.
견고성	잘못된 데이터나 데이터 중복 등의 상황으로 중단되는 일 없이 수행되어야 한다.
안정성/신뢰성	오류가 발생하면 오류의 발생 위치, 시간 등을 추적할 수 있어야 한다.
성능	다른 응용 프로그램의 수행을 방해하지 않아야 하고, 지정된 시간 내에 처리가 완료되어야 한다.

## 2 배치 스케줄러(Batch Scheduler)

배치 스케줄러는 일괄 처리(Batch Processing) 작업이 설정된 주기에 맞춰 자동으로 수행되도록 지원해주는 도구이다.

- 배치 스케줄러는 특정 업무(Job)를 원하는 시간에 처리할 수 있도록 지원한다는 특성 때문에 잡 스케줄러(Job Scheduler)라고도 불린다.
- 주로 사용되는 배치 스케줄러에는 스프링 배치, Quartz 등이 있다.

### 스프링 배치(Spring Batch)

- Spring Source사와 Accenture사가 2007년 공동 개발한 오픈 소스 프레임워크이다.
- 스프링 프레임워크\*의 특성을 그대로 가져와 스프링이 가지고 있는 다양한 기능들을 모두 사용할 수 있다.
- 데이터베이스나 파일의 데이터를 교환하는데 필요한 컴포넌트들을 제공한다.

- 로그 관리, 추적, 트랜잭션 관리, 작업 처리 통계, 작업 재시작 등의 다양한 기능을 제공한다.
- 스프링 배치의 주요 구성 요소와 역할

Job	수행할 작업 정의
Job Launcher	실행을 위한 인터페이스
Step	Job 처리를 위한 제어 정보
Job Repository	Step의 제어 정보를 포함하여 작업 실행을 위한 모든 정보 저장

### Quartz

- 스프링 프레임워크로 개발되는 응용 프로그램들의 일괄 처리를 위한 다양한 기능을 제공하는 오픈 소스 라이브러리이다.
- 수행할 작업과 수행 시간을 관리하는 요소들을 분리하여 일괄 처리 작업에 유연성을 제공한다.
- Quartz의 주요 구성 요소와 역할

Scheduler	실행 환경 관리
Job	수행할 작업 정의
JobDetail	Job의 상세 정보
Trigger	Job의 실행 스케줄 정의



### 기출문제 따라잡기

### Section 124

출제예상

**1. 배치 프로그램이 가지는 필수 요소에 대한 설명으로 잘못된 것은?**

- ① 필수 요소에는 대용량 데이터, 자동화, 견고성, 안정성, 성능이 있다.
- ② 심각한 오류가 아니라면 사용자의 개입 없이 수행되어야 한다.
- ③ 잘못된 데이터나 중복이 발생하면 일시 정지한 후 관리자의 조치를 기다려야 한다.
- ④ 시스템 내의 다른 프로그램의 기능 수행을 방해하지 않아야 한다.

사소한 오류로 대량의 작업이 멈춰버린다면 곤란하지 않을까요? 배치 프로그램은 관리자가 항상 모니터링 할 수 없다는 것을 고려하여 오류 발생 시 이를 처리하는 방법을 정의할 수 있도록 만들어져 있습니다.

출제예상

**2. 배치 프로그램의 수행 주기에 속하지 않는 것은?**

- ① 정기 배치
- ② 상시 배치
- ③ On-Demand 배치
- ④ 이벤트성 배치

배치 프로그램의 수행 주기 3가지 '정기, 이벤트성, On-Demand'를 기억해 두세요.

▶ 정답 : 1. ③ 2. ②



## 전문의가의 조언

패키지 소프트웨어는 어려운 내용이 아니니 가벼운 마음으로 정리하세요.

**전용 개발 소프트웨어**  
전용 개발 소프트웨어는 패키지 소프트웨어에 대응되는 용어로, 사업 환경에 맞춰 직접 개발한 소프트웨어를 의미합니다.

## 전문의가의 조언

패키지 소프트웨어를 사용하면 개발 조직은 필요하지 않지만 시스템에 대한 이해를 갖고 커스터마이징 및 관리를 담당할 인력은 필요합니다.

## 1 패키지 소프트웨어(Package Software)의 개요

패키지 소프트웨어는 기업에서 일반적으로 사용하는 여러 기능들을 통합하여 제공하는 소프트웨어를 의미한다.

- 기업에서는 패키지 소프트웨어를 구입하여 기업 환경에 적합하게 커스터마이징(Customizing)하여 사용한다.
- 패키지 소프트웨어를 이용하여 시스템을 구축하는 방식을 패키지 개발 방식이라고 한다.
- 기능 요구사항을 70% 이상 충족시키는 패키지 소프트웨어가 있을 때만 사용하는 것이 적합하다.

## 2 패키지 소프트웨어의 특징

패키지 소프트웨어는 요구사항을 분석하여 업무 특성에 맞게 전용으로 개발되는 소프트웨어\*와 비교하여 안정성, 라이선스, 생산성 등에서 차이가 있다.

- 패키지 소프트웨어는 전문 업체에 의해 품질이 검증되었고, 국제/산업계 표준을 준수하고 있어 안정적인 이용이 가능하다.
- 소프트웨어에 대한 라이선스가 판매자에게 있기 때문에 시스템 구축 후 기능 추가 및 코드 재사용 등에 제약이 발생한다.
- 개발 조직을 갖추어야 할 필요성이 없어 비용을 절감할 수 있고, 이미 개발된 소프트웨어를 사용하기 때문에 프로젝트 기간이 단축된다.
- 기존에 보유하고 있던 시스템과의 상호 연동 및 연계가 어려울 수 있다.
- 결함이 발생한 경우 판매처의 프로세스에 따라 보완되므로 이용자의 사정에 따라 능동적인 대처를 기대하기는 어렵다.



### 잠깐만요 패키지 소프트웨어와 전용 개발 소프트웨어의 비교

	패키지 소프트웨어	전용 개발 소프트웨어
기능 요구사항	70% 이상 충족시키는 패키지 소프트웨어가 있는 경우 이용	모든 기능 요구사항 반영 가능
안정성	품질이 검증되었고, 업계 표준 준용	개발자의 역량에 따라 달라짐
라이선스	판매자	회사

생산성	개발을 위한 인력과 시간이 절약됨	개발을 위한 인력과 시간이 필요함
호환성	보장이 안 됨	설계 단계부터 고려하여 개발
유지보수	결함 발생 시 즉시 대응이 어려움	결함 발생 시 즉시 대응이 가능



## 기출문제 따라잡기

Section 125

출제예상

### 1. 패키지 소프트웨어에 대한 설명으로 잘못된 것은?

- ① 기업에서 일반적으로 사용하는 기능들을 통합하여 제공하는 소프트웨어이다.
- ② 기능 요구사항을 70% 이상 충족시켰을 때 사용하는 것이 좋다.
- ③ 기업 환경에 적합하게 커스터마이징(Customizing)하여 사용한다.
- ④ 패키지 개발 방식은 개발 인력을 갖춰 직접 개발하는 방식을 말한다.

패키지 개발 방식은 실제로 개발하는 것이 아닌 외부의 상용 소프트웨어 패키지를 구매하여 사용하는 방식입니다.

출제예상

### 2. 다음 중 패키지 개발 방식에 대한 설명으로 잘못된 것은?

- ① 패키지 소프트웨어를 이용하여 시스템을 구축하는 개발 방식으로 전용 개발 방식과 대응된다.
- ② 직접 개발하는 전용 개발 방식에 비해 안정성, 품질, 호환성 등이 뛰어나다.
- ③ 라이선스에 대한 권리가 판매자에게 있기 때문에 프로그램을 분해하거나 모듈을 재사용하는데 제한이 있다.
- ④ 이미 개발된 소프트웨어를 이용하기 때문에 프로젝트 기간이 단축된다.

호환성이 뛰어나다는 말은 기존의 시스템과 연동에 문제가 없다는 말입니다. 외부에서 제작된 패키지 소프트웨어가 애초부터 연동을 고려하여 제작된 전용 개발 소프트웨어에 비해 호환성이 떨어질 수 있을까요?

출제예상

### 3. 패키지 소프트웨어와 전용 개발 소프트웨어의 특성에 대한 설명으로 잘못된 것은?

- ① 패키지 소프트웨어를 사용하는 개발 방식은 기능 요구사항을 70% 이상 충족한 경우 사용하는 것이 좋다.
- ② 패키지 소프트웨어는 전문적인 개발사에 의해 안정성이 보장되지만 전용 개발 소프트웨어는 개발자의 역량이 부족한 경우 안정성이 보장되지 않는다.
- ③ 패키지 소프트웨어는 전용 개발 소프트웨어와 달리 소프트웨어 개발·관리를 위한 어떠한 인력도 필요로 하지 않는다.
- ④ 복잡한 사업 환경에서는 기존의 시스템과 요구사항들을 고려하여 맞춤 제작되는 전용 개발 소프트웨어가 적합하다.

패키지 소프트웨어는 직접 개발하지는 않지만 소프트웨어를 사용하는 입장에서 유지보수를 포함한 여러 관리 작업이 필요하기 때문에 관리 인력은 있어야 합니다.

▶ 정답 : 1. ④ 2. ② 3. ③



### 1. 소프트웨어 개발을 위한 개발 환경 구축에 대한 설명으로 옳지 않은 것은?

- ① 소프트웨어가 운영될 환경과 동일한 구조로 구축한다.
- ② 성능, 편의성, 경제성 등 다양한 비즈니스적 요소를 고려하여 구축한다.
- ③ 요구사항을 구체적으로 분석하여 필요한 하드웨어와 소프트웨어 환경을 구축한다.
- ④ 소프트웨어 환경은 개발 프로그램을 의미하며 요구사항 관리 도구부터 테스트 도구까지 다양한 도구로 구성된다.

### 2. 웹 서버에 대한 설명으로 잘못된 것은?

- ① 브라우저로 요청과 응답을 처리하는 프로토콜인 HTTP/HTTPS를 지원한다.
- ② HTML, CSS 등의 정적 파일들을 저장하고 관리한다.
- ③ 네트워크 트래픽의 포화를 방지하는 가상 호스팅 기능을 지원한다.
- ④ 사용자가 적합한 권한을 갖고 접속하는지 확인하는 인증 기능을 지원한다.

### 3. 소프트웨어 환경 중 개발 소프트웨어에 대한 설명으로 옳지 않은 것은?

- ① 요구사항 수집과 분석 등을 도와주는 소프트웨어가 포함된다.
- ② 설계 모델링 도구에는 PlantUML, ArgoUML 등이 있다.
- ③ 구현 도구는 개발 언어를 말하며, 대표적인 언어로 C, Java, Python 등이 있다.
- ④ 산출물들을 버전별로 관리해주는 버전 관리 도구가 포함되어 있다.

### 4. 개발 언어 선정 시 고려해야 하는 기준들에 대한 설명으로 잘못된 것은?

- ① 개발하려는 소프트웨어의 목적에 맞게 선정해야 한다.
- ② 오래되고 많이 사용되어지고 있는 개발 언어 보다는 최신 언어를 활용하는 것이 좋다.
- ③ 코드의 작성 및 구현이 효율적이어야 한다.
- ④ 언어에 대한 개발자들의 이해도가 높아야 한다.

### 5. 개발 언어 선정 시 고려해야 하는 기준에 속하지 않는 특성은?

- ① 안정성                      ② 효율성
- ③ 친밀성                      ④ 적정성

### 6. 서버 개발에 대한 설명 중 옳지 않은 것은?

- ① 서버 개발은 서버 프로그램을 제작하여 서버를 구축하는 것을 말한다.
- ② 서버 개발로 구축되는 서버는 웹 서버이다.
- ③ 서버 개발에는 JavaScript, PHP 등의 언어가 사용된다.
- ④ 서버 개발을 위해 다양한 프레임워크들이 존재하며 프레임워크들은 언어에 종속적이다.

### 7. 다음에서 설명하는 서버 개발 프레임워크는?

- Java 플랫폼을 위한 오픈 소스의 경량형 애플리케이션 프레임워크이다.
- 동적인 웹 사이트의 개발을 위해 다양한 서비스를 제공한다.
- 전자정부 표준 프레임워크의 기반 기술로 사용되고 있다.

- ① Spring
- ② Node.js
- ③ Django
- ④ .Net

### 8. 서버 프로그램 구현 방법에 대한 설명으로 잘못된 것은?

- ① 응용 소프트웨어와 동일한 방식으로 구현된다.
- ② 각 모듈들은 독립성을 보장할 수 있도록 구현되어야 한다.
- ③ 모듈의 결합도와 응집도를 강하게 하여 과도한 상호작용을 배제해야 한다.
- ④ 재사용이 가능한 공통 모듈을 구분하는 것이 생산성 향상에 도움이 된다.

### 9. 프레임워크의 특성에 해당하지 않는 것은?

- ① 재사용이 가능한 모듈의 제공으로 생산성을 향상시킨다.
- ② 캡슐화를 통해 소프트웨어의 품질을 향상시킨다.
- ③ 다형성을 활용하여 다양한 소프트웨어를 개발하는 것이 가능하다.
- ④ 모든 객체의 제어를 개발자가 직접 관리함으로써 문제 발생 시 즉각 대응이 가능하다.





**10.** 프레임워크의 특성 중 다형성(Polymorphism)을 통한 인터페이스 확장이 가능하여 다양한 형태와 기능을 가진 애플리케이션의 개발이 가능함을 의미하는 것은?

- ① 모듈화(Modularity)
- ② 제어의 역흐름(Inversion of control)
- ③ 확장성(Extensibility)
- ④ 재사용성(Reusability)

**11.** 소프트웨어 개발 보안 점검 항목에 해당하지 않는 것은?

- ① 세션 통제                      ② 암호화
- ③ 코드 오류                      ④ 보안 기능

**12.** 동시 수행이나 다수의 프로세스가 동작하는 환경에서 시스템이 원활히 동작되는지 확인하는 소프트웨어 개발 보안 점검 항목은?

- ① 시간 및 상태
- ② 입력 데이터 검증 및 표현
- ③ 에러처리
- ④ 캡슐화

**13.** API에 대한 설명으로 옳지 않은 것은?

- ① 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스다.
- ② 개발에 필요한 여러 기능들을 활용할 수 있도록 도와준다.
- ③ 응용 프로그램 개발의 생산성은 향상시켜주지만 개발 난이도는 증가한다.
- ④ Windows에서 창이나 파일에 접근하기 위해서는 Windows API를 사용해야 한다.

**14.** API(Application Programming Interface) 중 누구나 무료로 사용할 수 있도록 공개된 API를 무엇이라 하는가?

- ① Free API                      ② Java API
- ③ SUS                          ④ Open API

**15.** 다음 중 사용자와 상호 작용 없이 여러 작업들이 미리 정해진 일련의 순서에 따라 일괄적으로 처리되는 것을 의미하는 용어는?

- ① Batch Program
- ② API
- ③ Application Program
- ④ ORM

**16.** 배치 스케줄러에 대한 설명으로 잘못된 것은?

- ① 일괄 처리 작업을 지원하는 프레임워크 또는 라이브러리 등을 의미한다.
- ② 특정 업무를 특정 시간에 배치 및 처리하기 때문에 Job 스케줄러라고도 불린다.
- ③ 주요 배치 스케줄러에는 스프링 배치(Spring Batch)와 퀴즈(Quartz)가 있다.
- ④ 스프링 배치는 Scheduler, Job, Trigger 등의 구성 요소를 갖고 있다.

**17.** 배치 스케줄러의 하나인 Quartz에 대한 설명으로 옳지 않은 것은?

- ① 다양한 작업 조건으로 배치 작업 수행이 가능하나 오류 추적이 어렵다.
- ② 스프링 프레임워크로 개발되는 응용 프로그램을 지원한다.
- ③ 무료로 다양한 기능을 제공하는 오픈 소스 라이브러리이다.
- ④ Scheduler, Job, JobDetail 등의 요소로 구성되어 있다.

**18.** 다음에서 설명하는 용어는?

- 기업 환경에 적합하도록 커스터마이징 하여 사용하는 외부의 상용 소프트웨어이다.
- 기업에 필요한 여러 기능들을 통합하여 제공한다.
- 기능 요구사항을 70% 이상 만족시키는 경우에 사용하는 것이 좋다.

- ① 전용 개발 소프트웨어
- ② 패키지 소프트웨어
- ③ 라이브러리
- ④ 통합 개발 환경 도구

**19.** 다음 중 패키지 소프트웨어와 전용 개발 소프트웨어에 대한 설명으로 가장 옳지 않은 것은?

- ① 패키지 소프트웨어는 전문 업체에 의해 품질이 검증되어 있다.
- ② 라이선스를 누가 갖고 있는지에 따라 구분할 수 있다.
- ③ 전용 개발 소프트웨어는 설계 단계부터 고려되어 호환성이 보장된다.
- ④ 패키지 소프트웨어는 결함 발생 시 제공사의 즉각적인 대응으로 빠른 대처가 가능하다.

**1. Section 121**

소프트웨어 환경은 시스템 소프트웨어와 개발 소프트웨어로 구성된다.

**2. Section 121**

- 가상 호스팅(Virtual Hosting)은 하나의 서버로 여러 개의 도메인 이름을 연결하는 기능이다.
- 네트워크 트래픽의 포화를 방지하기 위해 응답 속도를 제한하는 기능은 대역폭 제한(Bandwidth Throttling)이다.

**3. Section 121**

구현 도구는 언어를 통해 개발을 할 수 있도록 지원하는 통합 개발 환경 도구 등을 의미하며, 종류에는 Eclipse, IntelliJ IDEA, Netbeans 등이 있다.

**4. Section 121**

②번의 내용은 개발 언어의 선정 기준인 범용성에 위배되는 내용이다. 최신 언어는 다른 개발 사례가 적어 문제 발생 시 대응하는데 많은 시행착오를 겪을 수 있다.

**5. Section 121**

개발 언어의 선정 기준에는 적정성, 효율성, 이식성, 친밀성, 범용성 등이 있다.

**6. Section 122**

서버 개발로 구축되는 서버는 웹 애플리케이션 서버(WAS)이다.

**8. Section 122**

과도한 상호작용을 배제한다는 것은 기능적 독립성을 갖는다는 의미이며, 기능적 독립성을 갖기 위해서는 결합도는 약하게, 응집도는 강하게 해야 한다.

**9. Section 122**

④번의 내용은 프레임워크의 특성 중 제어의 역흐름(Inversion of Control)에 위배되는 내용이다.

- ※ 제어의 역흐름(Inversion of Control) : 프레임워크가 개발자가 통제·관리해야 하는 객체들의 제어를 수행하여 생산성을 향상시켜주는 것

**10. Section 122**

- 모듈화(Modularity) : 캡슐화를 통해 모듈화를 강화하고 설계 및 구현의 변경에 따른 영향을 최소화함
- 재사용성(Reusability) : 재사용 가능한 모듈을 제공하여 생산성을 향상시킴

**11. Section 123****소프트웨어 개발 보안 점검 항목**

세션 통제, 입력 데이터 검증 및 표현, 보안 기능, 시간 및 상태, 에러처리, 코드 오류, 캡슐화, API 오용

**12. Section 123**

- 입력 데이터 검증 및 표현 : 입력 데이터에 대한 유효성 검증체계를 갖추고, 검증 실패 시 이를 처리할 수 있도록 코딩하는 것
- 에러처리 : 소프트웨어 실행 중 발생할 수 있는 오류들을 사전에 정의하여 에러로 인해 발생할 수 있는 문제들을 예방하는 것
- 캡슐화 : 데이터(속성)와 데이터를 처리하는 함수를 하나의 객체로 묶어 코딩하는 것

**13. Section 123**

API는 OS나 라이브러리 등에 있는 기능을 사용할 수 있게 해주는 기능으로, 직접 기능을 구현하는 것에 비해 생산성 향상은 물론 개발 난이도도 하락한다.

**15. Section 124**

- API(Application Programming Interface) : 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스
- 응용 프로그램(Application Program) : 특정 업무를 처리하기 위해 만들어진 프로그램
- ORM(Object-Relational Mapping) : 객체지향 프로그래밍의 객체(Object)와 관계형 데이터베이스(Relational Database)의 데이터를 연결(Mapping)하는 기술

**16. Section 124**

스프링 배치의 주요 구성 요소에는 Job, JobLauncher, Step, JobRepository가 있다.

**17. Section 124**

Quartz는 오류가 발생하면 오류의 발생 위치, 시간 등을 추적할 수 있는 안정성, 신뢰성을 필수적으로 갖추고 있다.

**18. Section 125**

- 전용 개발 소프트웨어 : 사업 환경에 맞추어 직접 개발한 소프트웨어
- 라이브러리 : 개발 편의를 위해 자주 사용되는 코드, 클래스, 값, 자료형 등의 다양한 자원들을 모아놓은 것
- 통합 개발 환경 도구 : 개발에 필요한 편집기, 컴파일러,



디버거 등의 다양한 툴을 하나의 인터페이스로 제공하는 소프트웨어

### 19. Section 125

패키지 소프트웨어는 결함이 발생했을 때 판매처의 프로세스에 따라 보완되므로, 이용자의 사정에 따라 즉각적이고 능동적인 대처가 어렵다.



m·e·m·o

A series of horizontal dotted lines for writing, spanning the width of the page.



# 2 장

## 프로그래밍 언어 활용

126 데이터 타입 **B** 등급

127 변수 **A** 등급

128 연산자 **A** 등급

129 제어문 **A** 등급

130 반복문 **A** 등급

131 배열과 문자열 **A** 등급

132 포인터 **A** 등급

133 절차적 프로그래밍 언어 **C** 등급

134 객체지향 프로그래밍 언어 **C** 등급

135 스크립트 언어 **C** 등급

136 선언형 언어 **C** 등급

137 라이브러리 **C** 등급

138 데이터 입 · 출력 **A** 등급

139 예외 처리 **C** 등급

140 프로토타입 **C** 등급



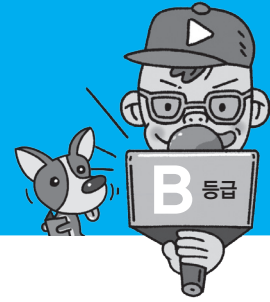
이 장에서 꼭 알아야 할 키워드 **Best 10**

1. 변수 2. 연산자 3. if 4. for 5. while 6. do~while 7. 배열 8. 포인터 9. scanf 10. printf

# SECTION 126

● LIVE

## 데이터 타입



### 전문가의 조언

프로그래밍 언어 활용에서 학습하는 내용은 C언어를 기반으로 수록하였습니다. JAVA 프로그램은 C, C++ 언어를 베이스로 했기 때문에 C언어와 문법이 거의 같습니다. 이번 장에서는 정보처리 기사 필기 시험에 출제될 정도의 범위 내에서 C언어 프로그램과 다른 점만 별도로 설명하였습니다.

### 전문가의 조언

데이터 타입은 변수가 가질 수 있는 값의 길이, 성질을 의미합니다. 데이터 타입의 유형을 구분할 수 있도록 정리하세요.

#### 변수(Variable)

컴퓨터가 명령을 처리하는 도중 발생하는 값을 저장하기 위한 공간으로, 변할 수 있는 값을 의미합니다. 자세한 내용은 Section 127을 참조하세요.

### 전문가의 조언

데이터 타입의 크기는 사용하는 컴퓨터나 운영체제에 따라 조금씩 다릅니다. 당연히 기억 범위도 다르겠죠. 기억 범위를 외우려고 노력하지 마세요. 대략 저 정도 크기의 데이터가 들어가는구나! 정도만 알아두세요.

예 Windows(64비트)에서 long은 4바이트, long double은 8바이트지만 Linux(64비트)에서는 long은 8바이트, long double은 16바이트입니다.

## 1 데이터 타입

데이터 타입(Data Type)은 변수(Variable)\*에 저장될 데이터의 형식을 나타내는 것으로, 변수에 값을 저장하기 전에 문자형, 정수형, 실수형 등 어떤 형식의 값을 저장할지 데이터 타입을 지정하여 변수를 선언해야 한다.

### • 데이터 타입의 유형

유형	기능	예
정수 타입(Integer Type)	정수, 즉 소수점이 없는 숫자를 저장할 때 사용한다.	1, -1, 10, -100
부동 소수점 타입(Floating Point Type)	소수점 이하가 있는 실수를 저장할 때 사용한다.	$0.123 \times 10^2$ , $-1.6 \times 2^3$
문자 타입(Character Type)	<ul style="list-style-type: none"> <li>한 문자를 저장할 때 사용한다.</li> <li>작은따옴표(" ") 안에 표시한다.</li> </ul>	'A', 'a', '!', '*'
문자열 타입(Character String Type)	<ul style="list-style-type: none"> <li>문자열을 저장할 때 사용한다.</li> <li>큰따옴표(" ") 안에 표시한다.</li> </ul>	"Hello!", "1+2=3"
불린 타입(Boolean Type)	<ul style="list-style-type: none"> <li>조건의 참(True), 거짓(False) 여부를 판단하여 저장할 때 사용한다.</li> <li>기본값은 거짓(False)이다.</li> </ul>	true, false
배열 타입(Array Type)	<ul style="list-style-type: none"> <li>같은 타입의 데이터 집합을 만들어 저장할 때 사용한다.</li> <li>데이터는 중괄호({ }) 안에 콤마(,)로 구분하여 값을 나열한다.</li> </ul>	{1, 2, 3, 4, 5}

## 2 C/C++의 데이터 타입 크기 및 기억 범위

종류	데이터 타입	크기	기억 범위
문자	char	1Byte	-128 ~ 127
부호없는 문자형	unsigned char	1Byte	0 ~ 255
정수	short	2Byte	-32,768 ~ 32,767
	int	4Byte	-2,147,483,648 ~ 2,147,483,647
	long	4Byte	-2,147,483,648 ~ 2,147,483,647
	long long	8Byte	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

부호없는 정수형	unsigned short	2Byte	0 ~ 65,535
	unsigned int	4Byte	0 ~ 4,294,967,295
	unsigned long	4Byte	0 ~ 4,294,967,295
실수	float	4Byte	$1.2 \times 10^{-38} \sim 3.4 \times 10^{38}$
	double	8Byte	$2.2 \times 10^{-308} \sim 1.8 \times 10^{308}$
	long double	8Byte	$2.2 \times 10^{-308} \sim 1.8 \times 10^{308}$

### 3 JAVA의 데이터 타입 크기 및 기억 범위

종류	데이터 타입	크기	기억 범위
문자	char	2Byte	0 ~ 65,535
정수	byte	1Byte	-128 ~ 127
	short	2Byte	-32,768 ~ 32,767
	int	4Byte	-2,147,483,648 ~ 2,147,438,647
	long	8Byte	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
실수	float	4Byte	$1.4 \times 10^{-45} \sim 3.4 \times 10^{38}$
	double	8Byte	$4.9 \times 10^{-324} \sim 1.8 \times 10^{308}$
논리	boolean	1Byte	true 또는 false

#### 따라잡기



#### 기출문제 따라잡기

Section 126

이전기출

1. C 언어에서 정수형 변수를 선언할 때 사용하는 것은?

- ① char
- ② int
- ③ float
- ④ double

'정수'를 영어로 'integer'라고 합니다.

이전기출

2. 다음 중 C언어에서 사용하는 기본적인 데이터 형식이 아닌 것은?

- ① long
- ② integer
- ③ double
- ④ float

C언어에서 정수형 자료를 선언할 때 사용하는 예약어는 'integer'가 아니라 'int'입니다.

▶ 정답 : 1. ② 2. ②



## 기출문제 따라잡기

Section 126

이전기출

3. 다음 중 C언어에서 문자형 자료를 나타내는 데이터 형식은?

- ① void                                      ② char
- ③ float                                      ④ int

"문자"를 영어로 "Character"라고 합니다.

이전기출

4. 다음 중 C언어의 데이터 형식에 해당하지 않는 것은?

- ① double                                      ② long
- ③ char                                        ④ signed

C언어에서 signed는 데이터 형식이 아닙니다. 부호 없는 정수 변수를 선언할 때 사용하는 예약어는 unsigned입니다.

이전기출

5. 다음의 C언어 데이터 유형(Data Type) 가운데 메모리를 가장 많이 차지하는 것은?

- ① char                                        ② int
- ③ long                                        ④ double

char은 1Byte, int는 4Byte, long은 4Byte, double은 8Byte입니다.

출제예상

6. 다음 중 조건이 참인지 거짓인지 판단하고자 할 때 사용하는 데이터 타입은?

- ① Integer Type
- ② Boolean Type
- ③ Character String Type
- ④ Floating Point Type

불린(Boolean)은 참과 거짓을 나타내는 숫자 1과 0만을 이용하는 방식을 의미합니다.

출제예상

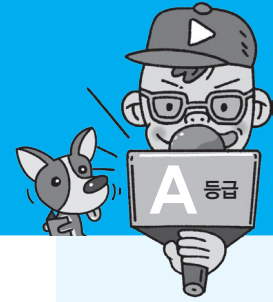
7. 다음 중 데이터 타입에 대한 설명으로 가장 옳지 않은 것은?

- ① 문자 타입은 문자 하나를 저장할 때 사용하며, 큰따옴표 안에 입력한다.
- ② 문자열 타입은 문자열을 저장할 때 사용하며, 큰따옴표 안에 입력한다.
- ③ 배열 타입은 여러 데이터를 하나로 묶어서 저장하고자 할 때 사용하며, 사용할 값을 중괄호 안에 입력한다.
- ④ 부동 소수점 타입은 실수값을 저장할 때 사용한다.

문자는 작은따옴표, 문자열은 큰따옴표로 묶어서 입력한다는 것을 기억해 두세요.

▶ 정답 : 3. ② 4. ④ 5. ④ 6. ② 7. ①





## 1 변수의 개요

변수(Variable)는 컴퓨터가 명령을 처리하는 도중 발생하는 값을 저장하기 위한 공간으로, 변할 수 있는 값을 의미한다.

- 변수는 저장하는 값에 따라 정수형, 실수형, 문자형, 포인터형 등으로 구분한다.

## 2 변수명 작성 규칙

- 영문자, 숫자, \_(under bar)를 사용할 수 있다.
- 첫 글자는 영문자나 \_(under bar)로 시작해야 하며, 숫자는 올 수 없다.
- 글자 수에 제한이 없다.
- 공백이나 \*, +, -, / 등의 특수문자를 사용할 수 없다.
- 대 · 소문자를 구분한다.
- 예약어를 변수명으로 사용할 수 없다.
- 변수 선언 시 문장 끝에 반드시 세미콜론(;)을 붙여야 한다.

잠깐만요



### 예약어

예약어는 정해진 기능을 수행하도록 이미 용도가 정해져 있는 단어로, 변수 이름이나 다른 목적으로 사용할 수 없습니다.

- C언어에는 다음과 같은 예약어가 있습니다.

구분		예약어
제어문	반복	do, for, while
	선택	case, default, else, if, switch
	분기	break, continue, goto, return
자료형		char, double, enum, float, int, long, short, signed, struct, typedef, union, unsigned, void
기억 클래스		auto, extern, register, static
기타		const, sizeof, volatile



### 전문가의 조언

수학에서 변수(變數)는 임의의 값을 대입할 수 있는 문자로, 값이 언 제라도 변할 수 있기 때문에 변수라고 합니다. 컴퓨터 언어에서는 수뿐만 아니라 문자, 문자열 등을 저장할 수 있는 공간의 이름을 의미합니다. 그 공간에 다른 값을 저장하면 값이 바뀌잖아요. 변수의 작성 규칙은 확실히 숙지하고 넘어가세요.



### 전문가의 조언

변수명 작성 규칙은 언어에 따라 다를 수 있으며, 여기서는 C언어를 기준으로 설명하였습니다.

#### 상수(Constant)

1, 2, 'a', "Hello"와 같이 프로그램이 시작되어 값이 한 번 결정되면 프로그램이 종료될 때까지 변경되지 않는 정보를 의미합니다.

auto는 기본값으로 생략이 가능합니다.

**예제** 다음의 변수명을 C언어의 변수명으로 사용할 수 있는지 여부를 쓰시오.

변수명	설명
2abc	변수명의 첫 글자를 숫자로 시작하여 사용할 수 없다.
sum*	특수문자 '*'를 변수명에 사용할 수 없다.
for	예약어를 변수명으로 사용할 수 없다.
ha p	변수명 중간에 공백을 사용할 수 없다.
Kim, kim	C언어는 대소문자를 구분하기 때문에 Kim과 kim은 서로 다른 변수로 사용할 수 있다.

#### 잠깐만요



#### 변수를 상수로 만들어 사용하기

- 변수는 프로그램을 실행하는 도중 발생한 값을 저장하기 위한 공간으로, 변수의 값은 변경될 수 있습니다. 하지만 변수에 저장된 값을 프로그램이 종료될 때까지 변경되지 않도록 상수로 만들어 사용할 수 있는데, 이런 경우 C언어에서는 const라는 예약어를 사용합니다.
- 변수처럼 상수에 이름을 붙여 기호화하여 사용한다고 하여 심볼릭(Symbolic) 상수라고도 합니다.

**예** const float PI = 3.1415927;

- const : 변수를 상수로 변경하는 예약어입니다. const를 자료형 뒤에 붙여 'int const a = 5;'와 같이 사용할 수도 있습니다.
  - float PI : 실수형으로 변수 PI를 선언하지만 const 예약어로 인해 PI는 상수가 됩니다.
  - 3.1415927 : 저장되는 값 그 자체로, 리터럴(Literal)이라고 합니다.
- ∴ 이렇게 선언되면 PI는 변수가 아닌 상수이므로, 이후 PI는 프로그램 안에서 3.14159270이란 값으로 고정되어 사용됩니다.

## 3 기억 클래스

변수 선언 시 메모리 내에 변수의 값을 저장하기 위한 기억영역이 할당되는데, 할당되는 기억영역에 따라 사용 범위에 제한이 있다. 이러한 기억영역을 결정하는 작업을 기억 클래스(Storage Class)라 한다.

- C언어에서는 다음과 같이 5가지 종류의 기억 클래스를 제공한다.

종류	기억영역	예약어	생존기간	사용 범위
자동 변수	메모리(스택)	auto*	일시적	지역적
레지스터 변수	레지스터	register		
정적 변수(내부)	메모리(데이터)	static	영구적	전역적
정적 변수(외부)		extern		
외부 변수				

### 자동 변수(Automatic Variable)

자동 변수는 함수나 코드의 범위를 한정하는 블록 내에서 선언되는 변수이다.

- 함수나 블록이 실행되는 동안에만 존재하며 이를 벗어나면 자동으로 소멸된다.
- 초기화하지 않으면 쓰레기값(Garbage Value)\*이 저장된다.

### 외부 변수(External Variable)

외부 변수는 현재 파일이나 다른 파일에서 선언된 변수나 함수를 참조(reference)하기 위한 변수이다.

- 외부 변수는 함수 밖에서 선언한다.
- 함수가 종료된 뒤에도 값이 소멸되지 않는다.
- 초기화하지 않으면 자동으로 0으로 초기화\* 된다.
- 다른 파일에서 선언된 변수를 참조할 경우 초기화 할 수 없다.

### 정적 변수(Static Variable)

정적 변수는 함수나 블록 내에서 선언하는 내부 정적 변수와 함수 외부에서 선언하는 외부 정적 변수가 있다.

- 내부 정적 변수는 선언한 함수나 블록 내에서만 사용할 수 있고, 외부 정적 변수는 모든 함수에서 사용할 수 있다.
- 두 변수 모두 함수나 블록이 종료된 뒤에도 값이 소멸되지 않는다.
- 초기화는 변수 선언 시 한 번만 할 수 있으며, 초기화를 생략하면 자동으로 0으로 초기화 된다.

### 레지스터 변수(Register Variable)

레지스터 변수는 메모리가 아닌 CPU 내부의 레지스터에 기억영역을 할당받는 변수이다.

- 자주 사용되는 변수를 레지스터에 저장하여 처리 속도\*를 높이기 위해 사용한다.
- 함수나 블록이 실행되는 동안에만 존재하며 이를 벗어나면 자동으로 소멸된다.
- 레지스터의 사용 개수는 한정되어 있어 데이터를 저장할 레지스터가 없는 경우 자동 변수로 취급되어 메모리에 할당된다.
- CPU에 저장되어 메모리 주소를 가질 수 없기 때문에 변수의 주소를 구하는 주소 연산자(&)를 사용할 수 없다.

## 4 변수의 선언

변수는 일반적으로 다음과 같은 형식으로 선언한다.

자료형 변수명 = 값;

- **자료형** : 변수에 저장될 자료의 형식을 지정한다.
- **변수명** : 사용자가 원하는 이름을 임의로 지정한다. 단 변수명 작성 규칙에 맞게 지정해야 한다.
- **값** : 변수를 선언하면서 초기화할 값을 지정한다. 단 값은 지정하지 않아도 된다.

#### 쓰레기값(Garbage Value)

메모리의 데이터를 변경하기 전에 마지막으로 남아 있던 데이터를 의미합니다.

#### 초기화

변수를 선언할 때 또는 선언된 변수에 처음 값을 저장하는 것을 초기화라 합니다. 초기화 이후에 저장된 값을 변경할 때는 그냥 대입 또는 저장한다고 합니다.

#### 레지스터 변수의 처리 속도

일반적인 변수는 메모리(주 기억장치)에 저장된 데이터를 레지스터(CPU)로 가져와서 연산한 후 그 결과를 다시 메모리에 저장하지만 레지스터 변수는 레지스터에 데이터를 저장해 두고 연산을 수행하므로 처리 속도가 빠릅니다.

#### 단정도 / 배정도

float 자료형은 '단정도형', double과 long double 자료형은 '배정도형'이라고 표현합니다.

#### 1.23e-2

1.23e-20에서 e는 10의 자승을 의미하므로  $1.23 \times 10^{-2}$ , 즉 0.0123을 의미합니다.

#### 실수 자료형에 따른 실수형 상수 입력 방법

실수형 상수는 기본적으로 double 형으로 인식되기 때문에 double형은 실수를 그냥 입력하고, float형으로 입력하려면 실수 뒤에 "f" 또는 "F"를, long double형으로 입력하려면 실수 뒤에 "l" 또는 "L"을 붙여 입력해야 합니다.

#### 문자열 선언 방법

문자열을 선언할 때는 배열로 선언해야 합니다. 배열로 선언하려면 변수명 뒤에 대괄호([ ])를 표시하면 됩니다. 배열에 대한 자세한 내용은 'Section 131 배열과 문자열'에서 학습합니다.

**예** int a = 5;

- int : 자료의 형식을 정수형으로 지정한다.
- a : 변수명을 a로 지정한다.
- 5 : 변수 a를 선언하면서 초기값으로 5를 저장한다.

**예제 1** 다음과 같이 변수를 선언할 때 변수에 저장되는 값을 확인하시오.

변수 선언	설명
char aa = 'A';	문자형 변수 aa에 문자 'A'를 저장한다. 문자형 변수에는 한 글자만 저장되며, 저장될 때는 아스키 코드값으로 변경되어 정수로 저장된다. aa가 저장하고 있는 값을 문자로 출력하면 'A'가 출력되지만 숫자로 출력하면 'A'에 대한 아스키 코드 65가 출력된다.
char bb = '1'	문자 변수 bb에 '1'을 저장한다. 숫자를 작은따옴표로 묶을 경우 문자로 인식된다.
short si = 32768;	짧은 정수형 변수 si에 32767을 넘어가는 값을 저장했기 때문에 오버플로가 발생한다.
int in = 32768;	정수형 변수 in에 32768이 저장된다.
float fl = 24.56f;	단정도* 실수형 변수 fl에 실수 24.56을 저장한다.
double dfl = 24.5678;	배정도* 실수형 변수 dfl에 24.5678을 저장한다.
double c = 1.23e-2;	배정도 실수형 변수 c에 1.23e-2*를 저장한다.

**예제 2** 다음과 같이 변수를 선언할 때 잘못된 이유를 확인하시오.

변수 선언	설명	올바른 변수 선언
char a = 1,2345e-3;	배정도 실수형 상수를 char형으로 선언했기 때문에 오류가 발생한다.	double a = 1,2345e-3;
short a = 1,5e3f;*	단정도 실수형 상수를 short형으로 선언했기 때문에 오류가 발생한다.	float a = 1,5e3f;
int a = '1';	문자형 상수를 int로 선언했기 때문에 오류가 발생한다.	char a = '1';
float a = 'A';	문자형 상수를 float로 선언했기 때문에 오류가 발생한다.	char a = 'A';
double a = "hello";	문자열 상수를 double로 선언했기 때문에 오류가 발생한다.	char a[]* = "hello";
long long a = 1,5784E300L;*	배정도 실수형 상수를 long long형으로 선언했기 때문에 오류가 발생한다.	long double a = 1,5784E300L;
char a = 10;	정수형 상수를 char형으로 선언했기 때문에 오류가 발생한다.	int a = 10;



## 기출문제 따라잡기

Section 127

이전기출

1. 다음 중 C언어에서 사용하는 기억 클래스에 속하지 않는 것은?

- ① dynamic                      ② auto  
③ static                        ④ register

C언어에서 사용하는 4가지 기억 클래스는 auto, extern, register, static입니다.

이전기출

2. 프로그래밍 언어에서 예약어란?

- ① 프로그래머가 미리 설정한 변수  
② 데이터를 저장할 수 있는 이름이 부여된 기억 장소  
③ 시스템이 알고 있는 특수한 기능을 수행하도록 이미 용도가 정해져 있는 단어  
④ 프로그램이 수행되는 동안 변하지 않는 값을 나타내는 단어

예약어란 '어떤 것을 확보해 두기 위해 미리 약속하는 것'이죠? 이런 의미가 포함된 내용을 찾아보세요.

이전기출

3. C언어에서 저장 클래스를 명시하지 않은 변수는 기본적으로 어떤 변수로 간주하는가?

- ① AUTO                        ② REGISTER  
③ STATIC                      ④ EXTERN

저장 클래스를 명시하지 않을 경우 자동(Auto)으로 지정되는 저장 클래스는 무엇 일까요?

이전기출

4. 다음 중 C언어에서 사용되는 예약어가 아닌 것은?

- ① case                        ② switch  
③ virtual                      ④ enum

C언어에서 사용되는 예약어에는 case, switch, enum, else, goto, break 등이 있습니다.

이전기출

5. 다음 중 C언어에서 문장을 끝마치기 위해 사용되는 기호는?

- ① 콤마(,)                      ② 온점(.)  
③ 세미콜론(;)                ④ 콜론(:)

이것이 없으면 컴파일시 에러가 발생한다고 했죠? 이런 문제는 틀리면 안됩니다.

이전기출

6. 다음 중 C언어의 변수명을 잘못 표현한 것은?

- ① ABC1                        ② abc  
③ 135                        ④ abc3

변수명의 첫 글자로 숫자는 사용할 수 없습니다. 첫 글자는 영문자나 \_(밑줄)로 시작해야 합니다.

출제예상

7. 컴퓨터가 명령을 처리하는 도중 발생하는 값을 저장하기 위한 공간으로, 변할 수 있는 값을 의미하는 것은?

- ① 상수                        ② 변수  
③ 예약어                      ④ 주석

값이 항상 고정된 것은 상수, 값이 변하는 것은 변수 기억해 두세요.

출제예상

8. 다음 중 변수명 작성 규칙에 대한 설명으로 옳지 않은 것은?

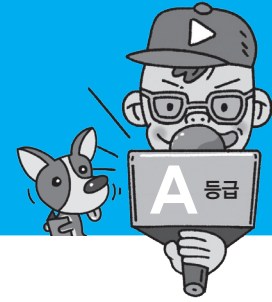
- ① 영문, 숫자, 언더바 등을 사용할 수 있다.  
② 글자 수에 제한이 없다.  
③ 공백을 사용할 수 없다.  
④ 예약어도 변수명으로 사용할 수 있다.

예약어는 변수명으로 사용할 수 없습니다.

▶ 정답 : 1. ① 2. ③ 3. ① 4. ③ 5. ③ 6. ③ 7. ② 8. ④

# SECTION 128 LIVE

## 연산자



### 전문가의 조언

산술 연산자, 관계 연산자, 비트 연산자, 논리 연산자, 조건 연산자 등에 해당하는 연산자의 종류를 기억하고, 연산자들의 개별적인 의미와 연산자 우선순위는 반드시 암기해야 합니다.

### 전문가의 조언

- 산술 연산자의 연산 우선 순위 (높음 → 낮음) : 증감 연산자 → 산술 연산자(\* / %) → 산술 연산자(+ -)
- 산술 연산자 중 \* / %는 우선순위가 같아 왼쪽에서 오른쪽 방향으로 놓인 순서대로 계산합니다.

## 1 산술 연산자

산술 연산자는 가, 감, 승, 제 등의 산술 계산에 사용되는 연산자를 말한다.

- 산술 연산자에는 일반 산술식과 달리 한 변수의 값을 증가하거나 감소시키는 증감 연산자가 있다.

연산자	의미	비고
+	덧셈	
-	뺄셈	
*	곱셈	
/	나눗셈	
%	나머지	
++	증가 연산자	• 전치 : 변수 앞에 증감 연산자가 오는 형태로 먼저 변수의 값을 증감시킨 후 변수를 연산에 사용한다(++a, --a).
--	감소 연산자	• 후치 : 변수 뒤에 증감 연산자가 오는 형태로 먼저 변수를 연산에 사용한 후 변수의 값을 증감시킨다(a++, a--).

### 문제 1 다음에 제시된 산술 연산식의 결과를 적으시오.

번호	산술 연산식	결과
①	10 + 15	
②	15 - 10	
③	3 * 5	
④	15 / 3	
⑤	15 % 2	
⑥	3 - 7 % 8 + 5	
⑦	-4 * 3 % -5 / 2	

⑥  $3 - 7 \% 8 + 5$

① (7)

② (-4)

③ (1)

⑦  $-4 * 3 \% -5 / 2$

① (-12)

② (-2)

③ (-1)

**결과** ① 25 ② 5 ③ 15 ④ 5 ⑤ 1 ⑥ 1 ⑦ -1

**문제 2** 다음에 제시된 산술 연산식의 결과를 적으시오(단 정수형 변수 a=2, b=3, c=4, d=5와 같이 선언되었다고 가정한다.).

번호	산술 연산식	결과
①	a = ++a + ++a;	
②	b = ++b - --c;	
③	c = ++b / b++;	
④	d = 10 % c++;	
⑤	a = ++c + c++ + ++c + c++;	
⑥	b = 10 + ++a;	
⑦	c = 10 - --d;	
⑧	c = ++a * b++;	

① a = ++a + ++a;

① ②  
③

- ① : a의 초기값이 2이고 ①이 전치 증가 연산자이므로 연산 전에 값이 증가하여 3이 됩니다.
- ② : a는 ①에서 3이 되었고 ②가 전치 증가 연산자이므로 연산 전에 값이 증가하여 4가 됩니다.
- ③ : ③을 수행하기 전에 a는 4가 된 상태이고 ③의 연산은 a+a와 같으므로 4+4의 결과인 8이 a에 저장됩니다.

② b = ++b - --c;

① ②  
③

- ① : b의 초기값이 3이고 ①이 전치 증가 연산자이므로 연산 전에 값이 증가하여 4가 됩니다.
- ② : c의 초기값이 4이고 ②가 전치 감소 연산자이므로 연산 전에 값이 감소하여 3이 됩니다.
- ③ : ①-②이므로 4-3의 결과인 1이 b에 저장됩니다.

③ c = ++b / b++;

① ②  
③

- ① : b의 초기값이 3이고 ①이 전치 증가 연산자이므로 연산 전에 값이 증가하여 4가 됩니다.
- ② : ②가 후치 증가 연산자이므로 연산에 사용되는 b는 ①에서 증가한 4가 됩니다.
- ③ : ③을 수행하기 전에 b는 4가 된 상태이고 ③의 연산은 b/b와 같으므로 4/4의 결과인 1이 c에 저장됩니다.

④ d = 10 % c++;

①  
②

- ① : c의 초기값이 4이고 ①이 후치 증가 연산자이므로 연산에 사용되는 c는 4가 됩니다.
- ② : 10 % 4의 결과인 2가 d에 저장됩니다.

⑤ a = ++c + c++ + ++c + c++;

① ③ ② ④  
⑤

- ① : c의 초기값이 4이고 ①이 전치 증가 연산자이므로 연산 전에 값이 증가하여 5가 됩니다.
- ② : c는 ①에서 5가 되었고 ②가 전치 증가 연산자이므로 연산 전에 값이 증가하여 6이 됩니다.
- ③ : ③이 후치 증가 연산자이므로 연산에 사용되는 c는 ①, ②에서 증가한 6이 됩니다.
- ④ : ④가 후치 증가 연산자이므로 연산에 사용되는 c는 ①, ②에서 증가한 6이 됩니다.
- ⑤ : ① + ② + ③ + ④ = 6 + 6 + 6 + 6 = 24



#### 전문가의 조언

연산자 우선 순위(높은 → 낮은)  
 증감 연산자 → 산술 연산자(\* / %)  
 → 산술 연산자(+ -) → 시프트 연  
 산자 → 관계 연산자(< > = >) →  
 관계 연산자(= != > <) → 비트 연산  
 자(& → ^ → |) → 논리 연산자(&&  
 → ||) → 조건 연산자 → 대입 연산  
 자 → 순서 연산자



#### 전문가의 조언

연산이 종료된 후 ②에 의해 후치  
 증가 연산이 적용되어 b는 5가 됩  
 니다.



#### 전문가의 조언

연산이 종료된 후 ③, ④에 의해 후  
 치 증가 연산이 적용되어 c는 8이  
 됩니다.

관계 연산자는 왼쪽을 기준으로 "왼쪽이 크다", "왼쪽이 크거나 같다"로 해석하면 됩니다.

⑥  $b = 10 + ++a;$



- ① : a의 초기값이 2이고 ①이 전치 증가 연산자이므로 연산 전에 값이 증가하여 3이 됩니다.
- ② :  $10 + ① = 10 + 3 = 13$

⑦  $c = 10 - --d;$



- ① : d의 초기값이 5이고 ①이 전치 감소 연산자이므로 연산 전에 값이 감소하여 4가 됩니다.
- ② :  $10 - ① = 10 - 4 = 6$

⑧  $c = ++a * b++;$



- ① : a의 초기값이 2이고 ①이 전치 증가 연산자이므로 연산 전에 값이 증가하여 3이 됩니다.
- ② : b의 초기값이 3이고 ②가 후치 증가 연산자이므로 연산에 사용되는 b는 초기값인 3이 됩니다.
- ③ :  $① * ② = 3 \times 3 = 9$

**결과** ① 8 ② 1 ③ 1 ④ 2 ⑤ 24 ⑥ 13 ⑦ 6 ⑧ 9

## 2 관계 연산자

관계 연산자는 두 수의 관계를 비교하여 참(true) 또는 거짓(false)을 결과로 얻는 연산자이다.

- 거짓은 0, 참은 1로 사용되지만 0외의 모든 숫자도 참으로 간주된다.

연산자	의미
$==$	같다
$!=$	같지 않다
$>$	크다*
$>=$	크거나 같다
$<$	작다
$<=$	작거나 같다

**문제** 다음 관계 연산식의 결과를 적으시오(단 정수형 변수  $a=5$ ,  $b=10$ 으로 선언되었다고 가정한다.).

번호	관계 연산식	결과
①	$a == 10$	
②	$b != 10$	
③	$a > 10$	
④	$b >= 10$	
⑤	$a < 10$	
⑥	$b <= 10$	




- ① a는 5이므로  $a == 10$ 은 거짓(false)입니다. 거짓은 0입니다.  
 ② b는 10이므로  $b != 10$ 은 거짓(false)입니다.  
 ③ a는 5이므로  $a > 10$ 은 거짓(false)입니다.  
 ④ b는 10이므로  $b >= 10$ 은 참(true)입니다. 참은 1입니다.  
 ⑤ a는 5이므로  $a < 10$ 은 참(true)입니다.  
 ⑥ b는 10이므로  $b <= 10$ 은 참(true)입니다.

**결과** ① 0 ② 0 ③ 0 ④ 1 ⑤ 1 ⑥ 1

### 3 비트 연산자

비트 연산자는 비트별(0, 1)로 연산하여 결과를 얻는 연산자이다.

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

비트 연산자 “|”는 키보드에서 엔터 키 위쪽의 를 Shift와 같이 누르면 입력되는 글자입니다.

**문제** 다음 비트 연산식의 결과를 적으시오(단 정수형 변수 a=5, b=7으로 선언되었다고 가정한다.).

번호	비트 연산식	결과
①	$a \& b$	
②	$a   b$	
③	$a \wedge b$	
④	$\sim b$	
⑤	$a \gg 1$	
⑥	$b \ll 3$	

① &(비트 and)는 두 비트가 모두 1일 때만 1이 되는 비트 연산자입니다.

C 언어에서 정수형 변수는 4바이트(32비트)이므로 각 변수의 값을 4바이트 이진수로 변환한 다음 비트별로 연산합니다.

```

5 = 0000 0000 0000 0000 0000 0000 0000 0101
7 = 0000 0000 0000 0000 0000 0000 0000 0111
& 0000 0000 0000 0000 0000 0000 0000 0101
0000 0000 0000 0000 0000 0000 0000 0101은 십진수로 5입니다.
```

② |(비트 or)는 두 비트 중 한 비트라도 1이면 1이 되는 비트 연산자입니다.

```

5 = 0000 0000 0000 0000 0000 0000 0000 0101
7 = 0000 0000 0000 0000 0000 0000 0000 0111
| 0000 0000 0000 0000 0000 0000 0000 0111
0000 0000 0000 0000 0000 0000 0000 0111은 십진수로 7입니다.
```

#### 패딩 비트

Shift에서 자리를 이동한 후 생기는 왼쪽이나 오른쪽 끝의 빈 자리에 채워지는 비트를 말합니다. C언어와 JAVA는 모두 부호화 2의 보수법을 사용하기 때문에 부호화 2의 보수법의 음수에 대한 패딩 비트만 알아두면 됩니다. 양수는 항상 빈 자리에 0이 채워지기 때문에 신경쓰지 않아도 됩니다.

• 양수 : Shift Left, Shift Right 모두 0이 채워집니다.

• 음수

- Shift Left : 왼쪽으로 이동하므로 오른쪽의 빈 자리에는 0이 채워집니다.

- Shift Right : 오른쪽으로 이동하므로 맨 왼쪽의 부호 비트를 제외한 빈 자리에 1이 채워집니다.

③ ^ (비트 xor)는 두 비트가 모두 같으면 0, 서로 다르면 1이 되는 연산자입니다.

```

5 = 0000 0000 0000 0000 0000 0000 0000 0101
7 = 0000 0000 0000 0000 0000 0000 0000 1111
^   0000 0000 0000 0000 0000 0000 0000 0110
0000 0000 0000 0000 0000 0000 0010은 십진수로 2입니다.

```

④ ~ (비트 not)는 각 비트의 부정을 만드는 연산자입니다.

```

7 = 0000 0000 0000 0000 0000 0000 0000 0111
~  1111 1111 1111 1111 1111 1111 1111 1000

```

부호화 2의 보수법을 사용하는 C언어나 JAVA에서는 맨 왼쪽의 비트는 부호 비트로, 0이면 양수이고 1이면 음수입니다. 원래의 값을 알기 위해서는 ...1111 1000에 대한 2의 보수를 구합니다. ...0000 1000은 십진수로 8이고 원래 음수였으므로 -를 붙이면 -8입니다.

⑤ >>는 오른쪽 시프트 연산자이므로, a에 저장된 값을 오른쪽으로 1비트 이동시킨 다음 그 값을 다시 a에 저장시킵니다. int는 4바이트이므로 4바이트 이진수로 변환하여 계산하면 됩니다.

• 4바이트에 5를 이진수로 표현하면 다음과 같습니다.

	32	31	30	...	20	...	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
5	0	0	0	...	0	...	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
													...	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
													...	256	128	64	32	16	8	4	2	1

부호 비트

부호를 제외한 전체 비트를 오른쪽으로 1비트 이동시킵니다. 부호는 맨 왼쪽의 0이고, 양수에 대한 패딩 비트\*에는 0이 들어옵니다.

	32	31	30	...	20	...	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
2	0	0	0	...	0	...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
													...	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
													...	256	128	64	32	16	8	4	2	1

부호 비트 패딩 비트

• 이것을 10진수로 변환하면 2입니다.

⑥ <<는 왼쪽 시프트 연산자이므로, b에 저장된 값을 왼쪽으로 3비트 이동시킨 다음 그 값을 다시 b에 저장시킵니다. 정수형 변수는 4바이트이므로 4바이트 이진수로 변환하여 계산하면 됩니다.

• 4바이트에 7을 이진수로 표현하면 다음과 같습니다.

	32	31	30	...	20	...	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
7	0	0	0	...	0	...	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
													...	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
													...	256	128	64	32	16	8	4	2	1

부호 비트

• 부호를 제외한 전체 비트를 왼쪽으로 3비트 이동시킵니다. 부호는 맨 왼쪽의 0입니다. 양수이므로 빈 자리(패딩 비트)에는 0이 들어오면 됩니다.

	32	31	30	...	20	...	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
56	0	0	0	...	0	...	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
													...	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
													...	256	128	64	32	16	8	4	2	1

부호 비트

패딩 비트

이것을 10진수로 변환하면 56(32+16+8)입니다.

**결과** ① 5 ② 7 ③ 2 ④ -8 ⑤ 2 ⑥ 56

## 4 논리 연산자

논리 연산자는 두 개의 논리 값을 연산하여 참(true) 또는 거짓(false)을 결과로 얻는 연산자이다. 관계 연산자와 마찬가지로 거짓은 0, 참은 1이다.

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

**문제** 다음 논리 연산식의 결과를 적으시오(단 정수형 변수 a=2, b=3, c=0, d=1, e=1로 선언되었다고 가정한다.).

번호	논리 연산식	결과
①	a > 3 && b > 2	
②	a > 3    b > 2	
③	!c	
④	a == 2 && b != 3	
⑤	a & b && c	
⑥	++d && --e	

① a > 3 && b > 2

① ②

③

- ① : a는 2이므로 a > 3은 거짓(false)입니다.
- ② : b는 3이므로 b > 2는 참(true)입니다.
- ③ : &&는 모두 참일 때만 참이므로 결과는 거짓(false)입니다. 거짓은 0입니다.

② a > 3 || b > 2

① ②

③

- ① : a는 2이므로 a > 3은 거짓(false)입니다.
- ② : b는 3이므로 b > 2는 참(true)입니다.
- ③ : ||는 하나라도 참이면 참이므로 결과는 참(true)입니다. 참은 1입니다.

③ c는 0이고 !c는 c의 부정이므로 결과는 1입니다.

④ a == 2 && b != 3

① ②

③

- ① : a는 2이므로 a == 2는 참(true)입니다.
- ② : b는 3이므로 b != 3은 거짓(false)입니다.
- ③ : &&는 모두 참일 때만 참이므로 결과는 거짓(false)입니다.



### 전문가의 조언

연산자 우선 순위(높은 → 낮은)  
 증감 연산자 → 산술 연산자(\* / %)  
 → 산술 연산자(+ -) → 시프트 연산자  
 → 관계 연산자(< > =) → 논리 연산자(&& → ^ → |)  
 → 비트 연산자(& → ^ → |) → 논리 연산자(&& → ||)  
 → 조건 연산자 → 대입 연산자 → 순서 연산자

⑤ a & b && c

①

②

- ① &(비트 and)는 두 비트가 모두 1일 때만 1이 되는 비트 연산자입니다.

C 언어에서 정수형 변수는 4바이트(32비트)이므로 각 변수의 값을 4바이트 이진수로 변환한 다음 비트별로 연산합니다.

2 = 0000 0000 0000 0000 0000 0000 0000 0010

3 = 0000 0000 0000 0000 0000 0000 0000 0011

& 0000 0000 0000 0000 0000 0000 0000 0010

0000 0000 0000 0000 0000 0000 0010은 십진수로 2입니다.

- ② : ① && c = 2 && 0 = 0

⑥ ++d && --e

①

②

③

- ① : d의 초기값이 1이고 ①이 전치 증가 연산자이므로 연산 전에 값이 증가하여 2가 됩니다.
- ② : e의 초기값이 1이고 ②가 전치 감소 연산자이므로 연산 전에 값이 감소하여 0이 됩니다.
- ③ : ① && ② = 2 && 0 = 0

결과 ① 0 ② 1 ③ 1 ④ 0 ⑤ 0 ⑥ 0

## 5 대입 연산자

연산 후 결과를 대입하는 연산식을 간략하게 입력할 수 있도록 대입 연산자를 제공합니다. 대입 연산자는 산술, 관계, 비트, 논리 연산자에 모두 적용할 수 있다.

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

**문제** 다음 대입 연산식의 결과를 적으시오(단 정수형 변수 a=2, b=3, c=4, d=5로 선언되었다고 가정한다.).

번호	대입 연산식	결과
①	$a += 3;$	
②	$b *= 3;$	
③	$c \% = 3;$	
④	$d >> = 1;$	
⑤	$c += 10 + ++a;$	
⑥	$d *= 10 - b++;$	
⑦	$a += b += c;$	
⑧	$d += b *= c /= a;$	
⑨	$a -= ++d / b--;$	
⑩	$b += c *= a << 2;$	
⑪	$a \% = c   b \& d - b;$	
⑫	$c *= d << (b == ++a);$	

④ 4바이트에 5를 이진수로 표현하면 다음과 같습니다.

	32	31	30	...	20	...	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
5	0	0	0	...	0	...	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

• 부호를 제외한 전체 비트를 오른쪽으로 1비트 이동시킵니다. 부호는 맨 왼쪽의 0이고, 양수에 대한 패딩 비트에는 0이 들어옵니다.

	32	31	30	...	20	...	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
2	0	0	0	...	0	...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

• 이것을 10진수로 변환하면 2입니다.

$$⑤ c += 10 + ++a \rightarrow c = c + (10 + ++a) \rightarrow c = c + (10 + 3) \rightarrow c = c + 13 \rightarrow c = 4 + 13$$

$$⑥ d *= 10 - b++ \rightarrow d = d * (10 - b++) \rightarrow d = d * (10 - 3) \rightarrow d = d * 7 \rightarrow d = 5 * 7$$

$$⑦ a += b += c \rightarrow a = a + (b += c) \rightarrow a = a + (b = b + c) \rightarrow a = a + (b = 3 + 4) \rightarrow a = a + 7 \rightarrow a = 2 + 7$$

$$⑧ d += b *= c /= a \rightarrow d = d + (b *= (c = c / a)) \rightarrow d = d + (b *= (c = 4 / 2)) \rightarrow d = d + (b *= 2) \rightarrow d = d + (b = b * 2) \rightarrow d = d + (b = 3 * 2) \rightarrow d = d + 6 \rightarrow d = 5 + 6$$

$$⑨ a -= ++d / b-- \rightarrow a = a - (++d / b--) \rightarrow a = a - (6 / 3) \rightarrow a = a - 2 \rightarrow a = 2 - 2$$

$$⑩ b += c *= a << 2 \rightarrow b = b + (c = c * (a << 2)) \rightarrow b = b + (c = c * 8) \rightarrow b = b + (c = 4 * 8) \rightarrow b = b + 32 \rightarrow b = 3 + 32$$

$$⑪ a \% = c | b \& d - b \rightarrow a = a \% (c | b \& d) \rightarrow a = a \% (4 | 3 \& d) \rightarrow a = a \% (7 \& d) \rightarrow a = a \% (7 \& 5) \rightarrow a = a \% 5 \rightarrow a = 2 \% 5$$

$$⑫ c *= d << (b == ++a) \rightarrow c = c * (d = d << (b == ++a)) \rightarrow c = c * (d = d << (3 == 3)) \rightarrow c = c * (d = d << 1) \rightarrow c = c * (d = 5 << 1) \rightarrow c = c * 10 \rightarrow c = 4 * 10$$

**결과** ① 5 ② 9 ③ 1 ④ 2 ⑤ 17 ⑥ 35 ⑦ 9 ⑧ 11 ⑨ 0 ⑩ 35 ⑪ 2 ⑫ 40



#### 전문가의 조언

연산자 우선 순위(높은 → 낮은)  
 증감 연산자 → 산술 연산자(\* / %)  
 → 산술 연산자(+ -) → 시프트 연산자  
 → 관계 연산자(< <= > >=) → 관계 연산자(== !=) → 비트 연산자(& → ^ → |) → 논리 연산자(&& → ||) → 조건 연산자 → 대입 연산자 → 순서 연산자



#### 전문가의 조언

##### ⑩ a << 2

부호를 제외한 전체 비트를 왼쪽으로 2비트 이동시킵니다. 부호는 맨 왼쪽의 0이고, 양수에 대한 패딩 비트에는 0이 들어옵니다.

$$2 = \dots 0000.0010$$

$$8 = \dots 0000.1000$$

##### ⑪

##### • 4 | 3

(비트 or)는 두 비트 중 한 비트라도 1이면 1이 되는 비트 연산자입니다.

$$4 = \dots 0000.0100$$

$$3 = \dots 0000.0011$$

$$| \dots 0000.0111(7)$$

##### • 7 & 5

(비트 and)는 두 비트가 모두 1일 때만 1이 되는 비트 연산자입니다.

$$7 = \dots 0000.0111$$

$$5 = \dots 0000.0101$$

$$\& \dots 0000.0101(5)$$

##### ⑫ 5 << 1

부호를 제외한 전체 비트를 왼쪽으로 1비트 이동시킵니다. 부호는 맨 왼쪽의 0이고, 양수에 대한 패딩 비트에는 0이 들어옵니다.

$$5 = \dots 0000.0101$$

$$10 = \dots 0000.1010$$

## 6 조건 연산자

조건 연산자는 조건에 따라 서로 다른 수식을 수행한다.

형식

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

**문제** 다음 조건 연산식의 결과를 적으시오(단 정수형 변수 a=1, b=2, c=3, d=4와 같이 선언되었다고 가정한다.).

번호	조건 연산식	결과
①	b *= a > b ? a : b;	
②	c -= a < b ? a - b : b - a;	
③	d %= c < d ? c++ : d++;	
④	c += b < b ? ++a : b++;	
⑤	d /= d % 3 ? a * b : d % c;	
⑥	a += ++a % b++ ? c * d : b / c;	

① b \*= a > b ? a : b;

①

②

- ① : a는 1이고 b는 2이므로 조건(a>b)이 거짓이 되어 b 값이 사용됩니다.
- ② : b = b \* b = 4

② c -= a < b ? a - b : b - a;

①

②

- ① : a는 1이고 b는 2이므로 조건(a<b)이 참이 되어 'a - b'의 결과인 -1이 사용됩니다.
- ② : c = c - (-1) = 3 + 1

③ d %= c < d ? c++ : d++;

①

②

- ① : c는 3이고 d는 4이므로 조건(c<d)이 참이 되어 c++의 결과인 4가 사용됩니다.
- ② : d = d % 3 = 4 % 3

④ c += b < b ? ++a : b++;

①

②

- ① : b는 2이므로 조건(b<b)이 거짓이 되어 b++의 결과인 3이 사용됩니다.
- ② : c = c + 3 = 3 + 3

⑤ d /= d % 3 ? a \* b : d % c;

①

②

- ① : d는 4이므로 'd % 3'은 1이 됩니다. 조건에서 1은 참과 같으므로 'a \* b'의 결과인 2가 사용됩니다.
- ② : d = d / 2 = 4 / 2

⑥ a += ++a % b++ ? c \* d : b / c;

①

②

- ① : a는 10이고 b는 20이므로 '++a % b++'은 '2 % 2'로 0이 됩니다. 조건에서 0은 거짓과 같으므로 'b / c'의 결과가 사용됩니다. 조건에서의 b++에 의해 후치 증가 연산이 적용되므로 b는 30이 되어 'b / c'의 결과는 1이 됩니다.
- ② : a = a + 1 = 2 + 1

결과 ① 4 ② 4 ③ 1 ④ 5 ⑤ 2 ⑥ 3

## 7 기타 연산자

연산자	의미
sizeof	자료형의 크기를 표시한다.
, (coma)	<ul style="list-style-type: none"> <li>• 콤마로 구분하여 한 줄에 두 개 이상의 수식을 작성하거나 변수를 정의한다.</li> <li>• 왼쪽에서 오른쪽으로 순서대로 수행되며, 순서 연산자라 부르기도 한다.</li> </ul>
(자료형)	<ul style="list-style-type: none"> <li>• 사용자가 자료형을 다른 자료형으로 변환할 때 사용하는 것으로, cast(캐스트) 연산자라고 부른다.</li> <li>• 변환할 자료형을 괄호 안에 넣어서 변환할 값이나 변수명 앞에 놓는다.</li> </ul> <p>예 a = (int)1.3 + (int)1.4; 1.3을 정수형으로 변환한 값 1과 1.4를 정수형으로 변환한 값 1이 더해진 2가 a에 저장된다.</p>

## 8 연산자 우선순위

- 한 개의 수식에 여러 개의 연산자가 사용되면 기본적으로 아래 표의 순서대로 처리된다.
- 아래 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)		
삼항 연산자	논리 연산자	&&(논리 and)   (논리 or)	→	↓
	조건 연산자	? :		
	대입 연산자	= += -= *= /= %= <(< >)= 등		
순서 연산자	순서 연산자	,	→	낮음



#### 전문가의 조언

- ① :  $-\neg a$ 에 의해 처음에는 2를 갖지만 ③의 전치 증가 연산이 적용되어 계산에 사용될 때는 3이 됩니다.
- ⑦ :  $c - ①$ 이 계산될 때 a는 ③의 전치 증가 연산이 적용된 후의 값인 3이 사용됩니다.

#### 문제

다음 연산식의 결과를 적으시오(단 정수형 변수  $a=3$ ,  $b=4$ ,  $c=5$ ,  $d=6$ 로 선언되었다고 가정한다.).

번호	연산식	결과
①	$a * b + c >= d \ \&\& \ d / a - b != 0$	
②	$d \% b + ++a * c -- \parallel c -- -a >= 10$	

①  $a * b + c >= d \ \&\& \ d / a - b != 0$

①(12)      ②(2)  
③(17)      ④(-2)  
⑤(1)      ⑥(1)  
⑦(1)

②  $d \% b + ++a * c -- \parallel c -- -a >= 10$

⑤(2)    ③(3)    ②(5)    ①(3)  
④(15)  
⑥(17)      ⑦(2)  
⑧(0)  
⑨(1)

결과 ① 1 ② 1



#### 기출문제 따라잡기

Section 128

이전기출

1. 다음의 관계 연산자 중 “A와 B가 같지 않다”의 의미를 갖는 것은?

- ①  $A < > B$       ②  $A != B$   
③  $A < = B$       ④  $A \& = B$

관계 연산자를 나열해 볼까요? '같다'는  $==$ , '크다'는  $>$ , '작다'는  $<$ , '크거나 같다'는  $> =$ , '작거나 같다'는  $< =$ , 그럼 '같지 않다'는? '같다'를 조금 변형해 보면 되지 않을까요?

이전기출

2. 다음 중 나머지를 구하는 연산자는?

- ①  $\%$       ②  $@$   
③  $\#$       ④  $!$

이런 문제는 틀리면 안 되겠죠?

이전기출

3. 다음 중 관계 연산자에 해당하지 않는 것은?

- ①  $<$       ②  $\%$   
③  $==$       ④  $!=$

관계 연산자는 두 항의 크기를 비교하는 것으로 '같다', '같지 않다', '크다', '크거나 같다', '작다', '작거나 같다'가 있습니다. 이와 관련 없는 것을 찾아보세요.

이전기출

4. 다음의 연산자 중 오른쪽에서 왼쪽으로의 결합규칙을 따르지 않는 것은?

- ①  $sizeof$       ②  $<<$   
③  $!$       ④  $++$

연산자 중에서 오른쪽에서 왼쪽으로의 결합 규칙을 따르는 것은 단항 연산자와 대입 연산자입니다.





## 기출문제 따라잡기

Section 128

출제예상

5. 어떤 자료형을 다른 자료형으로 바꾸고 싶을 때 사용하는 연산자는?

- ① 산술 연산자                      ② 관계 연산자  
③ 논리 연산자                      ④ 캐스트 연산자

산술은 더하고 빼고 곱하고 나누는 것. 관계는 크기를 비교하는 것. 논리는 두 개의 논리값을 연산하는 것입니다. 그럼 캐스트 연산자의 역할은?

출제예상

6. 다음 중 연산자 우선 순위가 옳은 것은? (단, 오른쪽 마지막 연산자가 가장 높은 우선순위를 가짐)

- ① +=, &, ==, <<, +, \*, ++  
② +=, <<, &, ==, +, \*, ++  
③ +=, ==, &, <<, +, \*, ++  
④ +=, &, ==, +, \*, <<, ++

연산자 우선 순위(낮음~높음)  
순서 → 대입 → 조건 → 이항(논리 → 비트 → 관계 → 시프트 → 산술) → 단항

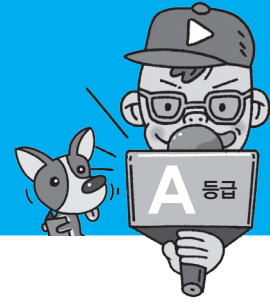
이전기출

7. 다음 중 비트 단위 논리 연산자의 종류에 해당되지 않는 것은?

- ① ^                                      ② ~  
③ &                                      ④ ?

비트 단위 논리 연산자에는 비트 not, 비트 and, 비트 xor, 비트 or가 있습니다.

▶ 정답 : 1. ② 2. ① 3. ② 4. ② 5. ④ 6. ① 7. ④



## 전문의의 조언

i문의 문법을 묻는 문제나 i문이 포함된 코드를 제시하고 그 결과를 묻는 문제가 출제될 것으로 예상됩니다. 형식별로 주어진 예제를 확실히 이해하고 넘어가세요.

### 반복문

반복문은 일정한 횟수를 반복하는 명령문으로 다음 섹션에서 자세히 공부합니다.

## 1 제어문의 개념

컴퓨터 프로그램은 명령어가 서술된 순서에 따라 무조건 위에서 아래로 실행되는데, 조건을 지정해서 진행 순서를 변경할 수 있다. 이렇게 프로그램의 순서를 변경할 때 사용하는 명령문을 제어문이라고 한다.

- 제어문의 종류에는 if문, 다중 if문, switch문, goto, 반복문\* 등이 있다.

## 2 단순 if문

if문은 조건에 따라서 실행할 문장을 달리하는 제어문이며, 단순 if문은 조건이 한 개일 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)**

i는 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

**if(조건)**

**{ 실행할 문장1;**

{ } 사이에 조건이 참일 경우 실행할 문장을 적는다.

**실행할 문장2;**

**...**

**}**

### #include <stdio.h>

표준 입·출력과 관련된 함수를 정의해 놓은 파일의 이름입니다. 헤더 파일이라고 하는데, 사용하는 함수에 따라 포함시켜야 할 헤더 파일이 다릅니다. 여기서는 ③번 문장의 printf( ) 함수를 사용하기 때문에 포함시켰습니다. 무슨 함수를 쓸 때 어떤 헤더 파일을 포함시켜야 하는지는 문제 풀이와 관계 없기 때문에 기억해 둘 필요는 없습니다.

### 예제 1 a가 10보다 크면 a에서 10을 빼기

**#include <stdio.h>**

**main( )**

**{**

**int a = 15, b;**

**if (a > 10) ①**

a가 10보다 크면 ②번 문장을 실행하고, 아니면 ③번 문장으로 이동해서 실행을 계속한다.

**b = a - 10; ②**

①번의 조건식이 참일 경우 실행할 문장이다. b는 5가 된다.

```
printf("%d\n", b); ❸
```

여기서는 ❶번의 조건식이 거짓일 경우 실행할 문장이 없다. 조건 판단문을 벗어나면 무조건 ❸번으로 온다.

결과 5

- 형식2 : 조건이 참일 때와 거짓 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.

**예제 2** a가 b보다 크면 'a-b', 아니면 'b-a'를 수행하기

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

```
main( )
```

```
{
```

```
    int a = 10, b = 20, cha;
```

```
    if (a > b) ❶
```

a가 b보다 크면 ❷번 문장을 실행하고, 아니면 ❸번의 다음 문장인 ❹번 문장을 실행한다.

```
        cha = a - b; ❷
```

❶번의 조건식이 참일 경우 실행할 문장이다. 참이 아니기 때문에 초기화 시키지 않은 cha에는 알 수 없는 값이 그대로 있게 된다.

```
    else ❸
```

❶번의 조건식이 거짓일 경우 실행할 문장의 시작점이다.

```
        cha = b - a; ❹
```

❶번의 조건식이 거짓일 경우 실행할 실제 처리문이다. cha는 10이 된다.

```
    printf("%d\n", cha);
```

결과 10

### 3 다중 if문

다중 if문은 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.

**예제 1** 점수에 따라 등급 표시하기

```
#include <stdio.h>
main( )
{
    int jum = 85;
    if (jum >= 90) ❶
        printf("학점은 A입니다.\n"); ❷
    else if (jum >= 80) ❸
        printf("학점은 B입니다.\n"); ❹
    else if (jum >= 70) ❺
        printf("학점은 C입니다.\n"); ❻
    else ❼
        printf("학점은 F입니다.\n"); ❽
} ❾
```

jum이 90 이상이면 ❷번을 실행하고, 아니면 ❾번으로 이동한다.

"학점은 A입니다."를 출력하고, ❾번으로 이동하여 프로그램을 종료한다.

jum이 80 이상이면 ❹번을 실행하고, 아니면 ❾번으로 이동한다.

"학점은 B입니다."를 출력하고, ❾번으로 이동하여 프로그램을 종료한다.

jum이 70 이상이면 ❻번을 실행하고, 아니면 ❼번으로 이동한다.

"학점은 C입니다."를 출력하고, ❾번으로 이동하여 프로그램을 종료한다.

❽번의 조건식이 거짓일 경우 ❸번을 실행한다.

"학점은 F입니다."를 출력하고, ❾번으로 이동하여 프로그램을 종료한다.

결과 학점은 B입니다.

- 형식2 : if문 안에 if문이 포함된다.

```
if(조건1)
{
    if(조건2)
        실행할 문장1;
    else
        실행할 문장2;
}
else
    실행할 문장3;
```

조건1이 참일 경우 실행할 문장의 시작점이다.

조건2가 참일 경우 실행할 문장을 적는다.

조건2가 거짓일 경우 실행할 문장을 적는다.

조건1이 거짓일 경우 실행할 문장을 적는다.

**예제 2** 홀수, 짝수 판별하기

```
#include <stdio.h>
main( )
{
    int a = 21, b = 10;
    if (a % 2 == 0) ❶
        if (b % 2 == 0) ❷
            printf("모두 짝수\n"); ❸
    else ❹
```

a를 2로 나눈 나머지가 0이면 ❷번을 실행하고, 아니면 ❹번으로 이동한다.

b를 2로 나눈 나머지가 0이면 ❸번을 실행하고, 아니면 ❹번으로 이동한다.

"모두 짝수"를 출력하고, ❹번으로 이동하여 프로그램을 종료한다.

❷번의 조건식이 거짓일 경우 ❹번을 실행한다.

```

printf("a : 짝수, b : 홀수\n"); ⑤ "a : 짝수, b : 홀수"를 출력하고, ⑩번으로 이
                               동하여 프로그램을 종료한다.
else ⑥                          ①번의 조건식이 거짓일 경우 실행할 문장의 시작점이다.
    if (b % 2 == 0) ⑦            b를 2로 나눈 나머지가 0이면 ⑧번을 실행하고, 아니면 ⑨번으로 이동한다.
        printf("a : 홀수, b : 짝수\n"); ⑧ "a : 홀수, b : 짝수"를 출력하고, ⑩번으로 이
                                         동하여 프로그램을 종료한다.
    else ⑨                      ⑦번의 조건식이 거짓일 경우 실행할 문장의 시작점이다.
        printf("모두 홀수\n"); ⑩ "모두 홀수"를 출력하고, ⑩번으로 이동하여 프로그램을 종로
                                         한다.
} ⑪                             결과 a : 홀수, b : 짝수

```

## 4 switch문

switch문은 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리할 수 있는 제어 문이다.

### • 형식

```

switch(수식) ①
{
    case 레이블1: ③
        실행할 문장1;
        break;
    case 레이블2: ④
        실행할 문장2;
        break;
        :
    default:
        실행할 문장3;
} ⑤

```

- switch는 switch문에 사용되는 예약어로 그대로 입력한다.
- 수식 : '레이블1' ~ '레이블n'의 값 중 하나를 도출하는 변수나 수식을 입력한다.
- ②~⑤번이 switch문의 범위이다. '{'로 시작해서 '}'로 끝난다. 반드시 입력해야 한다.
- case는 switch문에서 레이블을 지정하기 위한 예약어로 그대로 입력해야 한다.
- 레이블1 : ①번 식의 결과가 될 만한 값 중 하나를 입력한다. 결과가 '레이블'과 일치하면 이곳으로 찾아온다. 식의 결과가 5종류로 나타나면 case문이 5번 나와야 한다.
- ①번 식의 결과가 ③번의 '레이블'과 일치할 때 실행할 문장이다.
- switch문을 탈출하여 ⑤번으로 간다.
- ①번의 식의 결과가 '레이블2'와 일치하면 찾아오는 곳이다.
- ①번의 식의 결과가 ④번의 '레이블2'와 일치할 때 실행할 문장이다.
- switch문을 탈출하여 ⑤번으로 간다.
- ①번의 식의 결과가 '레이블' ~ '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.

#### 예 switch (2)

```

{ case 3: printf("1");
  case 2: printf("2");
  case 1: printf("3");
} → 결과 23

```



#### 전문가의 조언

#문은 조건이 참과 거짓인 경우의 두 가지만 판별하여 제어를 이동해야 하므로 분기할 문장이 여러 곳일 경우 #문을 여러 번 중첩해 사용해야 하는 불편함이 있었습니다. 이때 switch문을 사용하면 간단하게 해결할 수 있습니다. switch문의 구조도 간단해 예제를 보면 의미를 쉽게 이해할 수 있습니다. 문법도 같이 기억해 두세요.



#### 전문가의 조언

default는 가장 마지막에 사용되어 다음 문장에서 바로 종료되기 때문에 break를 사용할 필요가 없습니다.

**예제** 점수(jum)에 따라 등급 표시하기

```
#include <stdio.h>
main( )
{
    int jum = 85;
    switch (jum / 10)
    {
        ❶
        case 10:
            case 9: ❷
                printf("학점은 A입니다.\n"); ❸ "학점은 A입니다."를 출력한다.
                break; ❹ break를 만나면 switch문을 탈출하여 ❹번으로 이동한다.
            case 8: ❺ "jum/10"이 8일 경우 찾아오는 곳이다. ❹, ❷번을 실행한다.
                printf("학점은 B입니다.\n"); ❻ "학점은 B입니다."를 출력한다.
                break; ❼ switch문을 탈출하여 ❹번으로 이동한다.
            case 7:
                printf("학점은 C입니다.\n");
                break;
            case 6:
                printf("학점은 D입니다.\n");
                break;
            default:
                case 10~6에 해당되지 않는 경우, 즉 jum이 59 이하인 경우 찾아오는 곳이다.
                printf("학점은 F입니다.\n"); "학점은 F입니다."를 출력한다.
            } ❽
    } ❾
```

jum을 10으로 나눠 결과에 해당하는 숫자를 찾아간다. 85/10은 8.5지만 C언어에서 정수 나눗셈은 결과도 정수이므로 결과는 8이다. 8에 해당하는 ❹번으로 이동하여 ❹, ❷번을 실행한다.

❶~❹번까지가 switch 조건문의 범위이다.

100점일 경우 'jum/10'의 결과인 10이 찾아오는 곳이지만 할 일은 'case 9'와 같으므로 아무것도 적지 않는다. 아무것도 적지 않으면 다음 문장인 ❷번으로 이동한다.

'jum/10'이 9일 경우 찾아오는 곳이다. ❹, ❷번을 실행한다.

break를 만나면 switch문을 탈출하여 ❹번으로 이동한다.

'jum/10'이 8일 경우 찾아오는 곳이다. ❹, ❷번을 실행한다.

switch문을 탈출하여 ❹번으로 이동한다.

결과 **학점은 B입니다.**



**전문가의 조언**

C언어로 실무 프로그램을 작성할 때는 goto문을 거의 사용하지 않지만 시험을 위한 C언어 문법의 출제 범위가 명확하지 않아 포함한 내용이니 가볍게 읽어 두세요.

## 5 goto문

goto문은 프로그램 실행 중 현재 위치에서 원하는 다른 문장으로 건너뛰어 수행을 계속하기 위해 사용하는 제어문이다.

- goto문은 원하는 문장으로 쉽게 이동할 수 있지만 많이 사용하면 프로그램의 이해와 유지 보수가 어려워져 거의 사용하지 않는다.
- 형식

**goto 레이블;** 레이블로 이동한다.

**레이블:** goto문의 주소값이다. '레이블' 형태로 작성되며, 레이블명은 사용자가 원하는 이름을 임의로 지정할 수 있다.

**실행할 문장**

**예제** 10보다 큰 값을 입력할 때까지 입력하기

```
#include <stdio.h>
main()
{
    int a; ❶
again: ❷
    scanf("%d", &a); ❸
    if (a <= 10) ❹
        goto again; ❺
    else ❻
        printf("%d는 10보다 큼니다.", a); ❼
} ❽
```

goto문의 주소값이다. 'again'이라는 레이블의 goto 주소이다.

정수형 변수 a에 10진수를 입력 받는다.

a의 값이 10 이하면 ❺번을 실행한다.

❶번과 ❸번 사이에 있는 'again' 레이블로 이동한다.

❹번의 조건식이 거짓일 경우 실행할 문장의 시작점이다.

변수 a의 값을 출력한 후 ❽번으로 이동하여 프로그램을 종료한다.



### 기출문제 따라잡기

Section 129

출제예상

1. 다음 C 프로그램의 출력 결과는?

```
#include <stdio.h>
void main(void)
{
    int a = 3, b = 10;
    if(b > 5)
        printf("%x\n", a + b);
    else
        printf("%x\n", b - a);
}
```

- ❶ 7                      ❷ 13  
❸ d                      ❹ a

```
#include <stdio.h>
void main(void)
{
    ❶ int a = 3, b = 10;
    ❷ if(b > 5)
    ❸ printf("%x\n", a + b);
        else
    ❹ printf("%x\n", b - a);
}
```

- ❶ 정수형 변수 a, b를 선언하고 a를 3으로, b를 10으로 초기화합니다.
- ❷ b가 5보다 크면 ❸을 실행하고, 그렇지 않으면 ❹을 실행합니다.
- ❸ 'a+b'를 실행하면 13이고, 이것을 16진수로 출력한(d) 후 커서를 다음 줄 앞으로 이동합니다.
- %x는 16진수를 출력하는 서식 지정자로, printf문의 자세한 내용은 Section 138을 참조하세요.
- 16진수 a는 10, b는 11, c는 12, d는 13, e는 14, f는 15입니다.
- ❹ 'b-a'를 실행하면 7이고, 이것을 16진수로 출력한(7) 후 커서를 다음 줄 앞으로 이동합니다.

▶ 정답 : 1. ❸



## 기출문제 따라잡기

Section 129

출제예상

2. 다음은 C 언어의 일부분이다. 실행 결과는?

```
char cc;
int dd = 50;
switch(dd)
{ case 50 : cc='x';
  case 30 : cc='y';
  default : cc='z'; }
printf("%c", cc);
```

- ① x                      ② y  
③ z                      ④ error 발생

break문이 없으므로 case문 전체가 실행됩니다. 그러므로 cc에는 가장 마지막 실행문에 의해 'z'가 저장됩니다.

출제예상

3. 다음 C 프로그램의 실행 결과는?

```
#include<stdio.h>
main()
{
  int a = 10;
  if (a == 10)
    printf("a는");
    printf("%d입니다.", a);
  else
    printf("a는");
    printf("%d이 아닙니다.", a);
}
```

- ① a는 10입니다.  
② error 발생  
③ a는 10이 아닙니다.  
④ 10입니다.

#문에서 조건이 참이거나 거짓일 때 실행할 문장이 두 문장 이상이면 중괄호({})를 입력하고 그 사이에 실행할 문장을 입력해야 하는데 중괄호({}) 없이 두 문장을 입력했으므로 error가 발생합니다.

출제예상

4. 다음 C 프로그램을 실행했을 경우 출력되는 값은 얼마인가?

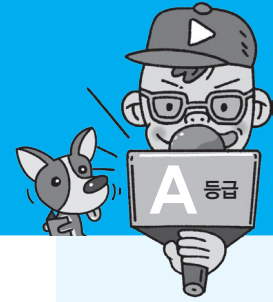
```
main()
{
  int a = 100;
  if (a = 200)
    a = 300;
  else
    a = 400;
  printf("%d\n", a);
}
```

- ① 100                      ② 200  
③ 300                      ④ 400

C언어에서는 값이 같은지 비교할 때는 관계 연산자 ==를 사용해야 합니다. if (a = 200)처럼 대입연산자 =을 이용하면 a에 200이 저장되고, 200은 참이므로 'a = 300'을 수행합니다.

▶ 정답 : 2. ③ 3. ② 4. ③





## 1 반복문의 개요

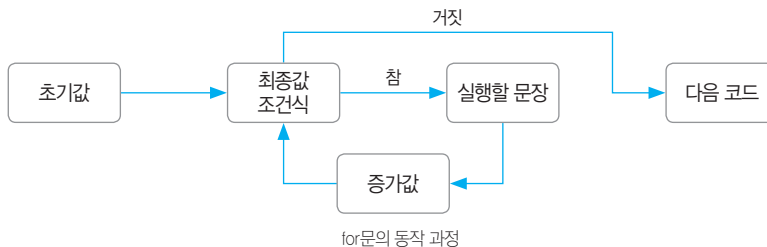
반복문은 제어문의 한 종류로 일정한 횟수를 반복하는 명령문을 말한다. 보통 변수의 값을 일정하게 증가시키면서 정해진 수가 될 때까지 명령이나 명령 그룹을 반복 수행한다.

- 반복문에는 for, while, do~while문이 있다.

## 2 for문

for문은 초기값, 최종값, 증가값을 지정하는 수식을 이용해 정해진 횟수를 반복하는 제어문이다.

- for문은 초기값을 정한 다음 최종값에 대한 조건이 참이면 실행할 문장을 실행한 후 초기값을 증가값 만큼 증가시키면서 최종값에 대한 조건이 참인 동안 실행할 문장을 반복 수행한다.



- 형식

**for(식1; 식2; 식3)**

- for는 반복문을 의미하는 예약어로 그대로 입력한다.
- 식1 : 초기값을 지정할 수식을 입력한다.
- 식2 : 최종값을 지정할 수식을 입력한다.
- 식3 : 증가값으로 사용할 수식을 입력한다.

**실행할 문장:**

식2가 참일 동안 실행할 문장을 입력한다. 실행할 문장이 두 문장 이상일 경우 { }를 입력하고 그 사이에 처리할 문장들을 입력한다.

- 초기값, 최종값, 증가값 중 하나 이상을 생략할 수 있고, 각각의 요소에 여러 개의 수식을 지정할 수도 있다.

**예 1** for(a = 1; sum <= 30; sum += a) : 증가값을 생략하고 실행할 문장에서 증가값을 만들

**예 2** for(a = 0; sum <= 10; a++, sum += a) : 증가값(a++, sum += a)을 두 개 지정함

**예 3** for(a = 0, b = 5; a <= 5, b >= 0; a++, b--) : 초기값, 최종값, 증가값을 모두 두 개씩 지정함



### 전문가의 조언

반복문에는 for, while, do~while문이 있습니다. for문은 초기값, 최종값, 증가값을 이용하고, while문과 do~while문은 조건이 참인 동안 반복문 실행한다는 것을 기억하세요. 그리고 반복문을 제어하는 break와 continue의 기능도 정리하세요.

### 예 1의 이해

반복 변수 a가 1에서 시작(초기값)하여 sum이 30보다 작거나 같은 동안(최종값) sum += a를 반복하여 수행합니다.

### 예 2의 이해

반복 변수 a가 0에서 시작(초기값)하여 a = a+1, sum = sum+a로 증가(증가값)하면서 sum이 10보다 작거나 같은 동안(최종값) 실행할 문장을 반복하여 수행합니다.

### 예 3의 이해

반복 변수 a가 0, b는 5에서 시작(초기값)하여 a = a+1, b = b-1로 증가(증가값)하면서 a가 5보다 작거나 같고 b가 0보다 큰 동안(최종값) 실행할 문장을 반복하여 수행합니다.

#### 예의 이해

반복 변수 a가 A부터 시작(초기값)하여 a = a+1, 즉 오름차순(A, B, C, ...)으로 증가(증가값)하면서 2일때 까지(최종값) 실행할 문장을 반복하여 수행합니다.

※ 한 글자만 저장하는 문자형 변수는 실제로 해당 문자를 저장하는 것이 아니라 해당 문자의 아스키 코드를 저장합니다. 아스키 코드에서 65 = A, 66 = B, ... 90 = Z이므로, 1씩 더한다는 것은 다음 문자를 가리키는 것과 동일합니다.

for문으로 실행할 문장이 한 줄인 경우 다음과 같이 입력해도 됩니다.

```
for(a = 1; a <= 5; a++) hap += a;
```

- for( ; ; )와 같이 조건에 참여하는 수식을 모두 생략하면 for문은 무한 반복한다.
- for문은 처음부터 최종값에 대한 조건식을 만족하지 못하면 한 번도 수행하지 않는다.
- 문자도 for문을 수행할 수 있다.

예 for(char a = 'A'; a <= 'Z'; a++) a에 'A, B, C ~ X, Y, Z' 순으로 저장함

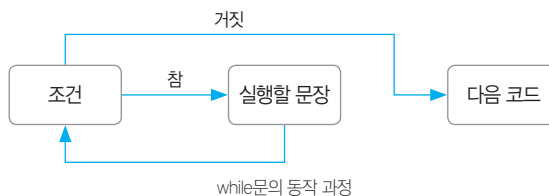
**예제** for문을 이용하여 1~5까지의 합을 더하는 다양한 방법이다.

코드	설명
<pre>int a, hap = 0; for (a = 1; a &lt;= 5; a++)     hap += a; ❶</pre>	반복 변수 a가 1에서 시작(초기값)하여 1씩 증가(증가값)하면서 5보다 작거나 같은 동안(최종값) ❶번을 반복하여 수행한다. 그러니까 'hap += a'를 5회 실행하며, 결과는 15이다.
<pre>int a, hap = 0; for (a = 0; a &lt; 5; a++, hap += a);</pre>	for문의 마지막에 ';'이 있으므로 실행할 문장 없이 for문만 반복 수행한다.
<pre>int a = 1, hap = 0; for (; a &lt;= 5; a++)*     hap += a; ❶</pre>	초기값을 지정하지 않았지만 변수 a 선언 시 1로 초기화 했기 때문에 a가 1부터 5보다 작거나 같은 동안 ❶번을 반복하여 수행한다.
<pre>int a, hap = 0; for (a = 0; a++ &lt; 5;)     hap += a; ❶</pre>	증가값을 지정하지 않았지만 최종값에서 'a++'를 수행하기 때문에 a가 0부터 5보다 작은 동안 ❶번을 반복하여 수행한다.
<pre>int a = 1, hap = 0; for (; a &lt;= 5;) {     hap += a;     a++; ❷ } ❸</pre>	초기값과 증가값을 지정하지 않았지만 변수 a 선언 시 1로 초기화 했고, ❷번을 수행하면서 a가 1씩 증가하기 때문에 a가 1부터 5보다 작거나 같은 동안 ❶~❸번 사이의 실행할 문장을 반복하여 수행한다. 실행할 문장이 2개 이상인 경우 중괄호 ({ })로 묶어준다.

### 3 while문

while문은 조건이 참인 동안 실행할 문장을 반복 수행하는 제어문이다.

- while문은 조건이 참인동안 실행할 문장을 반복 수행하다가 조건이 거짓이면 while문을 끝낸 후 다음 코드를 실행한다.
- while문은 조건이 처음부터 거짓이면 한 번도 수행하지 않는다.



- 형식

#### while(조건)

- while은 반복문에 사용되는 예약어로 그대로 입력한다.
- (조건) : 참이나 거짓을 결과로 갖는 수식을 '조건'에 입력한다. 참(1)\*을 직접 입력할 수도 있다.

#### 실행할 문장;

조건이 참인 동안 실행할 문장을 입력한다. 문장이 두 문장 이상인 경우 { }를 입력하고 그 사이에 처리할 문장들을 입력한다.

while(1)과 같이 조건을 1로 지정하면 while문을 무한 반복합니다.

**예제** 다음은 1~5까지의 합을 더하는 프로그램이다. 결과를 확인하시오.

```
#include <stdio.h>
main()
{
    int a = 0, hap = 0;
    while (a < 5) ❶
    { ❷
        a++; ❸
        hap += a; ❹
    } ❺
    printf("%d, %d\n", a, hap); ❻
}
```

a가 5보다 작은 동안 ❷~❹번 문장을 반복하여 수행한다.  
 ❷~❹번까지가 반복문의 범위이다. 반복문에서 실행할 문장이 하나인 경우는 { }를 생략해도 된다.  
 'a = a + 1;'과 동일하다. a의 값을 1씩 증가시킨다.  
 'hap = hap + a;'와 동일하다. a의 값을 hap에 누적시킨다.  
 반복문의 끝이다.  
 결과 5, 15  
 i가 5가 되었을 때 5를 hap에 누적한 다음 while 문을 벗어나기 때문에 i는 5로 끝난다.

반복문 실행에 따른 변수 a와 hap의 변화

a	hap
0	0
1	1
2	3
3	6
4	10
5	15

## 4 do~while문

do~while문은 조건이 참인 동안 정해진 문장을 반복 수행하다가 조건이 거짓이면 반복문을 벗어나는 while문과 같은 동작을 하는데, 다른 점은 do~while문은 실행할 문장을 무조건 한 번 실행한 다음 조건을 판단하여 탈출 여부를 결정한다는 것이다.

- do~while문은 실행할 문장을 우선 실행한 후 조건을 판별하여 조건이 참이면 실행할 문장을 계속 반복 수행하고, 거짓이면 do~while문을 끝낸 후 다음 코드를 실행한다.



- 형식

**do** do는 do~while문에 사용되는 예약어로, do~while의 시작 부분에 그대로 입력한다.

**실행할 문장;** 조건이 참인 동안 실행할 문장을 입력한다. 문장이 두 문장 이상인 경우 { }를 입력하고 그 사이에 실행할 문장들을 입력한다.

**while(조건);** while은 do~while문에 사용되는 예약어로, do~while의 끝 부분에 그대로 입력한다.  
 • (조건) : 참이나 거짓을 결과로 갖는 수식을 '조건'에 입력한다. 참()을 직접 입력할 수도 있다.

**예제** 다음은 1부터 10까지 홀수의 합을 더하는 프로그램이다. 결과를 확인하시오.

```
#include <stdio.h>
main ()
{
    int a = 1, hap = 0;
```

반복문 실행에 따른 변수 hap과 a의 변화

hap	a
0	1
1	3
4	5
9	7
16	9
25	11

반복문 실행에 따른 변수 a와 hap의 변화

a	hap
0	0
1	1
2	
3	4
4	
5	9
6	

```
do ①
{ ②
    hap += a; ③
    a += 2; ④
} while(a < 10); ⑤
printf("%d, %d\n", a, hap); ⑥
```

do~while 반복문의 시작점이다. ②~⑤번 사이의 문장을 반복하여 수행한다.

②~⑤번까지가 반복문의 범위이다.

'hap = hap + a'와 동일하다. a의 값을 hap에 누적시킨다.

'a = a + 2'와 동일하다. a의 값을 2씩 증가시킨다.

a가 10보다 작은 동안 ②~⑤번 사이의 문장을 반복하여 수행한다.

결과 11, 25

a가 9가 되었을 때 9를 hap에 누적한 다음 a에 2를 더 해 a가 11이 되었을 때 do~while문을 벗어나기 때문에 a는 11로 끝난다.

## 5 break, continue

switch문이나 반복문의 실행을 제어하기 위해 사용되는 예약어이다.

- break : switch문이나 반복문 안에서 break가 나오면 블록을 벗어난다.
- continue
  - continue 이후의 문장을 실행하지 않고 제어를 반복문의 처음으로 옮긴다.
  - 반복문에서만 사용된다.

**예제** 다음은 1~5까지의 합을 더하되 2의 배수는 배제하는 프로그램이다. 결과를 확인하시오.

```
#include <stdio.h>
main( )
```

```
{
    int a = 0, hap = 0;
```

```
    while(1) ①
```

조건이 참(1)이므로 무한 반복한다. 중간에 반복을 끝내는 문장이 반드시 있어야 한다.

```
    { ②
```

②~④번까지가 반복문의 범위이다.

```
        a++; ③
```

'a = a + 1'과 동일하다. a의 값을 1씩 증가시킨다.

```
        if(a > 5) ④
```

a가 5보다 크면 ⑤번 문장을 수행하고, 아니면 ⑥번 문장을 수행한다.

```
            break; ⑤
```

반복문을 탈출하여 ⑩번으로 이동한다.

```
            if (a % 2 == 0) ⑥
```

a를 2로 나눈 나머지가 0이면, 즉 a가 2의 배수이면 ⑦번 문장을 수행하고, 아니면 ⑧번 문장으로 이동한다.

```
                continue; ⑦
```

이후의 문장, 즉 ⑧번을 생략하고 반복문의 처음인 ②번으로 이동한다. 2의 배수는 hap에 누적되지 않는다.

```
                hap += a; ⑧
```

'hap = hap + a'와 동일하다. a의 값을 hap에 누적시킨다.

```
        } ⑨
```

반복문의 끝이다.

```
    printf("%d, %d\n", a, hap); ⑩
```

결과 6, 9



## 기출문제 따라잡기

Section 130

이전기출

1. C언어의 FOR문, COBOL 언어의 PERFORM문에 해당하는 것은?

- ① 반복문                      ② 종료문  
③ 입출력문                  ④ 선언문

지금 공부하고 있는 섹션명이 무엇이죠?

이전기출

2. 다음 C 프로그램에서 최종적으로 출력되는 n, t의 값을 순서대로 나열하면?

```
void main(void)
{
    int n = 0, t = 0;
    do
    {
        t += n;
        printf("n=%2d, t=%2d\n", n++, t);
    }
    while(n < 10);
}
```

- ① 10, 55                      ② 9, 45  
③ 10, 45                      ④ 9, 55

사용된 코드의 의미는 다음과 같습니다.

```
void main(void)
{
    ① int n = 0, t = 0;
    ② do
    {
        ③ t += n;
        ④ printf("n=%2d, t=%2d\n", n++, t);
    }
    ⑤ while(n < 10);
}
```

- ① 정수형 변수 n과 t를 선언한 후 0으로 초기화 합니다.  
② do~while 반복문의 시작점입니다. ③, ④번 문장을 반복하여 수행합니다.  
③ 't = t + n;'과 동일합니다. n의 값을 t에 누적시킵니다.  
④ n++와 t를 정수형 10진수로 출력한 후 커서를 다음 줄 맨 처음으로 이동합니다.  
- printf(): 표준 출력 함수  
- %2d: 두 자리의 정수형 10진수로 출력  
- \n: 커서를 다음 줄 맨 처음으로 이동  
- n++: 'n = n + 1'과 동일한데, 변수 뒤에 연산자가 있는 후위 연산이므로 먼저 변수를 명령문에 사용한 다음 변수의 값을 증가시킵니다. 즉 n의 값을 출력한 후 1을 증가시킵니다.  
⑤ n이 10보다 작은 동안 ③, ④번 문장을 반복하여 수행합니다.  
반복문 실행에 따른 변수의 변화는 아래와 같습니다.

반복횟수	n	t	출력
	0	0	
1	1	0	n= 0, t= 0
2	2	1	n= 1, t= 1
3	3	3	n= 2, t= 3
4	4	6	n= 3, t= 6
5	5	10	n= 4, t= 10
6	6	15	n= 5, t= 15
7	7	21	n= 6, t= 21
8	8	28	n= 7, t= 28
9	9	36	n= 8, t= 36
10	10	45	n= 9, t= 45

이전기출

3. 다음 C 코드 결과로 나타날 수 있는 값은?

```
void main()
{
    int k;
    k = 1;
    while(k < 60)
    {
        if(k % 4 == 0)
            printf("%d\n", k-2);
        k++;
    }
}
```

- ① 0                              ② 8  
③ 24                            ④ 30

사용된 코드의 의미는 다음과 같습니다.

```
void main() {
    ① int k;
    ② k = 1;
    ③ while(k < 60)
    {
        ④ if(k % 4 == 0)
        ⑤ printf("%d\n", k-2);
        ⑥ k++;
    }
}
```

- ① 정수형 변수 k를 선언합니다.  
② k를 1로 초기화합니다.  
③ k가 60보다 작은 동안 ④~⑥ 사이의 문장을 반복하여 수행합니다.  
④ k를 4로 나눈 나머지가 0이면 ⑤번을 실행합니다.

▶ 정답: 1. ① 2. ② 3. ④



## 기출문제 따라잡기

Section 130

⑤ 'k-2'의 결과를 정수형 10진수로 출력한 후 커서를 다음 줄 맨 처음으로 이동합니다.

- printf(): 표준 출력 함수
- %d: 정수형 10진수로 출력
- \n: 커서를 다음 줄 맨 처음으로 이동

⑥ 'k = k + 1;'과 동일합니다. k의 값을 1 증가시킵니다.

반복문 실행에 따른 변수의 변화는 아래와 같습니다.

반복횟수	k < 60	k % 4	k	출력
			1	
1	False	1	2	
2	False	2	3	
3	False	3	4	
4	False	0	5	2
5	False	1	6	
6	False	2	7	
7	False	3	8	
8	False	0	9	6
9	False	1	10	
10	False	2	11	
11	False	3	12	
12	False	0	13	10
⋮	⋮		⋮	⋮

위와 같이 k가 60보다 작은 동안 실행하면 2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54가 출력됩니다.

이전기술

4. C언어에서 반복 처리를 위한 명령문이 아닌 것은?

- ① for
- ② while
- ③ continue
- ④ do ~ while

반복문을 제어하는 기능 2가지! break와 continue! 잊지마세요.

이전기술

5. C언어에서 while 문이 수행될 때, 종괄호로 둘러싸인 while문의 몸체는 몇 번 수행되는가?

```
int sum = 0;
int i = 1;
while(sum < 20)
{
    sum = sum + i;
    i = i + 1;
}
```

- ① 1
- ② 3
- ③ 6
- ④ 20

반복문 실행에 따른 변수의 변화는 아래와 같습니다.

반복횟수	sum	i
	0	1
1	1	2
2	3	3
3	6	4
4	10	5
5	15	6
6	21	7

※ sum이 20보다 작은 동안이므로 총 6회 수행됩니다.

※ sum이 15가 되었을 때 20보다 작다는 조건에 맞으므로 한 번 더 수행하여 sum은 21이 되고, i는 7이 되었을 때 반복문을 종료합니다.

이전기술

6. 프로그램 언어의 문장구조 중 성격이 다른 하나는?

- ① while(expression) statement;
- ② for(expression-1; expression-2; expression-3) statement;
- ③ if(expression) statement-1; else statement-2;
- ④ do {statement;} while(expression);

반복문은 3가지입니다. while, for, do!

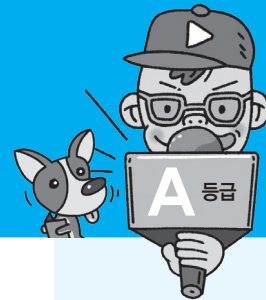
이전기술

7. C언어의 do ~ while문에 대한 설명 중 틀린 것은?

- ① 문의 조건이 거짓인 동안 루프처리를 반복한다.
- ② 문의 조건이 처음부터 거짓일 때도 문을 최소 한번은 실행한다.
- ③ 무조건 한 번은 실행하고 경우에 따라서는 여러 번 실행하는 처리에 사용하면 유용하다.
- ④ 문의 맨 마지막에 “;”이 필요하다.

do~while문은 조건이 참인 동안 루프 처리를 반복합니다.

▶ 정답 : 4. ③ 5. ③ 6. ③ 7. ①



## 1 배열의 개념

배열은 동일한 데이터 유형을 여러 개 사용해야 할 경우 이를 손쉽게 처리하기 위해 여러 개의 변수들을 조합해서 하나의 이름으로 정의해 사용하는 것을 말한다.

- 배열은 하나의 이름으로 여러 기억장소를 가리키기 때문에 배열에서 개별적인 요소들의 위치는 첨자를 이용하여 지정한다.
- 배열은 변수명 뒤에 대괄호 [ ]를 붙이고 그 안에 사용할 개수를 지정한다.
- C언어에서 배열의 위치는 0부터 시작된다.
- 배열은 행 우선으로 데이터가 기억장소에 할당된다.
- C 언어에서 배열 위치를 나타내는 첨자 없이 배열 이름을 사용하면 배열의 첫 번째 요소의 주소를 지정하는 것과 같다.

## 2 1차원 배열

- 1차원 배열은 변수들을 일직선상의 개념으로 조합한 배열이다.
- 형식

자료형 변수명[개수];

- 자료형 : 배열에 저장할 자료의 형을 지정한다.
- 변수명 : 사용할 배열의 이름으로 사용자가 임의로 지정한다.
- 개수 : 배열의 크기를 지정하는 것으로 생략할 수 있다.

예 int a[5]\* : 5개의 요소를 갖는 정수형 배열 a

	첫 번째	두 번째	세 번째	네 번째	다섯 번째
배열 a	a[0]	a[1]	a[2]	a[3]	a[4]

※ a[3] : a는 배열의 이름이고, 3은 첨자로서 배열 a에서의 위치를 나타낸다. a[3]에 4를 저장시키려면 'a[3] = 4'와 같이 작성한다.

예제 1 1차원 배열 a의 각 요소에 10, 11, 12, 13, 14를 저장한 후 출력하기

```
#include <stdio.h>
main( )
{
    int a[5];
```

5개의 요소를 갖는 정수형 배열 a를 선언한다. 선언할 때는 사용할 개수를 선언하고, 사용할 때는 첨자를 0부터 사용하므로 주의해야 한다.

	첫 번째	두 번째	세 번째	네 번째	다섯 번째
배열 a	a[0]	a[1]	a[2]	a[3]	a[4]

### 전문가의 조언

배열은 편리하고 유용한 자료 구조지만 확실하게 이해하지 못하면 효율적으로 사용하기 힘든 자료 구조이므로 학습 유도를 위해 자주 출제될 것으로 예상됩니다.

예제를 통해 배열의 사용법과 특징을 확실히 이해하고 넘어가세요.

### int a[5]

배열을 선언할 때는 int a[5]와 같이 5를 입력하여 5개의 요소임을 선언하며 사용할 때는 a[0]~a[4]까지 5개의 요소를 사용한다.

```
int i;
for (i = 0; i < 5; i++)
```

```
    a[i] = i + 10; ❶
```

```
for (i = 0; i < 5; i++)
```

```
    printf("%d ", a[i]); ❷
```

```
}
```

정수형 변수 `i`를 선언한다

반복 변수 `i`가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ❶번 문장을 반복하여 수행한다. 그러니까 ❶번 문장을 5회 반복하는 것이다.

배열 `a`의 `i`번째에 `i+10`을 저장시킨다. `i`는 0~4까지 변화하므로 배열 `a`에 저장된 값은 다음과 같다.

배열 a	10	11	12	13	14
	a[0]	a[1]	a[2]	a[3]	a[4]

반복 변수 `i`가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ❷번 문장을 반복하여 수행한다.

배열 `a`의 `i`번째를 출력한다. `i`는 0~4까지 변화하므로 출력 결과는 다음과 같다. 서식 문자열에 `\n`이 없기 때문에 한 줄에 붙여서 출력한다.

결과 10 11 12 13 14



## JAVA에서의 배열 처리

JAVA에서는 항상된 for문을 사용할 수 있는데, 항상된 for문은 객체를 대상으로만 가능합니다. JAVA에서는 배열을 객체로 취급하며, 배열을 이용하여 작업할 때 필요할 만한 내용은 이미 API로 만들어 두었기 때문에 잘 골라서 사용하면 됩니다. 배열에 대한 기본은 C언어에서 배웠기 때문에 바로 예제를 보면서 설명합니다.

**예제 1** 다음은 위의 C 프로그램을 JAVA로 구현한 것이다. 프로그램의 출력 결과를 확인해 보시오.

```
public class Example {
    public static void main(String[] args){
        ❶ int a[] = new int[5];
        ❷ int i;
        ❸ for (i = 0; i < 5; i++)
        ❹     a[i] = i + 10;
        ❺ for (i = 0; i < 5; i++)
        ❻     System.out.printf("%4d ", a[i]);
    }
}
```

### 코드 해설

❶ 배열을 선언하는 부분이 조금 다릅니다. 배열은 JAVA에서 객체로 취급되며, 객체 변수\*는 'new' 명령을 이용해서 생성해야 합니다.

• `int a[]` : `a`는 정수 배열 변수라는 의미입니다. JAVA에서는 'int[]'과 같이 대괄호를 자료형 바로 뒤에 적는 것을 선호하는데, C언어에서는 이렇게 표기할 수 없습니다.

• `new int[5]` : 5개짜리 정수 배열을 생성합니다.

• C언어처럼 사용하려면 다음과 같이 배열을 선언하면서 초기값을 주면 됩니다.

```
int a[] = {0,0,0,0,0};
```

②~⑤ C 프로그램과 동일합니다.

⑥ 결과 10 11 12 13 14

### 객체 변수

객체 변수, 정확히 말하면 heap 영역에 객체를 생성하고 생성된 객체가 있는 곳의 주소를 객체 변수에 저장하는 것입니다. JAVA에서는 주소를 제어할 수 없으므로 그냥 객체 변수를 생성한다고 이해해도 됩니다.



**예제 2** 다음은 JAVA에서 향상된 for문을 사용한 예제이다. 결과를 확인하시오.

```
public class Example {
    public static void main(String[] args) {
        ① int[] a = {90,100,80,70,60,50,30};
        ② int hap = 0;
        ③ float avg;
        ④ for (int i : a)
        ⑤     hap = hap + i;
        ⑥ avg = (float)hap / a.length;
        ⑦ System.out.printf("%4d, %4.2f", hap, avg);
    }
}
```

#### 코드 해설

- ① 배열을 선언하면서 초기값을 지정합니다. 개수를 정하지 않으면 초기값으로 지정된 수만큼 배열의 요소가 만들어 집니다. 이걸 C언어와 같습니다.

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
배열 a	90	100	80	70	60	50	30

- ② 총점을 구하기 위해 정수형 변수 hap을 선언하고 초기값으로 0을 할당합니다.  
 ③ 평균을 구하기 위해 실수형 변수 avg를 선언합니다.  
 ④ 향상된 반복문입니다. a 배열의 요소 수만큼 ⑤번을 반복 수행합니다.
- int i : a 배열의 각 요소가 일시적으로 저장될 변수를 선언합니다. a 배열과 형이 같아야 합니다. a 배열이 정수면 정수, 문자면 문자여야 합니다.
  - a : 배열의 이름을 입력합니다. a 배열이 7개의 요소를 가지므로 각 요소를 i에 저장하면서 ⑤번을 7번 수행합니다.

- ⑤ i의 값을 hap에 누적합니다. i는 a 배열 각 요소의 값을 차례로 받습니다.  
 변수의 변화는 다음과 같습니다.

첫 번째 수행 : a 배열의 첫 번째 값이 i를 거쳐 hap에 누적됩니다.

hap	i	a						
90	90	90	100	80	70	60	50	30

두 번째 수행 : a 배열의 두 번째 값이 i를 거쳐 hap에 누적됩니다.

hap	i	a						
190	100	90	100	80	70	60	50	30

세 번째 수행 : a 배열의 세 번째 값이 i를 거쳐 hap에 누적됩니다.

hap	i	a						
270	80	90	100	80	70	60	50	30

이런 방식으로 a 배열의 요소 수만큼 반복합니다.

- ⑥ 총점이 저장되어 있는 hap을 실수형으로 변환한 후 a 배열의 요소 수로 나눠 평균을 구합니다.
- length : 클래스에는 클래스의 속성과 수행해야 할 메소드가 포함되어 있습니다. length는 배열 클래스의 속성으로 배열 요소의 개수가 저장되어 있습니다. a 배열은 7개의 요소를 가지므로 a.length는 7을 가지고 있습니다.
  - a.length : 개체 변수의 이름과 메소드는 .(마침표)로 연결하여 사용합니다.
- ⑦ 결과 480, 68.57



#### 전문가의 조언

C언어에서는 향상된 FOR문을 사용할 수 없습니다.

### 3 2차원 배열

- 2차원 배열은 변수들을 평면, 즉 행과 열로 조합한 배열이다.
- 형식

자료형 변수명[행개수][열개수]

- 자료형 : 배열에 저장할 자료의 형을 지정한다.
- 변수명 : 사용할 배열의 이름으로 사용자가 임의로 지정한다.
- 행개수 : 배열의 행 크기를 지정한다.
- 열개수 : 배열의 열 크기를 지정한다.

**예** int b[3][3] : 3개의 행과 3개의 열을 갖는 정수형 배열 b



b[0][2] : b는 배열의 이름이고, 0은 행 첨자, 2는 열 첨자로서 배열 b에서의 위치를 나타낸다.

**예제** 3행 4열의 배열에 다음과 같이 숫자 저장하기

배열 a

1	2	3	4
5	6	7	8
9	10	11	12

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

```
main( )
```

```
{
```

```
int a[3][4];
```

```
int i, j, k = 0;
```

```
for (i = 0; i < 3; i++) ①
```

3행 4열의 크기를 갖는 정수형 배열 a를 선언한다.

정수형 변수 i, j, k를 선언하고 k를 0으로 초기화 한다.

반복 변수 i가 0에서 시작하여 1씩 증가하면서 3보다 작은 동안 ②~③번을 반복하여 수행한다. 결국 ③번 문장을 3회 반복한다.

```
{ ②
```

②~③이 ①번 반복문의 반복 범위이지만 실제 실행할 문장은 ③번 하나이다.

```
for (j = 0; j < 4; j++) ③
```

반복 변수 j가 0에서 시작하여 1씩 증가하면서 4보다 작은 동안 ④~⑦번을 반복하여 수행한다.

• i가 0일 때 j는 0에서 3까지 4회 반복

• i가 1일 때 j는 0에서 3까지 4회 반복

• i가 2일 때 j는 0에서 3까지 4회 반복 수행하므로 ⑤~⑦번을 총 12회 수행한다.

```
{ ④
```

④~⑦이 ③번 반복문의 반복 범위이다.

```
k++; ⑤
```

k를 1씩 증가시킨다. k는 총 12회 증가하므로 1~12까지 변한다.

```
a[i][j] = k; ⑥
```

배열 a의 i행 j열에 k를 기억시킨다. a[0][0]~a[2][3]까지 1~12가 저장된다.

```
} ⑦
```

④번의 짝이다.

```
} ⑧
```

②번의 짝이다.

```
}
```

첫 번째 { 의 짝이자 프로그램의 끝이다.

## 4 배열의 초기화

- 배열 선언 시 초기값을 지정할 수 있다.
- 배열을 선언할 때 배열의 크기를 생략하는 경우에는 반드시 초기값을 지정해야 초기값을 지정한 개수 만큼의 배열이 선언된다.

### 예 1차원 배열 초기화

방법 1 `char a[3] = {'A', 'B', 'C'}`

방법 2 `char a[] = {'A', 'B', 'C'}`

배열 a	A	B	C
	a[0]	a[1]	a[2]

### 예 2차원 배열 초기화

방법 1 `int a[2][4] = { {10, 20, 30, 40}, {50, 60, 70, 80} };`

방법 2 `int a[2][4] = {10, 20, 30, 40, 50, 60, 70, 80}`

	a[0][0]	a[0][1]	a[0][2]	a[0][3]
배열 a	10	20	30	40
	50	60	70	80
	a[1][0]	a[1][1]	a[1][2]	a[1][3]

- 배열의 초기 값을 '숫자,(콤마)'로 지정하면 배열의 첫 번째 요소에는 지정한 숫자가 입력되고, 나머지 요소에는 0이 입력된다.

예 `int a[5] = { 3, };`

배열 a	3	0	0	0	0
	a[0]	a[1]	a[2]	a[3]	a[4]

**예제** 2차원 배열에 다음과 같이 초기화 한 후 a[0][0]과 a[1][1]의 값 출력하기

배열 a

10	20	30	40
50	60	70	80

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

```
main( )
```

```
{
```

```
    int a[2][4] = {
        {10, 20, 30, 40},
        {50, 60, 70, 80}
    };
```

```
    printf("%d ", a[0][0]);
    printf("%d\n", a[1][1]);
}
```

2행 4열의 정수형 배열 a를 선언한 후 값을 할당한다.

	a[0][0]	a[0][1]	a[0][2]	a[0][3]
배열 a	10	20	30	40
	50	60	70	80
	a[1][0]	a[1][1]	a[1][2]	a[1][3]

a[0][0]의 값 10을 출력한다.

a[1][1]의 값 60을 출력한다. '\n'으로 인해 커서는 다음 줄로 이동한다.

## 5 배열 형태의 문자열 변수

- C언어에서는 큰따옴표(“ ”)로 묶인 글자는 글자 수에 관계없이 문자열로 처리된다.
- C언어에는 문자열을 저장하는 자료형이 없기 때문에 배열, 또는 포인터를 이용하여 처리한다.
- 형식

```
char 배열이름[크기] = “문자열”
```

- 배열에 문자열을 저장하면 문자열의 끝을 알리기 위한 널 문자('\0')가 문자열 끝에 자동으로 삽입된다.
- 배열에 문자열을 저장할 때는 배열 선언 시 초기값으로 지정해야 하며, 이미 선언된 배열에는 문자열을 저장할 수 없다.
- 문자열 끝에 자동으로 널 문자('\0')가 삽입되므로, 널 문자까지 고려하여 배열 크기를 지정해야 한다.

예 char a[5] = “love” → 

l	o	v	e	\0
---	---	---	---	----

**예제** 다음의 출력 결과를 확인하시오.

```
#include <stdio.h>
main( )
{
    char a = 'A';
    char b[9] = "SINAGONG";
    char *c = "SINAGONG";

    printf("%c\n", a);
    printf("%s\n", b);
    printf("%s\n", c);
}
```

문자형 변수 a에 문자 'A'를 저장한다. 문자형 변수에는 한 글자만 저장되며, 저장될 때는 아스키 코드값으로 변경되어 정수로 저장된다. a가 저장하고 있는 값은 문자로 출력하면 'A'가 출력되지만 숫자로 출력하면 'A'에 대한 아스키 코드 65가 출력된다.

9개의 요소를 갖는 배열 b를 선언하고 다음과 같이 초기화한다. 저장되는 글자는 8자이지만 문자열의 끝에 자동으로 저장되는 널 문자('\0')를 고려하여 크기를 9로 지정한 것이다.

S	I	N	A	G	O	N	G	\0
b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]

포인터 변수 c에 “SINAGONG”이라는 문자열이 저장된 곳의 주소를 저장한다.

변수 a의 값을 문자로 출력한다.

배열 위치를 나타내는 첨자 없이 배열 이름을 사용하면 배열의 첫 번째 요소의 주소를 지정하는 것과 같으므로 배열 b의 첫 번째 요소가 가리키는 곳의 값을 문자열로 출력한다.

포인터 변수 c가 가리키는 곳의 값을 문자열로 출력한다.

**결과**

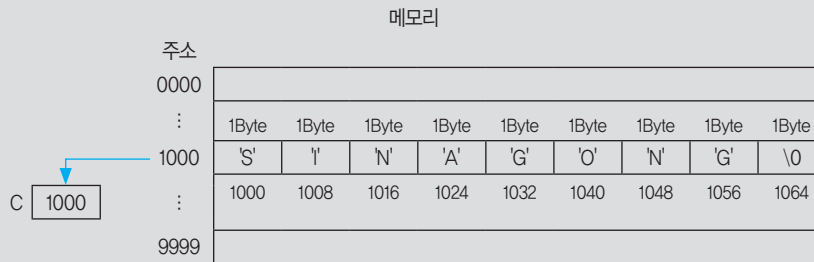
```
A
SINAGONG
SINAGONG
```



### 전문가의 조언

포인터 변수는 다음 섹션에서 배웁니다. 지금은 '포인터에는 문자열의 시작 주소가 기억되는 구나' 하는 정도로만 이해하고 넘어가세요.

## 코드 해설



위 코드 중 ❶번을 실행할 경우 메모리를 그려보면 다음과 같다.



## JAVA의 문자열

C언어에서는 문자열을 배열에 넣고 배열의 이름을 이용하든지 포인터 변수를 이용해 처리했지만 JAVA에서는 주소를 컨트롤하는 기능이 없기 때문에 불가능합니다. 하지만 JAVA에서는 문자열을 처리할 수 있도록 클래스를 제공합니다. 클래스를 제공하므로 당연히 그에 따른 속성과 메소드도 지원하는데 여기서는 문제 풀이에 꼭 필요한 속성과 메소드만 언급하도록 하겠습니다.

**예제** 다음은 문자열을 거꾸로 출력하는 JAVA 프로그램이다. 결과를 확인하시오.

```
public class Example {
    public static void main(String[] args){
        ❶ String str = "Information!";
        ❷ int n = str.length();
        ❸ char[] st = new char [n];
        ❹ n--;
        ❺ for (int k = n; k >= 0; k--) {
        ❻ st[n-k] = str.charAt(k);
        ❼ }
        ❽ for (char k : st) {
        ❾ System.out.printf("%c", k);
        }
    }
}
```

## 코드 해설

- 문자열 변수 str을 선언하면서 초기값으로 "Information!"을 할당한다. 객체 변수를 생성할 때는 예약어 new를 입력해야 하지만 문자열 변수는 초기값을 이용해 new 없이 바로 생성할 수 있다.
- 문자열 클래스에서 length() 메소드는 해당 문자열의 길이를 반환한다. 즉 정수형 변수 n에 str에 저장된 문자열의 길이 12가 저장된다.
- JAVA에서는 배열도 클래스이므로 생성할 때는 new를 사용해야 한다. n에 12가 저장되어 있으므로 st는 12개의 요소를 갖는 문자 배열로 생성된다.



## 전문가의 조언

여기서 지정한 주소는 임의로 정한 것이며, 이해를 돕기 위해 주소를 실제 표현되는 16진수가 아니라 10진수로 표현했습니다.

- `char[] st` : 문자 배열이고 배열 이름은 `st`이다. `'char st[]'`처럼 입력해도 된다. `st`만 임의로 입력하고 나머지는 그대로 적어준다.
- `new` : 객체 변수를 생성하는 예약어다. 그대로 입력한다.
- `char [n]` : 문자 배열의 크기를 지정하므로 12개 요소를 갖는 문자 배열이 생성된다. `n`을 제외한 나머지는 항상 그대로 입력한다.

④ 12개짜리 배열이지만 배열의 첨자는 0부터 시작하여 11까지 사용하기 때문에 첨자로 사용할 변수의 값을 1 감소시킨다.

`n = 11`

	<code>st[0]</code>	<code>st[1]</code>	<code>st[2]</code>	<code>st[3]</code>	<code>st[4]</code>	<code>st[5]</code>	<code>st[6]</code>	<code>st[7]</code>	<code>st[8]</code>	<code>st[9]</code>	<code>st[10]</code>	<code>st[11]</code>
<code>st</code>												

⑤ 정수형 변수 `k`를 반복 변수로 선언하면서 초기값으로 11을 갖고, 1씩 감소시키면서 0보다 크거나 같은 동안 ⑥번을 반복 실행한다. 실행할 문장이 한 개이므로 ⑥, ⑦번의 종결호는 없어도 된다. 반복 변수 `k`는 `for`문 안에서 선언한 지역 변수이기 때문에 `for`문을 벗어나면 소멸된다.

⑥ `charAt()` 메소드는 해당 문자열에서 인수에 해당하는 위치의 문자열을 반환한다.

첫 번째 수행

- `k = 11, n = 11`이므로 `str.charAt(k)`는 `'!'`을 반환한다. `'!'`을 `st[n-k]`번째, 즉 `st[0]`번째에 저장한다.

<code>st[0]</code>	<code>st[1]</code>	<code>st[2]</code>	<code>st[3]</code>	<code>st[4]</code>	<code>st[5]</code>	<code>st[6]</code>	<code>st[7]</code>	<code>st[8]</code>	<code>st[9]</code>	<code>st[10]</code>	<code>st[11]</code>
!											

두 번째 수행

`k = 10, n = 11`이므로 `str.charAt(k)`는 `'n'`을 반환한다. `'n'`을 `st[1]`에 저장한다.

<code>st[0]</code>	<code>st[1]</code>	<code>st[2]</code>	<code>st[3]</code>	<code>st[4]</code>	<code>st[5]</code>	<code>st[6]</code>	<code>st[7]</code>	<code>st[8]</code>	<code>st[9]</code>	<code>st[10]</code>	<code>st[11]</code>
!	n										

위와 같은 작업을 반복수행하다가 `k = 0`일 때, `'l'`를 `st[11]`에 저장한 다음 `k`는 `-1`이 되어 반복문을 벗어난다.

<code>st[0]</code>	<code>st[1]</code>	<code>st[2]</code>	<code>st[3]</code>	<code>st[4]</code>	<code>st[5]</code>	<code>st[6]</code>	<code>st[7]</code>	<code>st[8]</code>	<code>st[9]</code>	<code>st[10]</code>	<code>st[11]</code>
!	n	o	i	t	a	m	r	o	f	n	l

⑧ 향상된 반복문이다. `st` 배열의 각 요소를 처음부터 차례대로 문자 변수 `k`에 옮기면서 `st` 배열의 개수, 즉 ⑨번을 12회 반복 수행한다.

⑨ `k`에는 `st` 배열 각 요소의 값이 할당되므로 12회 수행을 마치면 출력 결과는 다음과 같다. 서식 문자열에 `'\n'`이 없으므로 한 줄에 붙여 출력한다.

결과 `!noitamrofni`



## 기출문제 따라잡기

## Section 131

출제예상

1. 동일한 성격을 가진 자료들의 모임을 무엇이라고 하는가?

- ① 스트링(string)                      ② 배열(array)  
 ③ 구조체(structue)                    ④ 열거형(enumeration)

이렇게 쉬운 문제는 틀리면 안되겠조. 배열은 중요하니 꼭 기억해 두세요.

이전기출

2. C언어에서 정수가 2Byte로 표현되고, "int a[2][3]"로 선언된 배열의 첫 번째 자료가 1000 번지에 저장되었다. 이때 a[1][1] 원소가 저장된 주소는?

- ① 1002                                      ② 1004  
 ③ 1006                                      ④ 1008

C언어에서는 배열 위치가 0부터 시작하는데, 문제에서 정수가 2Byte로 표현된다고 했으므로 "int a[2][3]"로 선언된 정수형 배열은 다음과 같이 첫 번째 자료의 위치인 1000번지부터 2Byte씩 할당됩니다.

주소						
0						
:						
1000	a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]
:	1000	1002	1004	1006	1008	1010

∴ a[1][1] 원소가 저장된 곳의 주소는 1008입니다.

출제예상

3. C언어에서 'int a[3][2] = {10, 11, 12, 13, 14, 15};'로 선언하였을 경우 a[1][1]의 값은?

- ① 12    ② 13  
 ③ 14    ④ 15

C언어에서 배열의 첨자는 0부터 시작하므로 a[1][1]은 a 배열의 2행 2열을 의미합니다. 배열 a는 다음과 같이 초기화됩니다.

10	11
12	13
14	15

▶ 정답 : 1. ② 2. ④ 3. ②



## 전문의가의 조언

포인터의 개념이 어려울 수 있습니다. 하지만 포인터는 중요한 개념이므로 반드시 이해하고 넘어가야 합니다. 포인터의 개념과 더불어 포인터 변수의 용도도 기억해 두세요. 참고로 JAVA에서는 포인터 변수를 사용할 수 없습니다.

## 전문의가의 조언

**포인터의 개념**  
앞에서 어떤 수나 문자를 저장하기 위해 변수를 사용했습니다. 사실 이 변수는 기억장소의 어느 위치에 대한 이름이며 그 위치는 주소로도 표현할 수 있습니다. 우리는 친구 홍길동의 집에 모이기 위해 "홍길동네 집으로 와"라고 말하기도 하지만 홍길동의 집 주소인 "서울시 마포구 서교동 00번지로 와"라고 말하기도 합니다. C언어에서는 변수의 주소를 포인터라고 하고, 포인터를 저장할 수 있는 변수를 포인터 변수라고 합니다. 변수의 주소인 포인터는 출력할 수도 있고 포인터가 가리키는 곳에 값을 저장하거나 읽어 오는 등 다양한 조작이 가능합니다. 이런 기능 때문에 C언어는 주소를 제어할 수 있는 기능이 있다고 말합니다.

**메모리 영역**  
운영체제는 다음과 같이 메모리 영역을 할당하여 프로그램을 관리합니다.

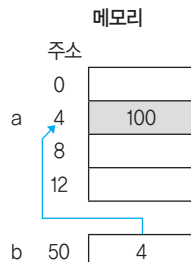
- 코드 영역 : 실행할 프로그램의 코드가 저장됨
- 데이터 영역 : 전역 변수와 정적 변수가 저장됨
- 힙 영역 : 필요에 의해 동적으로 할당되는 영역임
- 스택 영역 : 함수의 매개 변수와 지역 변수가 저장됨

## 1 포인터와 포인터 변수

포인터는 변수의 주소를 말하며, C언어에서는 주소를 제어할 수 있는 기능을 제공합니다.

- C언어에서 변수의 주소를 저장할 때 사용하는 변수를 포인터 변수라 한다.
- 포인터 변수를 선언할 때는 자료의 형을 먼저 쓰고 변수명 앞에 간접 연산자 \*를 붙인다(예 `int *a`).
- 포인터 변수에 주소를 저장하기 위해 변수의 주소를 알아낼 때는 변수 앞에 번지 연산자 &를 붙인다(예 `a = &b`).
- 실행문에서 포인터 변수에 간접 연산자 \*를 붙이면 해당 포인터 변수가 가리키는 곳의 값을 말한다(예 `c = *a`).
- 포인터 변수는 필요에 의해 동적으로 할당되는 메모리 영역\*인 힙 영역에 접근하는 동적 변수이다.
- 포인터 변수의 용도
  - 연결된 자료 구조를 구성하기 위해 사용한다.
  - 동적으로 할당된 자료 구조를 지정하기 위해 사용한다.
  - 배열을 인수로 전달하기 위해 사용한다.
  - 문자열을 표현하기 위해 사용한다.
  - 커다란 배열에서 요소를 효율적으로 저장하기 위해 사용한다.
  - 메모리에 직접 접근하기 위해 사용한다.

예를 들어, a 변수에 100을 저장시키고, a 변수의 주소를 포인터 변수 b에 기억시켰다면 다음 그림과 같이 표현하고 말할 수 있다.



- a는 메모리의 4번지에 대한 이름이다.
- a 변수의 주소는 4다.
- a 변수에는 100이 기억되어 있다.
- 4번지에는 100이 기억되어 있다.
- &a는 a 변수의 주소를 말한다. 즉 &a는 4다.
- 포인터 변수 b는 a 변수의 주소를 기억하고 있다.
- 포인터 변수가 가리키는 곳의 값을 말할 때는 \*을 붙인다.
- \*b는 b에 저장된 주소가 가리키는 곳에 저장된 값을 말하므로 100이다.



**예제 1** 다음 C언어로 구현된 프로그램의 출력 결과를 확인하시오.

```
main()
{
    int a = 50; ①
    int *b; ②
    b = &a; ③
    *b = *b+20; ④

    printf("%d, %d", a, *b); ⑤
}
```

정수형 변수 a를 선언하고 50으로 초기화한다.

정수형 변수가 저장된 곳의 주소를 기억할 포인터 변수 b를 선언한다.

정수형 변수 a의 주소를 포인터 변수 b에 기억시킨다. b에는 a의 주소가 저장된다.

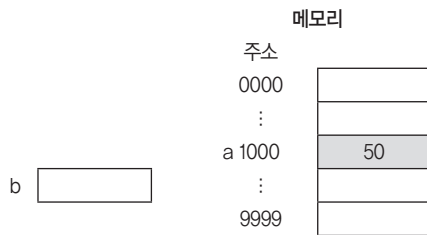
b가 가리키는 곳의 값에 20을 더한다. b가 가리키는 곳이 a이므로 결국 a의 값도 바뀌는 것이다.

결과 70, 70

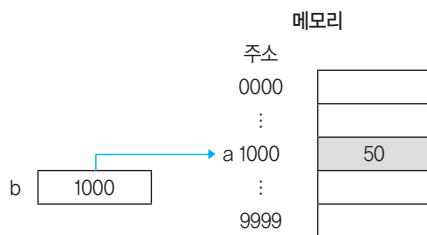
- ②와 같이 선언할 때 \*는 해당 변수가 포인터 변수라는 것을 의미한다.
- ④, ⑤와 같이 사용할 때 \*를 붙이면 그 포인터 변수가 가리키는 곳의 값을 의미한다.

위 코드의 실행 과정에 따라 메모리의 변화를 그려보면 다음과 같다.

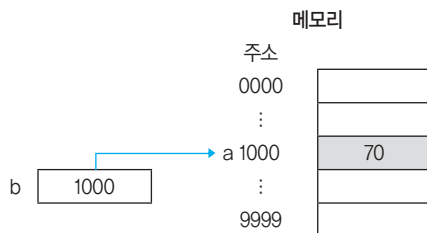
- ①, ②번 수행 : 주기억장치의 빈 공간 어딘가에 a라는 이름을 붙이고 그 곳에 50을 저장한다.



- ③번 수행 : 변수 a의 주소가 b에 기억된다는 것은 b가 변수 a의 주소를 가리키고 있다는 의미이다.



- ④번 수행 : b가 가리키는 곳의 값에 20을 더해 다시 b가 가리키는 곳에 저장한다. 그곳은 변수 a의 주소이므로 변수 a의 값도 저절로 변경되는 것이다.



#### 전문가의 조언

포인터 변수는 주로 문자열 처리나 함수 간의 자료 처리, 배열의 요소 지정 등에 활용됩니다. 여기서 다루는 예제에서는 포인터 변수에 주소를 저장하고 포인터 변수를 이용해 주소에 접근하는 기본적인 개념만 이해하고 넘어가세요.



#### 전문가의 조언

여기서 지정한 주소는 임의로 정한 것이며, 이해를 돕기 위해 주소를 실제 표현되는 16진수가 아니라 10진수로 표현했습니다.



#### 전문가의 조언

##### 포인터 표기 방법

a에 저장된 값은 정수형 배열의 시작 주소입니다. a의 값을 1 증가 시킨다는 것은 현재 a가 가리키고 있는 정수형 자료의 주소에서 다음 정수형 자료의 주소로 가리키는 주소를 증가시킨다는 것입니다. 정수형 자료의 크기는 4바이트이므로 다음 물리적 메모리의 주소는 4Byte 증가한 곳을 가리키는 것입니다. **예제**를 통해 배열의 시작 주소에서 1번씩, 즉 4Byte씩 증가시키는 것을 그림으로 확인해 보세요.

**예제 2** 다음 C언어로 구현된 프로그램의 출력 결과를 확인하시오.

```
main( )
{
    int a = 3, *b; ①    정수형 변수 a와 정수형 포인터 변수 b를 선언하고 a를 3으로 초기화한다.
    b = &a; ②          정수형 변수 a의 주소를 포인터 변수 b에 기억시킨다. b에는 a의 주소가 저장된다.
    printf("%d", ++*b); ③    포인터 변수 b가 가리키는 곳의 값(3)을 1증가(++ )시킨 후 출력한다.
}
```

결과 4

### 3 포인터와 배열

배열을 포인터 변수에 저장한 후 포인터를 이용해 배열의 요소에 접근할 수 있다.

- 배열 위치를 나타내는 첨자를 생략하고 배열의 대표명만 지정하면 배열의 첫 번째 요소의 주소를 지정하는 것과 같다.
- 배열 요소에 대한 주소를 지정할 때는 일반 변수와 동일하게 & 연산자를 사용한다.

**예** int a[5], \*b;

b = a → 배열의 대표명을 적었으므로 a 배열의 시작 주소인 a[0]의 주소를 b에 저장한다.

b = &a[0] → a 배열의 첫 번째 요소인 a[0]의 주소(&)를 b에 저장한다.

	a[0]	a[1]	a[2]	a[3]	a[4]	← 배열 표기 방법
배열 a	첫 번째	두 번째	세 번째	네 번째	다섯 번째	
	*(a+0)	*(a+1)	*(a+2)	*(a+3)	*(a+4)	← 포인터 표기 방법

- 배열의 요소가 포인터인 포인터형 배열을 선언할 수 있다.

**예제** 다음의 출력 결과를 확인하시오.

```
main( )
{
    int a[5];          5개의 요소를 갖는 정수형 배열 a를 선언한다. 선언할 때 사용할 개수를 선언하고, 사용할 때는 첨자를 0부터 사용한다.

    int i;             정수형 변수 i를 선언한다.
    int *p; ①          반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ②번을 반복 수행한다.
    for (i = 0; i < 5; i++)

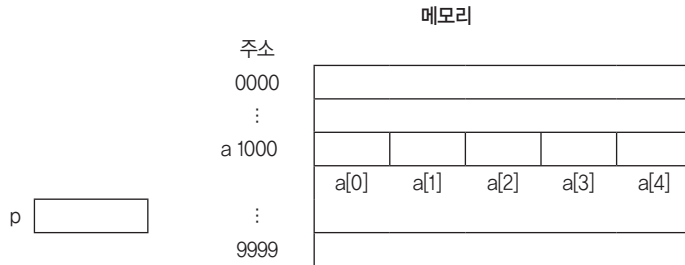
        a[i] = i + 10; ②

    p = a; ③           반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ④번을 반복하여 수행한다.
    for (i = 0; i < 5; i++)

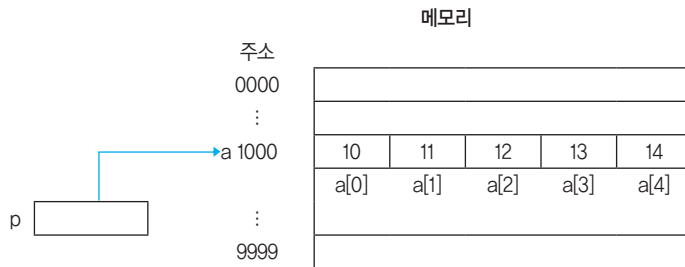
        printf("%d ", *(p+i)); ④    결과 10 11 12 13 14
}
```

코드의 실행 과정에 따라 메모리의 변화를 그려보면 다음과 같다.

- ❶ 정수형 변수가 저장된 곳의 주소를 기억할 정수형 포인터 변수  $p$ 를 선언한다.



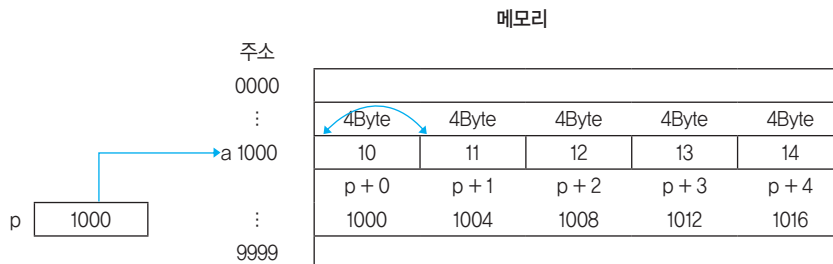
- ❷ 배열  $a$ 의  $i$ 번째에  $i+10$ 을 저장한다.  $i$ 는 0~4까지 변하므로 배열  $a$ 에 저장된 값은 다음과 같다.



- ❸ 배열명  $a$ 는 배열의 주소이므로 포인터 변수  $p$ 에는 배열  $a$ 의 시작 위치가 기억된다. 배열의 이름은 주소이므로 ' $p = \&a$ '처럼 입력하지 않도록 주의해야 한다.



- ❹  $p$ 에 저장된 값은 정수형 배열의 시작 주소이다.  $p$ 의 값을 1 증가시킨다는 것은 현재  $p$ 가 가리키고 있는 정수형 자료의 주소에서 다음 정수형 자료의 주소로 가리키는 주소를 증가시킨다는 것이다. 정수형 자료의 크기는 4바이트이므로 다음 물리적 메모리의 주소는 4Byte 증가한 곳을 가리키는 것이다.  $p$ 에 저장된 배열의 시작 주소에서 1번지씩, 즉 4Byte씩 증가시키는 것을 그림으로 표현하면 다음과 같다.



#### 전문가의 조언

포인터의 증가가 헷갈리면 정수형 자료의 크기가 4Byte이기 때문에 포인터를 1 증가시키면 물리적인 주소는 4Byte가 증가한다고 기억해 두세요. 만약  $p$ 가 문자 배열의 주소를 가지고 있다면 문자형 자료의 크기는 1Byte이므로 포인터를 1 증가시킬 때 물리적인 메모리의 주소도 1Byte 증가합니다.

- **p+0** : 배열의 시작 주소에 0을 더했으므로, 배열의 시작 주소인 '1000' 번지 그대로이다.
- **\*(p+0)** : '1000' 번지의 값은 10이다. 10을 출력한다.
- **p+1** : '1000'에서 한 번지 증가한 주소는 '1004' 번지이다.
- **\*(p+1)** : '1004' 번지의 값은 11이다. 11을 출력한다.
- **p+2** : '1000'에서 두 번지 증가한 주소는 '1008' 번지이다.
- **\*(p+2)** : '1008' 번지의 값은 12이다. 12를 출력한다.
- ⋮

**문제** 다음 프로그램의 출력 결과를 적으시오.

번호	코드	결과
①	<pre>int a = 5, b, *c; c = &amp;a; b = ++*c; printf("%d", b);</pre>	
②	<pre>int a = 10, *b; b = &amp;a; for (int i = 0; i &lt; 5; i++)     *b += i; printf("%d", *b);</pre>	
③	<pre>int a = 31, b, *c, *d; c = &amp;a; d = &amp;b; *d = --*c % 3 ? a + a : a * a; printf("%d", *d);</pre>	
④	<pre>int a = 5, b = 7, c, *d; d = &amp;c; *d = a &amp; b; printf("%d", c);</pre>	

② 반복문 실행에 따른 변수 i, a, \*b의 변화

i	a	*b
	10	10
0	10	10
1	11	11
2	13	13
3	16	16
4	20	20
5		

①	<pre>int a = 5, b, *c; c = &amp;a; b = ++*c; printf("%d", b)</pre>	<p>정수형 변수 a의 주소를 포인터 변수 b에 기억시킵니다. b에는 a의 주소가 저장됩니다.</p> <p>포인터 변수 c가 가리키는 값(5)을 1증가(++) 시킨 값 6을 b에 저장합니다.</p>
②	<pre>int a = 10, *b; b = &amp;a; for (int i = 0; i &lt; 5; i++)     *b += i; printf("%d", *b);</pre>	<p>정수형 변수 a의 주소를 포인터 변수 b에 기억시킵니다. b에는 a의 주소가 저장됩니다.</p> <p>반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ①번을 반복하여 수행합니다.</p> <p>*b = *b + i; ① *b = *b + i'와 같습니다. 포인터 변수 b가 가리키는 곳의 값에 i를 누적합니다.</p> <p>포인터 변수 b가 가리키는 곳의 값을 출력합니다.</p>

③

```
int a = 31, b, *c, *d;
```

```
c = &a;
```

정수형 변수 a의 주소를 포인터 변수 c에 기억시킵니다. c에는 a의 주소가 저장됩니다.

```
d = &b;
```

정수형 변수 b의 주소를 포인터 변수 d에 기억시킵니다. d에는 b의 주소가 저장됩니다.

```
*d = --*c % 3 ? a + a : a * a;
```

포인터 변수 c가 가리키는 값(31)을 1감소(--)시킨 30을 3으로 나눈 나머지가 0이고, 0은 조건에서 거짓을 의미하므로 'a\*a'를 수행합니다. a는 초기에 31이 저장되었지만 '--\*c'에 의해 c가 가리키는 값, 즉 a의 값이 30으로 변경되었으므로 'a\*a'는 900이 됩니다. 이 값을 d가 가리키는 곳에 저장합니다.

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

포인터 변수 d가 가리키는 곳의 값을 출력합니다.

④

```
int a = 5, b = 7, c, *d;
```

```
d = &c;
```

정수형 변수 c의 주소를 포인터 변수 d에 기억시킵니다. d에는 c의 주소가 저장됩니다.

```
*d = a & b;
```

a(5=0101) & b(7=0111)의 값 5(0101)를 d가 가리키는 곳에 저장합니다. 그곳은 변수 c의 주소이므로 변수 c의 값도 5로 변경됩니다.

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

변수 c의 값을 출력합니다.

**결과** ① 6 ② 20 ③ 900 ④ 5



#### 전문가의 조언

**a & b**

&(비트 and)는 두 비트가 모두 1일 때만 1이 되는 비트 연산자입니다.

7 = ... 0000 0101

5 = ... 0000 0111

& ... 0000 0101(5)



#### 기출문제 따라잡기

Section 132

이전기출

1. C언어의 포인터 형(Pointer Type)에 대한 설명으로 틀린 것은?

- ① 포인터 변수는 기억장소의 번지를 기억하는 동적 변수이다.
- ② 포인터는 가리키는 자료형이 일치할 때 대입하는 규칙이 있다.
- ③ 보통 변수의 번지를 참조하려면 번지 연산자 #을 변수 앞에 쓴다.
- ④ 실행문에서는 간접 연산자 \*를 사용하여 포인터 변수가 지시하고 있는 내용을 참조한다.

C언어의 포인터 형(Pointer Type)에서 변수의 번지를 참조하려면 번지 연산자 &를 변수 앞에 씁니다.

출제예상

2. 포인터 자료형에 대한 설명으로 옳지 않은 것은?

- ① 고급 언어에서는 사용되지 않고 저급 언어에서 주로 사용되는 기법이다.
- ② 객체를 참조하기 위해 주소를 값으로 하는 형식이다.
- ③ 커다란 배열에 원소를 효율적으로 저장하고자 할 때 이용한다.
- ④ 하나의 자료에 동시에 많은 리스트의 연결이 가능하다.

우리가 지금 학습하고 있는 C언어는 고급 언어란 것을 염두에 두고 풀어보세요.

▶ 정답 : 1. ③ 2. ①



## 전문가의 조언

절차적 프로그래밍 언어는 실행 순서를 중시한다는 것을 중심으로 개념과 장·단점을 파악해 두세요.

### 프로그래밍 언어

프로그래밍 언어는 컴퓨터를 이용해 특정 문제를 해결하기 위한 프로그램을 작성하기 위해 사용되는 언어를 말합니다.

## 전문가의 조언

제시된 언어의 특징을 보고 어떤 언어를 말하는지 구분할 수 있도록 학습하세요. 특히 C언어는 눈여겨 보세요.

### 저급 언어와 고급 언어

저급 언어와 고급 언어의 구분은 언어가 저급이나 고급인지를 말하는 것이 아니라 기계 친화적이나 인간 친화적이나, 즉 기계가 이해하기 쉬우면 저급 언어, 인간이 이해하기 쉬우면 고급 언어입니다.

### 컴파일러(Compiler)

FORTAN, COBOL, C, ALGOL 등의 고급 언어로 작성된 프로그램을 기계어로 번역하는 프로그램입니다.

## 1 절차적 프로그래밍 언어\*의 개요

절차적 프로그래밍 언어는 일련의 처리 절차를 정해진 문법에 따라 순서대로 기술해 나가는 언어이다.

- 절차적 프로그래밍 언어는 프로그램이 실행되는 절차를 중요시 한다.
- 절차적 프로그래밍 언어는 데이터를 중심으로 프로시저를 구현하며, 프로그램 전체가 유기적으로 연결되어 있다.
- 절차적 프로그래밍 언어는 자연어에 가까운 단어와 문장으로 구성된다.
- 절차적 프로그래밍 언어는 과학 계산이나 하드웨어 제어에 주로 사용된다.

## 2 절차적 프로그래밍 언어의 장·단점

- 컴퓨터의 처리 구조와 유사하여 실행 속도가 빠르다.
- 같은 코드를 복사하지 않고 다른 위치에서 호출하여 사용할 수 있다.
- 모듈 구성이 용이하며, 구조적인 프로그래밍이 가능하다.
- 프로그램을 분석하기 어렵다.
- 유지 보수나 코드의 수정이 어렵다.

## 3 절차적 프로그래밍 언어의 종류

언어	특징
C	<ul style="list-style-type: none"> <li>• 1972년 미국 벨 연구소의 데니스 리치에 의해 개발되었다.</li> <li>• 시스템 소프트웨어를 개발하기 편리하여 시스템 프로그래밍 언어로 널리 사용된다.</li> <li>• 자료의 주소를 조작할 수 있는 포인터를 제공한다.</li> <li>• 고급 프로그래밍 언어*이면서 저급 프로그램 언어*의 특징을 모두 갖췄다.</li> <li>• UNIX의 일부가 C 언어로 구현되었다.</li> <li>• 컴파일러* 방식의 언어이다.</li> <li>• 이식성이 좋아 컴퓨터 기종에 관계없이 프로그램을 작성할 수 있다.</li> </ul>
ALGOL	<ul style="list-style-type: none"> <li>• 수치 계산이나 논리 연산을 위한 과학 기술 계산용 언어이다.</li> <li>• PASCAL과 C 언어의 모체가 되었다.</li> </ul>
COBOL	<ul style="list-style-type: none"> <li>• 사무 처리용 언어이다.</li> <li>• 영어 문장 형식으로 구성되어 있어 이해와 사용이 쉽다.</li> <li>• 4개의 DIVISION으로 구성되어 있다.</li> </ul>
FORTAN	<ul style="list-style-type: none"> <li>• 과학 기술 계산용 언어이다.</li> <li>• 수학과 공학 분야의 공식이나 수식과 같은 형태로 프로그래밍 할 수 있다.</li> </ul>



## 기출문제 따라잡기

Section 133

출제예상

1. 다음 중 절차적 프로그래밍 언어에 대한 설명으로 틀린 것은?

- ① 컴퓨터의 처리 구조와 유사하여 실행 속도가 빠르다.
- ② 실행되는 절차를 중시한다.
- ③ 데이터를 중심으로 프로시저를 구현한다.
- ④ 상속을 통한 재사용성이 높다.

상속을 통한 재사용성이 높은 것은 객체지향 프로그래밍 언어입니다. 객체지향 프로그래밍 언어는 다음 섹션에서 자세히 공부합니다.

이전기출

2. C언어에 대한 설명으로 옳지 않은 것은?

- ① 다양한 연산자를 제공한다.
- ② 이식성이 높은 언어이다.
- ③ 시스템 프로그래밍이 용이하다.
- ④ 기계어에 해당한다.

C언어는 기계어가 아니고 기계어로 번역해야 실행할 수 있는 컴파일러 방식의 언어입니다.

이전기출

3. 시스템 프로그래밍에 가장 적합한 언어는?

- ① C
- ② COBOL
- ③ Fortran
- ④ Pascal

시스템 프로그래밍 하면 C! 꼭 기억하세요.

▶ 정답 : 1. ④ 2. ④ 3. ①



## 전문가의 조언

객체지향 프로그래밍 언어가 무엇인지 의미를 정확히 알아야 하며, 그 의미를 중심으로 객체지향 프로그래밍의 장·단점을 정리해 두세요.

## 전문가의 조언

객체지향 프로그래밍의 각 구성 요소의 의미를 정확하게 구분해서 알아두세요.

**GUI(Graphical User Interface)**  
GUI는 아이콘이나 메뉴를 마우스로 선택하여 작업을 수행하는 그래픽 환경의 인터페이스입니다.

## 1 객체지향 프로그래밍 언어의 개요

객체지향 프로그래밍 언어는 현실 세계의 개체(Entity)를 기계의 부품처럼 하나의 객체로 만들어, 기계적인 부품들을 조립하여 제품을 만들 듯이 소프트웨어를 개발할 때도 객체들을 조립해서 프로그램을 작성할 수 있도록 한 프로그래밍 기법이다.

- 프로시저보다는 명령과 데이터로 구성된 객체를 중심으로 하는 프로그래밍 기법으로, 한 프로그램을 다른 프로그램에서 이용할 수 있도록 한다.

## 2 객체지향 프로그래밍 언어의 장·단점

- 상속을 통한 재사용과 시스템의 확장이 용이하다.
- 코드의 재활용성이 높다.
- 자연적인 모델링에 의해 분석과 설계를 쉽고 효율적으로 할 수 있다.
- 사용자와 개발자 사이의 이해를 쉽게 해준다.
- 대형 프로그램의 작성이 용이하다.
- 소프트웨어 개발 및 유지보수가 용이하다.
- 프로그래밍 구현을 지원해 주는 정형화된 분석 및 설계 방법이 없다.
- 구현 시 처리 시간이 지연된다.

## 3 객체지향 프로그래밍 언어의 종류

언어	특징
JAVA	<ul style="list-style-type: none"> <li>• 분산 네트워크 환경에 적용이 가능하며, 멀티스레드 기능을 제공하므로 여러 작업을 동시에 처리할 수 있다.</li> <li>• 운영체제 및 하드웨어에 독립적이며, 이식성이 강하다.</li> <li>• 캡슐화가 가능하고 재사용성 높다.</li> </ul>
C++	<ul style="list-style-type: none"> <li>• C 언어에 객체지향 개념을 적용한 언어이다.</li> <li>• 모든 문제를 객체로 모델링하여 표현한다.</li> </ul>
Smalltalk	<ul style="list-style-type: none"> <li>• 1세대 객체지향 프로그래밍 언어 중 하나로 순수한 객체지향 프로그래밍 언어이다.</li> <li>• 최초로 GUI*를 제공한 언어이다.</li> </ul>

## 4 객체지향 프로그래밍 언어의 구성 요소

객체지향 프로그래밍 언어의 구성 요소에는 객체(Object), 클래스(Class), 메시지(Message)가 있다.



객체(Object)	<ul style="list-style-type: none"> <li>• 데이터(속성)와 이를 처리하기 위한 연산(메소드)을 결합시킨 실체이다.</li> <li>• 데이터 구조와 그 위에서 수행되는 연산들을 가지고 있는 소프트웨어 모듈이다.</li> <li>• 속성(Attribute) : 한 클래스 내에 속한 객체들이 가지고 있는 데이터 값들을 단위 별로 정의하는 것으로서 성질, 분류, 식별, 수량 또는 현재 상태 등을 표현한다.</li> <li>• 메소드(Method) : 객체가 메시지를 받아 실행해야 할 때 구체적인 연산을 정의하는 것으로, 객체의 상태를 참조하거나 변경하는 수단이 된다.</li> </ul>
클래스(Class)	<ul style="list-style-type: none"> <li>• 두 개 이상의 유사한 객체들을 묶어서 하나의 공통된 특성을 표현하는 요소이다. 즉 공통된 특성과 행위를 갖는 객체의 집합이라고 할 수 있다.</li> <li>• 객체의 유형 또는 타입(Object Type)을 의미한다.</li> </ul>
메시지(Message)	<ul style="list-style-type: none"> <li>• 객체들 간에 상호작용을 하는데 사용되는 수단으로 객체의 메소드(동작, 연산)를 일으키는 외부의 요구 사항이다.</li> <li>• 메시지를 받은 객체는 대응하는 연산을 수행하여 예상된 결과를 반환하게 된다.</li> </ul>

## 5 객체지향 프로그래밍 언어의 특징

객체지향 프로그래밍 언어의 특징에는 캡슐화, 정보 은닉, 추상화, 상속성, 다형성 등이 있다.

캡슐화 (Encapsulation)*	<ul style="list-style-type: none"> <li>• 데이터(속성)와 데이터를 처리하는 함수를 하나로 묶는 것을 의미한다.</li> <li>• 캡슐화된 객체의 세부 내용이 외부에 은폐(정보 은닉)되어, 변경이 발생할 때 오류의 파급 효과가 적다.</li> <li>• 캡슐화된 객체들은 재사용이 용이하다.</li> </ul>
정보 은닉 (Information Hiding)*	캡슐화에서 가장 중요한 개념으로, 다른 객체에게 자신의 정보를 숨기고 자신의 연산만을 통하여 접근을 허용하는 것이다.
추상화 (Abstraction)*	<ul style="list-style-type: none"> <li>• 불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에만 중점을 두어 개략화하는 것, 즉 모델화하는 것이다.</li> <li>• 데이터의 공통된 성질을 추출하여 슈퍼 클래스를 선정하는 개념이다.</li> </ul>
상속성 (Inheritance)*	<ul style="list-style-type: none"> <li>• 이미 정의된 상위 클래스(부모 클래스)의 모든 속성과 연산을 하위 클래스가 물려받는 것이다.</li> <li>• 상속성을 이용하면 하위 클래스는 상위 클래스의 모든 속성과 연산을 자신의 클래스 내에서 다시 정의하지 않고서도 즉시 자신의 속성으로 사용할 수 있다.</li> </ul>
다형성 (Polymorphism)	<ul style="list-style-type: none"> <li>• 메시지에 의해 객체(클래스)가 연산을 수행하게 될 때 하나의 메시지에 대해 각 객체(클래스)가 가지고 있는 고유한 방법(특성)으로 응답할 수 있는 능력을 의미한다.</li> <li>• 객체(클래스)들은 동일한 메소드명을 사용하며 같은 의미의 응답을 한다.</li> </ul>



### 전문가의 조언

캡슐화와 상속성, 추상화를 중심으로 각각의 의미를 확실히 알아두세요.

### 캡슐화와 정보 은닉의 장점

- 유지보수의 용이성
- 객체 이용의 용이성

### 정보 은닉

캡슐로 된 감기약을 예로 들면 정보 은닉은 감기약에 어떤 재료가 들어 있는지 몰라도 감기가 걸렸을 때 먹는 약이라는 것만 알고 복용하는 것과 같은 의미입니다.

### 추상화(Abstraction)의 종류

- **과정 추상화** : 자세한 수행 과정을 정의하지 않고, 전반적인 흐름만 파악할 수 있게 설계하는 방법
- **데이터 추상화** : 데이터의 세부적인 속성이나 용도를 정의하지 않고, 데이터 구조를 대표할 수 있는 표현으로 대체하는 방법
- **제어 추상화** : 이벤트 발생의 정확한 절차나 방법을 정의하지 않고, 대표할 수 있는 표현으로 대체하는 방법

### 상속성(Inheritance)의 종류

- **단일 상속** : 하나의 상위 클래스로부터 상속받는 것
- **다중 상속** : 여러 개의 상위 클래스로부터 상속받는 것

이전기출

1. 객체지향의 기본 개념 중 객체가 메시지를 받아 실행해야 할 객체의 구체적인 연산을 정의한 것은?

- ① 메소드                      ② 추상화  
③ 상속성                    ④ 캡슐화

연산, 동작, 함수 하면 메소드! 속성, 변수, 자료 구조 하면 데이터! 반드시 기억해 두세요.

출제예상

2. 객체지향 분석에서 불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에만 중점을 두어 개략화시킨 것을 무엇이라고 하는가?

- ① 상속성                      ② 클래스  
③ 추상화                    ④ 메시지

가장 중요한 것만 개략화시킨 것! 추상화죠!

이전기출

3. 하나 이상의 유사한 객체들을 묶어서 하나의 공통된 특성을 표현한 것으로 데이터 추상화의 개념으로 볼 수 있는 것은?

- ① 객체(Object)                      ② 클래스(Class)  
③ 실체(Instance)                  ④ 메시지(Message)

클래스를 짧게 표현하자면 ‘공통된 특성과 행위를 갖는 객체의 집합’이라고 할 수 있습니다. 용어에 대한 문제가 참 많이 나오죠! 용어의 의미를 확실히 파악해야 합니다.

이전기출

4. 객체지향 개념에서 이미 정의되어 있는 상위 클래스(슈퍼 클래스 혹은 부모 클래스)의 메소드를 비롯한 모든 속성을 하위 클래스가 물려 받는 것을 무엇이라고 하는가?

- ① Abstraction
  - ② Method
  - ③ Inheritance
  - ④ Message

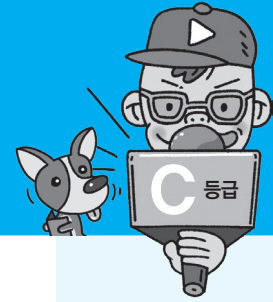
문제가 길어 어려워 보이지만 간단합니다. Inheritance(상속)은 상위 클래스의 속성을 하위 클래스가 물려받는 것이지요.

이전기출

5. 객체지향 기법에 대한 설명으로 거리가 먼 것은?

- ① 프로시저에 근간을 두고 프로그래밍을 구현하는 기법이다.
- ② 현실 세계를 모형화하여 사용자와 개발자가 쉽게 이해할 수 있다.
- ③ 소프트웨어의 재사용율이 높아진다.
- ④ 소프트웨어의 유지보수성이 향상된다.

객체지향 기법은 객체를 만들어 객체를 근간으로 프로그램을 구현합니다.



## 1 스크립트 언어(Script Language)의 개요

스크립트 언어는 HTML 문서 안에 직접 프로그래밍 언어를 삽입하여 사용하는 것으로, 기계어로 컴파일 되지 않고 별도의 번역기가 소스를 분석하여 동작하게 하는 언어이다.

- 게시판 입력, 상품 검색, 회원 가입 등과 같은 데이터베이스 처리 작업을 수행하기 위해 주로 사용한다.
- 스크립트 언어는 클라이언트의 웹 브라우저에서 해석되어 실행되는 클라이언트용 스크립트 언어와 서버에서 해석되어 실행된 후 결과만 클라이언트로 보내는 서버용 스크립트 언어가 있다.
  - 서버용 스크립트 언어 : ASP, JSP, PHP, 파이썬
  - 클라이언트용 스크립트 언어 : 자바 스크립트(JAVA Script)

## 2 스크립트 언어의 장 · 단점

- 컴파일 없이 바로 실행하므로 결과를 바로 확인할 수 있다.
- 배우고 코딩하기 쉽다.
- 개발 시간이 짧다.
- 소스 코드를 쉽고 빠르게 수정할 수 있다.
- 코드를 읽고 해석해야 하므로 실행 속도가 느리다.
- 런타임 오류가 많이 발생한다.

## 3 스크립트 언어의 종류

자바 스크립트 (JAVA Script)	<ul style="list-style-type: none"> <li>• 웹 페이지의 동작을 제어하는 데 사용되는 클라이언트용 스크립트 언어로, 클래스가 존재하지 않으며 변수 선언도 필요 없다.</li> <li>• 서버에서 데이터를 전송할 때 아이디, 비밀번호, 수량 등의 입력 사항을 확인하기 위한 용도로 많이 사용된다.</li> </ul>
ASP(Active Server Page)	<ul style="list-style-type: none"> <li>• 서버 측에서 동적으로 수행되는 페이지를 만들기 위한 언어로 마이크로 소프트사에서 제작하였다.</li> <li>• Windows 계열에서만 수행 가능한 프로그래밍 언어이다.</li> </ul>
JSP(Java Server Page)	JAVA로 만들어진 서버용 스크립트로, 다양한 운영체제에서 사용이 가능하다.



### 전문가의 조언

스크립트 언어의 개념을 파악하고, 종류는 서버용과 클라이언트용으로 구분하여 기억하세요. 그리고 각 스크립트 언어들의 개별적인 특징은 서로를 구분할 수 있을 정도로만 정리해 두세요.

### 인터프리터 언어

인터프리터 언어는 원시 프로그램을 줄 단위로 번역하여 바로 실행해 주는 언어로, 목적 프로그램을 생성하지 않고 즉시 실행 결과를 출력합니다.

PHP(Professional Hypertext Preprocessor)

- 서버용 스크립트 언어로, Linux, Unix, Windows 운영체제에서 사용 가능하다.
- C, Java 등과 문법이 유사하므로 배우기 쉬워 웹 페이지 제작에 많이 사용된다.

파이썬(Python)

객체지향 기능을 지원하는 대화형 인터프리터 언어\*로, 플랫폼에 독립적이고 문법이 간단하여 배우기 쉽다.



### 기출문제 따라잡기

Section 135

출제예상

1. 다음 중 자바를 이용한 서버측 스크립트이며 다양한 운영체제에서 사용 가능한 웹 프로그래밍 언어는?

- ① ASP
- ② JSP
- ③ PHP
- ④ DHTML

보기에 제시된 언어들이 무엇에 대한 약자인지 생각해 보고, 자바와 서버가 들어간 언어를 찾아보세요.

출제예상

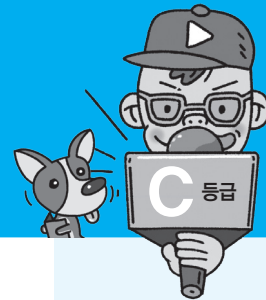
2. 다음 중 아래와 같은 특성을 갖는 웹 프로그래밍 언어로 옳은 것은?

- 클래스가 존재하지 않으며 변수 선언도 필요 없다.
- 소스코드가 HTML 문서에 포함되어 있다.
- 사용자의 웹 브라우저에서 직접 번역되고 실행된다.

- ① CGI
- ② XML
- ③ ASP
- ④ Java Script

사용자의 웹 브라우저에서 직접 번역되고 실행되는 것은 클라이언트용 스크립트 언어입니다. 보기 중에서 클라이언트용 스크립트 언어를 찾아보세요.

▶ 정답 : 1. ② 2. ④



## 1 선언형 언어

선언형 언어는 명령형 언어와 반대되는 개념의 언어로, 명령형 언어가 문제를 해결하기 위한 방법을 기술한다면 선언형 언어는 프로그램이 수행해야 문제를 기술하는 언어이다.

- 선언형 언어는 목표를 명시하고 알고리즘은 명시하지 않는다.\*
- 선언형 언어에는 함수형 언어와 논리형 언어 등이 있다.\*

함수형 언어	<ul style="list-style-type: none"> <li>• 수학적 함수를 조합하여 문제를 해결하는 언어로, 알려진 값을 함수에 적용하는 것을 기반으로 한다.</li> <li>• 적용형 언어라고도 한다.</li> <li>• 재귀호출이 자주 이용된다.</li> <li>• 병렬 처리에 유리하다.</li> <li>• 종류 : LISP</li> </ul>
논리형 언어	<ul style="list-style-type: none"> <li>• 기호 논리학에 기반을 둔 언어로, 논리 문장을 이용하여 프로그램을 표현하고 계산을 수행한다.</li> <li>• 선언적 언어라고도 한다.</li> <li>• 반복문이나 선택문을 사용하지 않는다.</li> <li>• 비절차적 언어이다.</li> <li>• 종류 : PROLOG</li> </ul>

잠깐만요



### 명령형 언어

명령형 언어는 순차적인 명령 수행을 기본으로 하는 언어로, 문제를 처리하기 위한 방법에 초점을 두고 코드를 작성합니다.

- 폰노이만 구조에 개념적인 기초를 두고 있습니다.
- 명령형 언어는 알고리즘을 명시하고 목표는 명시하지 않습니다.
- 특정 구문의 연산을 이용하여 상태를 변경시키고 프로그램을 동작시킵니다.
- 개체의 동작과 상태를 중요시 합니다.
- 명령형 언어에는 절차적 언어와 객체지향 언어가 있습니다.
- 종류 : FORTRAN, COBOL, C, JAVA 등

## 2 선언형 언어의 장·단점

- 가독성이나 재사용성이 좋다.
- 작동 순서를 구체적으로 작성하지 않기 때문에 오류가 적다.
- 프로그램 동작을 변경하지 않고도 관련 값을 대체할 수 있다.



### 전문가의 조언

선언형 언어와 명령형 언어의 차이점을 구분하여 알아두고, 선언형 언어의 종류를 기억해 두세요.

### 목표를 명시하고 알고리즘은 명시하지 않는다.

대표적인 선언형 언어로 HTML이 있는데, HTML은 웹페이지를 구성할 때 제목, 글꼴, 그림 등 웹 페이지에 표시할 것들을 묘사하지만 화면에 어떤 방법으로 표시할지는 묘사하지 않습니다.

### 함수형 언어와 논리형 언어

선언형 언어에는 함수형 언어나 논리형 언어 외에 다른 언어들도 포함됩니다. HTML의 경우는 완전한 선언형 언어지만 함수형 언어나 논리형 언어에 포함되지는 않습니다.

### 3 선언형 프로그래밍 언어 종류

HTML	인터넷의 표준 문서인 하이퍼텍스트 문서를 만들기 위해 사용하는 언어로, 특별한 데이터 타입이 없는 단순한 텍스트이므로 호환성이 좋고 사용이 편리하다.
LISP	<ul style="list-style-type: none"> <li>• 인공지능 분야에 사용되는 언어이다.</li> <li>• 기본 자료 구조가 연결 리스트 구조이며, 재귀(Recursion) 호출을 많이 사용한다.</li> </ul>
PROLOG	논리학을 기초로 한 고급 언어로, 인공 지능 분야에서의 논리적인 추론이나 리스트 처리 등에 주로 사용된다.
XML	<ul style="list-style-type: none"> <li>• 기존 HTML의 단점을 보완하여 웹에서 구조화된 폭넓고 다양한 문서들을 상호 교환할 수 있도록 설계된 언어이다.</li> <li>• HTML에 사용자가 새로운 태그(Tag)를 정의할 수 있으며, 문서의 내용과 이를 표현하는 방식이 독립적이다.</li> </ul>
Haskell	<ul style="list-style-type: none"> <li>• 함수형 프로그래밍 언어로 부작용(Side Effect)이 없다.</li> <li>• 코드가 간결하고 에러 발생 가능성이 낮다.</li> </ul>



#### 기출문제 따라잡기

Section 136

출제예상

#### 1. 다음 중 언어의 분류에 대한 설명으로 틀린 것은?

- ① 객체들이 모여서 하나의 프로그램이 되는 것은 객체 지향 언어이다.
- ② 폰노이만 구조에 개념적 기초를 둔 언어는 명령형 언어이다.
- ③ 명령문을 순서대로 나열한 것은 선언형 언어이다.
- ④ 수학적 함수를 조합하여 문제를 해결하는 언어는 함수형 언어이다.

명령문을 순서대로 나열하는 것은 명령형 언어입니다.

출제예상

#### 2. 다음 중 프로그래밍 언어와 해당 언어에 해당하는 프로그램의 연결이 옳지 않은 것은?

- ① 명령형 언어 - HTML
- ② 함수형 언어 - LISP
- ③ 논리 언어 - PROLOG
- ④ 객체지향 언어 - JAVA

HTML은 선언형 언어입니다.

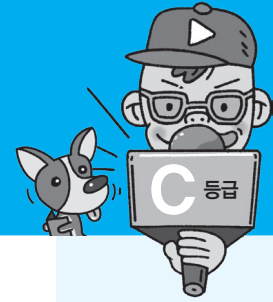
출제예상

#### 3. 프로그래밍 언어의 전형에 해당하지 않는 것은?

- ① 선언형 언어
- ② 함수형 언어
- ③ 명령형 언어
- ④ 추상형 언어

지금까지 공부하면서 한 번도 못봤던 것을 찾아보세요.

▶ 정답 : 1. ③ 2. ① 3. ④



## 1 라이브러리의 개념

라이브러리는 프로그램을 효율적으로 개발할 수 있도록 자주 사용하는 함수나 데이터들을 미리 만들어 모아 놓은 집합체이다.

- 자주 사용하는 함수들의 반복적인 코드 작성을 피하기 위해 미리 만들어 놓은 것으로, 필요할 때는 언제든지 호출하여 사용할 수 있다.
- 라이브러리에는 표준 라이브러리와 외부 라이브러리가 있다.
- **표준 라이브러리** : 프로그래밍 언어에 기본적으로 포함되어 있는 라이브러리로, 여러 종류의 모듈\*이나 패키지\* 형태이다.
- **외부 라이브러리** : 개발자들이 필요한 기능들을 만들어 인터넷 등에 공유해 놓은 것으로, 외부 라이브러리를 다운받아 설치한 후 사용한다.

## 2 C언어의 대표적인 표준 라이브러리

C언어는 라이브러리를 헤더 파일로 제공하는데, 각 헤더 파일에는 응용 프로그램 개발에 필요한 함수들이 정리되어 있다.

- C언어에서 헤더 파일을 사용하려면 '#include <stdio.h>'와 같이 include문을 이용해 선언한 후 사용해야 한다.

헤더 파일	기능
stdio.h	<ul style="list-style-type: none"> <li>• 데이터의 입·출력에 사용되는 기능들을 제공한다.</li> <li>• 주요 함수 : printf, scanf, fprintf, fscanf, fclose, fopen 등</li> </ul>
math.h	<ul style="list-style-type: none"> <li>• 수학 함수들을 제공한다.</li> <li>• 주요 함수 : sqrt, pow, abs 등</li> </ul>
string.h	<ul style="list-style-type: none"> <li>• 문자열 처리에 사용되는 기능들을 제공한다.</li> <li>• 주요 함수 : strlen, strcpy, strcmp 등</li> </ul>
stdlib.h	<ul style="list-style-type: none"> <li>• 자료형 변환, 난수 발생, 메모리 할당에 사용되는 기능들을 제공한다.</li> <li>• 주요 함수 : atoi, atof, srand, rand, malloc, free 등</li> </ul>
time.h	<ul style="list-style-type: none"> <li>• 시간 처리에 사용되는 기능들을 제공한다.</li> <li>• 주요 함수 : time, clock 등</li> </ul>

## 3 JAVA의 대표적인 표준 라이브러리

JAVA는 라이브러리를 패키지에 포함하여 제공하는데, 각 패키지에는 JAVA 응용 프로그램 개발에 필요한 메소드\*들이 클래스로 정리되어 있다.

### 전문가의 조언

C언어는 라이브러리를 헤더 파일로 제공하고 JAVA는 패키지로 제공한다. 이것을 염두에 두고 대표적인 표준 라이브러리들의 종류와 개별적인 기능을 정리해 두세요.

- **모듈** : 하나의 기능이 한 개의 파일로 구현된 형태
- **패키지** : 하나의 패키지 폴더 안에 여러 개의 모듈을 모아 놓은 형태

### 메소드(Method)

JAVA에서는 특정 기능을 수행하는 함수를 메소드라고 합니다.

- JAVA에서 패키지를 사용하려면 'import java.util'과 같이 import문을 이용해 선언한 후 사용해야 한다.
- import로 선언된 패키지 안에 있는 클래스의 메소드를 사용할 때는 클래스와 메소드를 마침표(.)로 구분하여 'Math.abs()'와 같이 사용한다.

패키지	기능
java.lang	<ul style="list-style-type: none"> <li>• 자바에 기본적으로 필요한 인터페이스, 자료형, 예외 처리 등에 관련된 기능을 제공한다.</li> <li>• import문 없이도 사용할 수 있다.</li> <li>• 주요 클래스 : String, System, Process, Runtime, Math, Error 등</li> </ul>
java.util	<ul style="list-style-type: none"> <li>• 날짜 처리, 난수 발생, 복잡한 문자열 처리 등에 관련된 기능을 제공한다.</li> <li>• 주요 클래스 : Date, Calendar, Random, StringTokenizer 등</li> </ul>
java.io	<ul style="list-style-type: none"> <li>• 파일 입·출력과 관련된 기능 및 프로토콜을 제공한다.</li> <li>• 주요 클래스 : InputStream, OutputStream, Reader, Writer 등</li> </ul>
java.net	<ul style="list-style-type: none"> <li>• 네트워크와 관련된 기능을 제공한다.</li> <li>• 주요 클래스 : Socket, URL, InetAddress 등</li> </ul>
java.awt	<ul style="list-style-type: none"> <li>• 사용자 인터페이스(UI)와 관련된 기능을 제공한다.</li> <li>• 주요 클래스 : Frame, Panel, Dialog, Button, Checkbox 등</li> </ul>



## 기출문제 따라잡기

Section 137

출제예상

### 1. 다음 중 라이브러리에 대한 설명으로 잘못된 것은?

- ① 라이브러리는 프로그램을 효율적으로 개발할 수 있도록 자주 사용하는 함수나 데이터들을 미리 만들어 모아 놓은 집합체이다.
- ② 자주 사용하는 함수들의 반복적인 코드 작성을 피하기 위해 미리 만들어 놓은 것으로, 필요할 때마다 호출하여 사용할 수 있다.
- ③ 라이브러리에는 표준 라이브러리와 외부 라이브러리가 있다.
- ④ 표준 라이브러리는 개발자들이 필요한 기능들을 만들어 인터넷 등에 공유해 놓은 것으로, 필요한 라이브러리는 다운받아 설치한 후 사용할 수 있다.

다운받아 설치한 후 사용해야 하는 라이브러리라면 외부에 있는 라이브러리겠네요.

출제예상

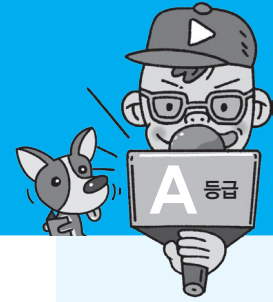
### 2. C언어의 대표적인 표준 라이브러리에 대한 설명이 잘못된 것은?

- ① stdio.h : 데이터의 입·출력에 사용되는 기능들을 제공
- ② string.h : 문자열 처리에 사용되는 기능들을 제공
- ③ stdlib.h : 수학 함수들을 제공
- ④ time.h : 시간 처리에 사용되는 기능들을 제공

수학(Mathematics)과 관련된 라이브러리의 명칭은 무엇일까요?

▶ 정답 : 1. ④ 2. ③





## 1 C언어의 표준 입 · 출력 함수의 개요

표준 입출력 함수(Input-Output Functions)란 키보드로 입력받아 화면으로 출력할 때 사용하는 함수로, 대표적으로 `scanf()`, `getchar()`, `gets()`, `printf()`, `putchar()`, `puts()` 등이 있다.

## 2 scanf() 함수

`scanf()` 함수는 C언어의 표준 입력 함수로, 키보드로 입력받아 변수에 저장하는 함수이다.

### 형식

`scanf(서식 문자열, 변수의 주소)`

- 서식 문자열 : 입력받을 데이터의 자료형을 지정한다.
- 변수의 주소 : 데이터를 입력받을 변수를 적는다. 변수의 주소로 입력 받아야 하기 때문에 변수에 주소연산자 &를 붙인다.

예 `scanf("%3d", &a);`

- ▶ % : 서식 문자열을 지정
- ▶ 3 : 입력 자릿수를 3자리로 지정
- ▶ d : 10진수로 입력
- ▶ &a : 입력받은 데이터를 변수 a의 주소에 저장

### 특징

- 입력받을 데이터의 자료형, 자릿수 등을 지정할 수 있다.
  - 한 번에 여러 개의 데이터를 입력 받을 수 있다.
  - 서식 문자열과 변수의 자료형은 일치해야 한다.
- 예 `scanf("%d %f", &i, &j);` → '%d'와 i, '%f'와 j는 자료형이 일치해야 한다.

### 서식 문자열

서식 문자열은 `printf()` 함수로 출력할 때도 동일하게 적용된다.

서식 문자열	의미
%d	정수형 10진수를 입 · 출력하기 위해 지정한다.
%u	부호없는 정수형 10진수를 입 · 출력하기 위해 지정한다.
%o	정수형 8진수를 입 · 출력하기 위해 지정한다.



#### 전문가의 조언

데이터 표준 입 · 출력 함수들의 기능과 형식을 기억하세요. 특히 데이터를 입력하거나 출력할 때 지정하는 서식 문자열의 지정 방법을 확실히 기억하고 넘어가세요.



#### 전문가의 조언

먼저 %d, %c, %s, %f만 기억해 두세요. 나머지는 다시 나올 때 그때 기억하면 됩니다. d는 decimal(10진수)의 약자, c는 character(문자)의 약자, s는 string(문자열)의 약자, f는 float(실수)의 약자라는 것을 알면 훨씬 쉽게 기억됩니다.

%x	정수형 16진수를 입 · 출력하기 위해 지정한다.
%c	문자를 입 · 출력하기 위해 지정한다.
%s	문자열을 입 · 출력하기 위해 지정한다.
%f	소수점을 포함하는 실수를 입 · 출력하기 위해 지정한다.
%e	지수형 실수를 입 · 출력하기 위해 지정한다.
%ld	long형 10진수를 입 · 출력하기 위해 지정한다.
%lo	long형 8진수를 입 · 출력하기 위해 지정한다.
%lx	long형 16진수를 입 · 출력하기 위해 지정한다.
%p	주소를 16진수로 입 · 출력하기 위해 지정한다.



## JAVA에서의 표준 입력

JAVA에서 키보드로 입력받은 값을 변수에 저장하려면 먼저 Scanner 클래스를 이용해 키보드로부터 값을 입력받는 객체 변수를 생성한 후 이를 사용해야 합니다.

### 형식

```
① Scanner scan01 = new Scanner(System.in);
② inNum = scan01.nextInt( );
```

#### ① 객체 변수 생성

- Scanner : 입력에 사용할 객체 변수를 생성할 때 사용하는 클래스 이름입니다. 그대로 적어줍니다.
- scan01 : 객체 변수명입니다. 사용자 임의로 적어줍니다.
- new : 객체 생성 예약어입니다. 그대로 적어줍니다.
- Scanner( ) : 클래스의 이름입니다. ( )를 붙여 그대로 적어줍니다.
- System.in : 표준 입력장치, 즉 키보드를 의미합니다. 키보드로부터 값을 입력받는 객체 변수를 생성할 것이므로 그대로 적어줍니다.

#### ② 객체 변수 활용

- inNum : 입력받은 값을 저장할 변수입니다. 이 변수는 미리 선언되어 있어야 합니다.
- scan01.nextInt( )
  - scan01 : 입력에 사용할 객체 변수 이름입니다. 객체 변수 생성 시 사용한 객체 변수 이름과 동일해야 합니다.
  - nextInt( ) : 입력받은 값을 정수형으로 반환\*합니다.

### Scanner 클래스의 입력 메소드

- next( ) : 입력값을 문자열로 반환
- nextLine( ) : 입력받은 라인 전체를 문자열로 반환
- nextInt( ) : 입력값을 정수형으로 반환
- nextFloat( ) : 입력값을 실수형으로 반환



### 전문가의 조언

배열명은 배열의 시작 주소를 의미하므로 배열명 앞에는 &를 붙이지 않아도 됩니다.

**문제** scanf( ) 함수를 이용하여 다음과 같이 데이터를 입력할 경우 변수에 기억되는 결과를 쓰시오.

번호	코드	입력 데이터	결과
①	scanf("%d", &i);	20	
②	scanf("%2d", &i);	125	
③	scanf("%4f", &j);	12,123	
④	scanf("%c", &a);	SINAGONG	
⑤	char b[8]; scanf("%4c", b);*	SINAGONG	

⑥	char b[8]; scanf("%s", b);	GIL BUT	
⑦	char b[8], c[8]; scanf("%s %2s", b, c);	GIL BUT	
⑧	char b[8]; scanf("%d %f %s", &i, &j, b);	345 2.62E-6 LOVE	
⑨	char b[8], c[8]; scanf("%c %5c", b, c);	LOVE ME	
⑩	scanf("%3d %5f", &i, &j);	123456789	
⑪	scanf("%3d\$\$\$%3f", &i, &j);	123\$\$\$456789	

- ① 정수형 10진수로 저장됩니다.  
 ② 앞에 2자리까지만 저장됩니다.  
 ③ 소수점을 포함하여 앞에 4자리까지만 저장됩니다.  
 ④ 입력한 데이터 중 앞에 1문자만 저장됩니다.  
 ⑤ 입력한 데이터 중 앞에 4자리까지만 저장됩니다.  
 ⑥ 입력한 데이터 중 빈 칸(공백)이 있으면 빈 칸 앞까지만 저장됩니다.  
 ⑦ 배열 b에는 입력한 데이터 중 빈 칸 앞까지만 저장되고 배열 c에는 입력한 데이터 중 빈 칸 이후 2자리까지만 저장됩니다.  
 ⑧ 입력한 데이터가 빈 칸으로 구분되어 i에는 정수형 10진수로, j에는 소수점을 포함하는 실수형으로, 배열 b에는 문자열로 저장됩니다.  
 ⑨ 배열 b에는 입력한 데이터 중 앞의 문자 1자리만 저장되고 배열 c에는 입력한 데이터 중 배열 b에 저장된 문자 1자리 이후 5자리까지만 저장됩니다.  
 ⑩ i에는 입력한 데이터 중 앞의 3자리까지만 저장되고 j에는 i에 저장된 3자리 이후 소수점을 포함하여 5자리까지만 저장되는데, 입력한 데이터가 정수이므로 5자리가 정수로 저장되고 기본적으로 소수점 이하 6자리가 0으로 저장됩니다.  
 ⑪ 입력한 데이터에 '\$\$\$'가 없으면 정상적으로 입력되지 않습니다. i에는 입력한 데이터 중 '\$\$\$'를 기준으로 앞에 3자리까지만 저장되고 j에는 입력한 데이터 중 '\$\$\$'를 기준으로 뒤에 3자리까지가 소수점을 포함하여 저장되는데, 입력한 데이터가 정수이므로 3자리가 정수로 저장되고 기본적으로 소수점 이하 6자리가 0으로 저장됩니다.

**결과** ① 20 ② 12 ③ 12.1 ④ S ⑤ SINA ⑥ GIL ⑦ b : GIL, c : BU  
 ⑧ i : 345, j : 2.62E-6, b : LOVE ⑨ b : L, c : OVE M ⑩ i : 123, j : 45678.000000  
 ⑪ i : 123, j : 456.000000

### 3 printf( ) 함수

printf( ) 함수는 C언어의 표준 출력 함수로, 인수로 주어진 값을 화면에 출력하는 함수이다.

형식

printf(서식 문자열, 변수)

- 서식 문자열 : 변수의 자료형에 맞는 서식 문자열을 입력한다.
- 변수 : 서식 문자열의 순서에 맞게 출력할 변수를 적는다. scanf( )와 달리 주소 연산자 &를 붙이지 않는다.

예 printf("%-8.2f", 200.2); → 200.20 ∨ ∨ (∨는 빈 칸을 의미함)

- ▶ % : 서식 문자임을 지정
- ▶ - : 왼쪽부터 출력
- ▶ 8 : 출력 자릿수를 8자리로 지정
- ▶ 2 : 소수점 이하를 2자리로 지정
- ▶ f : 실수로 출력

## 주요 제어문자

제어문자란 입력 혹은 출력 내용을 제어하는 문자이다.

문자	의미	기능
\n	new line	커서를 다음 줄 앞으로 이동한다.
\b	backspace	커서를 왼쪽으로 한 칸 이동한다.
\t	tab	커서를 일정 간격 띄운다.
\r	carriage return	커서를 현재 줄의 처음으로 이동한다.
\0	null	널 문자를 출력한다.
\'	single quote	작은따옴표를 출력한다.
\"	double quote	큰따옴표를 출력한다.
\a	alert	스피커로 벨 소리를 출력한다.
\\	backslash	역 슬래시를 출력한다.
\f	form feed	한 페이지를 넘긴다.

예 printf("%d\n", a); → a의 값을 정수형 10진수로 출력한 후 다음 줄로 이동한다.



## JAVA에서의 표준 출력

JAVA에서 값을 화면에 출력할 때는 printf( ) 메소드를 이용합니다.

### 형식

System.out.printf(서식 문자열, 변수)

- System.out.printf( ) : System 클래스의 서브 클래스인 out 클래스의 메소드 printf( )를 사용해서 출력한다는 의미입니다. printf( ) 메소드는 C언어의 printf( ) 함수와 사용법이 동일합니다.

예 System.out.printf("%-8.2f", 200.2); → 200.20 ∨ ∨ (∨는 빈 칸을 의미함)

- ▶ % : 서식 문자임을 지정
- ▶ - : 왼쪽부터 출력
- ▶ 8 : 출력 자릿수를 8자리로 지정
- ▶ 2 : 소수점 이하를 2자리로 지정
- ▶ f : 실수로 출력

**문제 1** printf( ) 함수를 이용하여 다음과 같이 데이터를 출력할 경우 결과를 쓰시오. (√는 빈칸을 의미함)

번호	코드	결과
①	printf("%d", 2543);	
②	printf("%3d", 2543);	
③	printf("%6d", 2543);	
④	printf("%-6d", 2543);	
⑤	printf("%06d", 2543);	
⑥	printf("%f", 245.2555);	
⑦	printf("%.3f", 245.2555);	
⑧	printf("%.8,2f", 245.2555);	
⑨	printf("%e", 25.43);	
⑩	printf("%.3s", "help me");	
⑪	printf("%3s", "help me");	
⑫	printf("%.8,6s", "help me");	
⑬	printf("%-8,6s", "help me");	
⑭	printf("250은 10진수로 %d\t 8진수로 %o\n", 250, 250);	
⑮	printf("a=%8,2f\t b=%e\n", 125.23f, 3141.592e-1);	
⑯	printf("\A\는 문자로 %c, 아스키코드로 %d\n", 'A', 'A');	

- ① 정수형으로 출력합니다.
- ② 전체 3자리를 확보한 후 오른쪽부터 출력하는데, 출력할 값이 지정한 자릿수보다 큰 경우에는 자릿수를 무시하고 모두 출력합니다.
- ③ 전체 6자리를 확보한 후 오른쪽부터 출력합니다.
- ④ 전체 6자리를 확보한 후 왼쪽부터 출력합니다.
- ⑤ 전체 6자리를 확보한 후 오른쪽부터 출력하되 왼쪽의 공백은 0으로 채워 출력합니다.
- ⑥ 자릿수가 지정되지 않았으므로 정수 부분은 모두 출력하고 소수점 이하는 기본적으로 6자리로 출력됩니다.
- ⑦ 정수 부분은 모두 출력하고 소수점 이하는 4자리에서 반올림하여 3자리까지만 출력합니다.
- ⑧ 전체 8자리를 확보한 후 소수점과 소수점 이하 2자리를 출력하고 남은 5자리에 정수 부분을 출력합니다.
- ⑨ 25.43을 정수 부분이 한 자리만 남도록 정규화하여 출력합니다.
- ⑩ 왼쪽을 기준으로 3글자만 출력합니다.
- ⑪ 전체 3자리를 확보한 후 출력하는데, 출력할 값이 지정한 자릿수보다 큰 경우에는 자릿수를 무시하고 모두 출력합니다.
- ⑫ 전체 8자리를 확보한 후 오른쪽부터 6글자만 출력합니다.
- ⑬ 전체 8자리를 확보한 후 왼쪽부터 6글자만 출력합니다.
- ⑭ "250은 10진수로 "를 그대로 출력하고 서식 문자열 '%d'에 대응하는 정수 값 250을 10진수로 출력하고 제어문자 '\t'로 인해 4칸을 뒀 다음 서식 문자열의 공백만큼 한 칸을 뒀습니다. 이어서 "8진수로 "를 출력하고 서식 문자열 '%o'에 대응하는 정수 값 250을 8진수로 출력합니다. '\n'으로 인해 커서는 다음 줄로 이동합니다.
- ⑮ "a="을 그대로 출력하고 서식 문자열 '%8,2f'에 대응하는 실수 값 125.23을 전체 8자리를 확보하여 오른쪽부터 소수점과 소수점 이하 2자리를 출력하고 남은 5자리에 정수 125를 출력합니다. 그리고 제어문자 '\t'로 인해 4칸을 뒀 다음 서식 문자열의 공백만큼 한 칸을 뒀습니다. 이어서 "b="을 출력하고 서식 문자열 '%e'에 대응하는 지수 값 3141.592e-1을 소수점 이상 한 자리만 표시하는 지수 형태로 출력합니다. '\n'으로 인해 커서는 다음 줄로 이동합니다.



#### 전문가의 조언

25.43을 정규화하면 정수 부분이 한 자리만 남도록 소수점의 위치를 조절한 후 소수점이 이동한 자리수 만큼을 e뒤에 표현합니다. 25.43을 정규화하면 2.543인데 기본적으로 소수점 자리는 6자리로 표현하므로 2.543000이며 이는 25.43에서 소수점 자리가 왼쪽으로 한 자리 이동하였으므로 2.543000e+01로 표현합니다.



#### 전문가의 조언

3141.592e-1은 3141.592×10<sup>-1</sup>을 의미합니다. 실수가 실수형 변수에 저장될 때는 정규화 과정을 거쳐 가수부를 한 자리만 남기므로 3.141592e+02가 저장됩니다.

⑩ “\A\”는 문자로 “를 그대로 출력하되 제어문자 ‘\’으로 인해 “A”를 작은따옴표로 묶어 출력합니다. 이어서 서식 문자열 ‘%c’에 대응하는 문자 “A”를 출력합니다. 그리고 콤마(,)를 출력한 다음 서식 문자열의 공백만큼 한 칸을 뚫니다. 이어서 “아스키코드로 “을 출력하고 서식 문자열 ‘%d’에 대응하는 문자 ‘A’에 해당하는 아스키코드 값을 정수형으로 출력합니다. ‘\n’으로 인해 커서는 다음 줄로 이동합니다.

**결과** ① 2543 ② 2543 ③ √ √ 2543 ④ 2543√ √ ⑤ 002543 ⑥ 245,255500 ⑦ 245,256  
 ⑧ √ √ 245,26 ⑨ 2,543000e+01 ⑩ hel ⑪ help me ⑫ √ √ help m ⑬ help m√ √  
 ⑭ 250은 10진수로 250 8진수로 372 ⑮ a=√ √ 125,23 b=3.141592e+02  
 ⑯ ‘A’는 문자로 A, 아스키코드로 65



### 출력 데이터가 여러 개인 경우

`printf("250은 10진수로 %d\t 8진수로 %o\n", 250, 250);` → 250은 10진수로 250 8진수로 372

**문제 2** 다음과 같이 `scanf()` 함수로 값을 입력받아 `printf()` 함수로 출력할 경우 결과를 쓰시오. (√는 빈칸을 의미함)

번호	코드	입력 데이터	코드	결과
①	<code>char a[5]; scanf("%d %e %s", &amp;i, &amp;j, a);</code>	5468 3.483E-2 GOOD	<code>printf("%4d %f %2s", i, j, a);</code>	
②	<code>scanf("%e", &amp;i);</code>	123.45E-1	<code>printf("%f\t %e\n", i, i);</code>	
③	<code>scanf("%d", &amp;i);</code>	300	<code>printf("[%5d], [%-5d], [%05d]", i, i, i);</code>	
④	<code>scanf("%2d \n \t %3d", &amp;i, &amp;j);</code>	12345678	<code>printf("i=%d j=%d\n", i, j);</code>	



### 전문가의 조언

지수형으로 저장된 3.483E-2를 실수형으로 변환하게 되면 소수점 자리수를 -2, 즉 왼쪽으로 2자리 이동한 0.03483이 됩니다.



### 전문가의 조언

지수형으로 저장된 123.45E-1을 실수형으로 변환하게 되면 소수점 자리수를 -1, 즉 왼쪽으로 1자리 이동한 12.345가 됩니다.

- `scanf("%d %e %s", &i, &j, a);`  
입력한 데이터가 빈 칸으로 구분되어 i에는 정수형 10진수로, j에는 지수형으로, 배열 a에는 문자열로 저장됩니다.
  - `printf("%4d %f %2s", i, j, a);`  
- %4d : 전체 4자리를 확보한 후 오른쪽부터 출력  
- %f : 실수형으로 출력하되 자릿수를 지정하지 않았으므로 정수 부분은 모두 출력하고 소수점 이하는 기본적으로 6자리로 출력  
- %2s : 전체 2자리를 확보한 후 출력하는데, 출력할 값이 지정한 자릿수보다 크므로 자릿수를 무시하고 모두 출력
- `scanf("%e", &i);`  
입력한 데이터가 지수형으로 저장됩니다.
  - `printf("%f\t %e\n", i, i);`  
- %f : 실수형으로 출력하되 자릿수를 지정하지 않았으므로 정수 부분은 모두 출력하고 소수점 이하는 기본적으로 6자리로 출력  
- %e : 지수형으로 출력하되 자릿수를 지정하지 않았으므로 정수 부분은 한 자리만 표시하고 소수점 이하는 기본적으로 6자리로 출력
- `scanf("%d", &i);`  
입력한 데이터가 정수형으로 저장됩니다.
  - `printf("[%5d], [%-5d], [%05d]", i, i, i);`  
- [%5d] : “l”를 그대로 출력하고 전체 5자리를 확보한 후 오른쪽부터 출력한 후 “)”를 그대로 출력

- [%-5d]: "["를 그대로 출력하고 전체 5자리를 확보한 후 왼쪽부터 출력한 후 "]"를 그대로 출력
- [%05d]: "["를 그대로 출력하고 전체 5자리를 확보한 후 오른쪽부터 출력하되 왼쪽의 공백은 0으로 채워 출력한 후 "]"를 그대로 출력

④ • `scanf("%2d \n \t %3d", &i, &j);`

- i에는 입력한 데이터 중 앞의 2자리까지만 저장되고 j에는 i에 저장된 2자리 다음 3자리까지만 저장됩니다.
- 입력에서 제어문자 '\n \t'는 무시됩니다.

• `printf("i=%d j=%d\n", i, j);`

- "i="을 그대로 출력하고 서식 문자열 '%d'에 대응하는 i의 값 12를 출력한 다음 서식 문자열의 공백만큼 한 칸을 땁니다. 이어서 "j="을 출력하고 서식 문자열 '%d'에 대응하는 j의 값 345를 출력합니다. '\n'으로 인해 커서는 다음 줄로 이동합니다.

**결과** ① 5468 0.034830 GOOD ② 12.345000 1.234500e+01 ③ [v v 300], [300 v v], [00300]  
④ i=12 j=345

## 4 기타 표준 입·출력 함수

입력	<code>getchar()</code>	키보드로 한 문자를 입력받아 변수에 저장하는 함수
	<code>gets()</code>	키보드로 문자열을 입력받아 변수에 저장하는 함수로, <b>Enter</b> 를 누르기 전까지를 하나의 문자열로 인식하여 저장함
출력	<code>putchar()</code>	인수로 주어진 한 문자를 화면에 출력하는 함수
	<code>puts()</code>	인수로 주어진 문자열을 화면에 출력한 후 커서를 자동으로 다음 줄 앞으로 이동하는 함수

**문제** 다음의 코드를 실행하여 데이터를 입력할 경우 출력되는 결과를 쓰시오.

번호	코드	입력 데이터	코드	결과
①	<code>a = getchar();</code>	GIL BUT	<code>putchar(a);</code>	
②	<code>putchar('G');</code>			
③	<code>putchar('G'+1);</code>			
④	<code>gets(b);</code>	GIL BUT	<code>puts(b);</code>	
⑤	<code>puts("GIL BUT");</code>			

① • `a = getchar();`

- 입력한 데이터 중 한 문자가 a에 저장됩니다.

• `putchar(a);`

- a에 저장된 한 문자를 출력합니다.

② 한 문자를 출력할 때는 문자를 작은따옴표(')로 묶어줍니다.

③ 문자 'G'에 해당하는 아스키코드 값 71에 1을 더한 값 72에 해당하는 문자 "H"를 출력합니다.

④ • `gets(b);`

- 입력한 데이터 전체가 b에 저장됩니다.

• `puts(b);`

- b에 저장된 문자열 전체를 출력합니다.

⑤ 큰따옴표(")로 묶인 문자열 전체를 출력합니다.

**결과** ① G ② G ③ H ④ GIL BUT ⑤ GIL BUT



## 기출문제 따라잡기

Section 138

이전기출

1. 다음 중 C언어에서 문자열을 출력하기 위해 사용되는 것은?

- ① %x
- ② %d
- ③ %s
- ④ %h

서식 문자열의 영문자가 무엇의 약자인지 알면 쉽게 맞힐 수 있는 문제네요.  
string(문자열)을 출력하기 위해서는 무엇을 사용해야 할까요?

이전기출

2. 다음 중 C언어의 함수 중 한 문자 입력 함수로 알맞은 것은?

- ① puts()
- ② putchar()
- ③ gets()
- ④ getchar()

한 문자(character)를 입력 받기(get) 위한 함수는 getchar, 문자열(string)을 입력 받기(get) 위한 함수는 gets입니다.

이전기출

3. 다음 C언어 프로그램을 실행한 결과 출력되는 값은?

```
#include<stdio.h>
void main (void)
{
    int a;
    a = 7;
    printf("%d", a+a);
}
```

- ① 11
- ② 12
- ③ 13
- ④ 14

문제의 코드를 하나씩 살펴보면 아래와 같습니다.

```
#include<stdio.h>
void main (void)
{
    int a;           a를 정수형 변수로 선언합니다.
    a = 7;           a에 7을 저장합니다.
    printf("%d", a+a); 'a+a'를 실행하면 14이고, 이것을 정수형 10진수로
                    출력하면 14가 그대로 출력됩니다.
}
```

이전기출

4. 다음은 C언어 프로그램이다. 실행 결과는? (단, √는 빈 칸을 의미한다.)

```
main()
{
    char *a;
    a = "MOUNTAIN";
    printf("%8.5s\n", a);
}
```

- ① √ √ √ MOUNT
- ② MOUNT √ √ √
- ③ MOUNTAIN
- ④ NTAIN

'%8.5s'는 전체 8자리를 확보하여 a 변수의 값 중 5자리(MOUNT)만 출력합니다.  
데이터는 기본적으로 오른쪽을 기준으로 출력됩니다.

※ 'char \*a'는 포인터 변수를 선언한 것으로, 포인터 변수에는 문자열을 저장할 수 있습니다.

이전기출

5. 다음은 C언어에서 입·출력 함수에 쓰이는 서식 문자열이다. 이 중에서 성격이 다른 하나는?

- ① %d
- ② %x
- ③ %o
- ④ %f

하나는 실수형 서식 문자열이고, 나머지는 모두 정수형 서식 문자열입니다.

이전기출

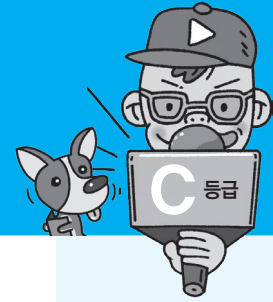
6. C언어에서 사용하는 제어문자에 대한 의미가 옳지 않은 것은?

- ① \n : new page
- ② \r : carriage return
- ③ \b : backspace
- ④ \t : tab

\n은 'new line'을 의미합니다.

▶ 정답 : 1. ③ 2. ④ 3. ④ 4. ① 5. ④ 6. ①





## 1 예외 처리의 개요

프로그램의 정상적인 실행을 방해하는 조건이나 상태를 예외(Exception)라고 하며, 이러한 예외가 발생했을 때 프로그래머가 해당 문제에 대비해 작성해 놓은 처리 루틴을 수행하도록 하는 것을 예외 처리(Exception Handling)라고 한다.

- 예외가 발생했을 때 일반적인 처리 루틴은 프로그램을 종료시키거나 로그를 남기도록 하는 것이다.
- C++, Ada, JAVA, 자바스크립트와 같은 언어에는 예외 처리 기능이 내장되어 있으며, 그 외의 언어에서는 필요한 경우 조건문을 이용해 예외 처리 루틴을 작성한다.
- 예외의 원인에는 컴퓨터 하드웨어 문제, 운영체제의 설정 실수, 라이브러리 손상, 사용자의 입력 실수, 받아들이지 못하는 연산, 할당하지 못하는 기억장치 접근 등 다양하다.

## 2 JAVA의 예외 처리

JAVA는 예외를 객체로 취급하며, 예외와 관련된 클래스를 java.lang 패키지에서 제공한다.

- JAVA에서는 try ~ catch 문을 이용해 예외를 처리한다.
- try 블록 코드를 수행하다 예외가 발생하면 예외를 처리하는 catch 블록으로 이동하여 예외 처리 코드를 수행하므로 예외가 발생한 이후의 코드는 실행되지 않는다.
- catch 블록에서 선언한 변수는 해당 catch 블록에서만 유효하다.
- try ~ catch 문 안에 또 다른 try ~ catch 문을 포함할 수 있다.
- try ~ catch 문 안에서는 실행 코드가 한 줄이라도 중괄호({})를 생략할 수 없다.

### 기본 형식

```
try {
    예외가 발생할 가능성이 있는 코드;
}
catch ( 예외객체1 매개변수 ) {
    예외객체1에 해당하는 예외 발생 시 처리 코드;
}
catch ( 예외객체2 매개변수 ) {
    예외객체2에 해당하는 예외 발생 시 처리 코드;
}
catch ( 예외객체n 매개변수 ) {
    예외객체n에 해당하는 예외 발생 시 처리 코드;
}
```



### 전문가의 조언

예외 처리의 개념, 그리고 JAVA에서의 예외 처리 방법과 JAVA의 주요 예외 처리 객체들의 개별적인 예외 발생 원인을 파악해 두세요.



#### 전문가의 조언

일반적으로 예외가 발생한 경우에는 'try문 → 해당 예외 catch문 → finally문' 순으로 진행되며, 예외가 발생하지 않은 경우에는 'try문 → finally문' 순으로 진행됩니다. finally 문은 예외 발생과 관계없이 무조건 수행되는 블록으로 생략이 가능합니다.

```
catch (Exception 매개변수) {
    예외객체1~n에 해당하지 않는 예외 발생 시 처리 코드;
}
finally {
    예외의 발생 여부와 관계없이 무조건 처리되는 코드;
}
```

### 3 JAVA의 주요 예외 객체

예외 객체	발생 원인
ClassNotFoundException	클래스를 찾지 못한 경우
NoSuchMethodException	메소드를 찾지 못한 경우
FileNotFoundException	파일을 찾지 못한 경우
InterruptedException	입출력 처리가 중단된 경우
ArithmeticException	0으로 나누는 등의 산술 연산에 대한 예외가 발생한 경우
IllegalArgumentException	잘못된 인자를 전달한 경우
NumberFormatException	숫자 형식으로 변환할 수 없는 문자열을 숫자 형식으로 변환한 경우
ArrayIndexOutOfBoundsException	배열의 범위를 벗어난 접근을 시도한 경우
NegativeArraySizeException	0보다 작은 값으로 배열의 크기를 지정한 경우
NullPointerException	존재하지 않는 객체를 참조한 경우



#### 기출문제 따라잡기

Section 139

이전기출

##### 1. 다음 중 예외 처리에 대한 설명으로 잘못된 것은?

- ① 프로그램의 정상적인 실행을 방해하는 조건이나 상태를 예외(Exception)라고 한다.
- ② 예외가 발생했을 때 프로그래머가 해당 문제에 대비해 작성해 놓은 처리 루틴을 수행하도록 하는 것을 예외 처리(Exception Handling)라고 한다.
- ③ 예외가 발생했을 때 일반적인 처리 루틴은 프로그램을 종료시키거나 로그를 남기도록 하는 것이다.
- ④ 예외 처리 기능은 프로그래밍 언어 자체에서는 지원하지 않으므로 프로그래머가 조건문을 이용해 예외 처리를 작성한다.

앞서 JAVA에서의 예외 처리 방법인 try ~ catch문을 학습했는데, 프로그래머가 조건문을 이용해 코딩하는 것이었나요?

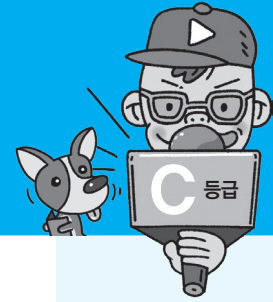
이전기출

##### 2. JAVA에서의 예외 처리에 대한 설명으로 잘못된 것은?

- ① JAVA는 예외를 객체로 취급하며, 예외와 관련된 클래스를 java.lang 패키지로 제공한다.
- ② JAVA에서는 try ~ catch 문을 이용해 예외를 처리한다.
- ③ try 블록에서 예외 발생 여부를 확인한 후 예외가 발생하면, catch 블록의 예외 처리 코드를 수행한다.
- ④ try 블록에서 예외가 발생하면 해당 예외 처리 catch 문으로 이동하여 예외 처리를 수행한 후 다시 돌아와 예외가 발생한 이후 try 블록의 코드를 실행한다.

try 블록 코드를 수행하다 예외가 발생하면 예외를 처리하는 catch 블록으로 이동하여 예외 처리 코드를 수행한 후 바로 finally 블록으로 이동하므로 수행 중이던 try 블록의 코드는 더 이상 수행하지 않습니다.

▶ 정답 : 1. ④ 2. ④



## 1 프로토타입(Prototype)의 개념

프로그래밍 언어에서 프로토타입이란 함수 원형(Function Prototype)이라는 의미로, 컴파일러에게 사용될 함수에 대한 정보를 미리 알리는 것이다.

- 함수가 호출되기 전에 함수가 미리 정의되는 경우에는 프로토타입을 정의하지 않아도 된다.
- 프로토타입은 본문이 없다는 점을 제외하고 함수 정의와 형태가 동일하다.
- 프로토타입에 정의된 반환 형식은 함수 정의에 지정된 반환 형식과 반드시 일치해야 한다.

## 2 C언어에서의 프로토타입 선언

C언어에서 프로토타입은 main() 함수 바깥쪽에 선언해야 한다.

### • 기본 형식

```
int func(int i, int j);
```

- int : 반환될 값의 자료형을 적는다. 반환될 값이 없으면 void를 적는다.
- func : 사용할 함수의 이름이다. 사용자가 임의로 지정하면 된다.
- (int i, int j) : 함수에서 사용할 매개변수다. 호출하는 곳에서 보내준 값의 순서와 자료형이 일치해야 한다.

- **예 1** 과 같이 main() 함수가 시작되기 전에 함수를 정의한 경우에는 프로토타입을 선언하지 않아도 되지만, **예 2** 와 같이 main() 함수가 시작된 후에 함수를 정의한 경우에는 main() 함수 전에 사용될 함수에 대한 프로토타입을 선언해야 한다.

### 예 1

```
int func(int i, int j)
{
    func 함수 코드;
}

main()
{
    메인 함수 코드;
}
```



### 전문가의 조언

프로토타입의 개념과 C언어에서의 프로토타입 선언 방법을 알아두세요.



### 전문가의 조언

main() 함수 앞에 프로토타입을 선언하는 것은 main() 함수가 시작되기 전에 이런 함수를 사용할 것이라고 알려주는 것입니다. 최근에는 컴파일러가 발전하여 사용할 함수를 main() 함수 뒤에 배치한 경우에도 프로토타입 선언 없이 컴파일 가능하지만, 프로토타입을 미리 선언하는 것이 원칙이라는 것을 기억해 두세요.

예 2

```
int func(int i, int j) // func() 함수의 프로토타입 선언

main()
{
    메인 함수 코드;
}

int func(int i, int j)
{
    func 함수 코드;
}
```



기출문제 따라잡기

Section 140

출제예상

1. 다음 중 프로토타입에 대한 설명으로 잘못된 것은?

- ① 프로그래밍 언어에서 프로토타입이란 함수 원형 (Function Prototype)이라는 의미이다.
- ② 프로토타입은 컴파일러에게 사용될 함수에 대한 정보를 미리 알리는 것이다.
- ③ 함수가 호출되기 전에 함수가 미리 정의되는 경우에는 프로토타입을 선언하지 않아도 된다.
- ④ 프로토타입에 정의된 반환 형식은 함수 정의에 지정된 반환 형식과 반드시 일치할 필요는 없다.

프로토타입이 컴파일러에게 사용될 함수에 대한 정보를 미리 알리는 것이기 때문에 프로토타입에 정의된 반환 형식과 함수에 지정된 반환 형식은 반드시 일치해야 합니다.

▶ 정답 : 1. ④



1. 다음 중 조건의 참(True)과 거짓(False) 여부를 판단하여 저장할 때 사용하는 데이터 타입은?

- ① Integer                      ② Character  
③ Boolean                    ④ Array

2. C언어의 제어 구조(Control Structure)에서 문장을 실행한 다음, 조건을 검사하여 반복 실행의 여부를 결정하는 문은?

- ① for문                      ② while문  
③ do~while문              ④ switch~case 문

3. C 언어에서 사용하는 기억클래스의 종류가 아닌 것은?

- ① Automatic Variable  
② Register Variable  
③ Message Variable  
④ Static Variable

4. 다음 중 변수명 작성에 대한 설명으로 가장 옳지 않은 것은?

- ① 예약어는 변수명으로 사용할 수 없다.  
② 첫 글자는 반드시 영문자나 \_(under bar)로 시작해야 한다.  
③ 중간에 공백을 포함할 수 없다.  
④ 대·소문자를 구분하지 않는다.

5. 다음 코드의 결과 값은?

```
int x = 4, y = 7;
int resultxy;
resultxy = x & y;
printf("%d ", resultxy);
```

- ① 0                      ② 4  
③ 7                      ④ 11

6. 다음 코드의 결과 값은?

```
int main(void)
{
    int x = 3;
    int resultxy;
    resultxy = 1 + x << 2;
    printf("%d", resultxy);
    return 0;
}
```

- ① 2                      ② 4  
③ 8                      ④ 16

7. 다음 코드의 실행 결과는?

```
int main(void)
{
    int a = 3, b = 6;
    int c, d, e;
    c = a & b;
    d = a | b;
    e = a ^ b;
    printf("%d %d %d\n", c, d, e);
}
```

- ① 2 2 5                      ② 2 7 5  
③ 5 2 2                      ④ 5 7 2

8. 다음 프로그램의 실행 결과에 의해 변수 a와 b에 저장된 값은? (단, "<<"는 왼쪽 시프트(Lsh), ">>"는 오른쪽 시프트(Rsh)를 의미한다.)

```
int a = 16, b = 64;
a = a >> 2;
b = b << 2;
```

- ① a = 4, b = 256              ② a = 8, b = 128  
③ a = 32, b = 32              ④ a = 64, b = 16

9. 다음 명령을 실행했을 때 a와 b의 값으로 옳은 것은?

```
if (a >= 5 )
{ b = 7;
  if (a > 10) b = b + 5;
}
else b = 6;
```

- ① a가 5일 때 b는 5  
② a가 11일 때 b는 12  
③ a가 7일 때 b는 6  
④ a가 3일 때 b는 7

10. 'switch(a)~'로 시작하는 명령문이 있다고 할 때, 다음 설명 중 틀린 것은?

- ① a의 값에 따라 실행할 명령문이 달라진다.  
② a의 값으로 실수형도 가능하다.  
③ a와 일치하는 레이블이 없으면 default 이하의 명령을 실행한다.  
④ switch문 안에 switch문을 또 쓸 수 있다.

**11. 다음 C 프로그램에서 최종적으로 출력되는 값은?**

```
void main()
{
    int i = 10, hap = 0;
    while (i > 1)
    {
        i--;
        if (i % 3 == 1)
            hap += i;
    }
    printf("%d\n", hap);
}
```

- ① 10                      ② 11  
③ 12                      ④ 13

**12. C언어에서 포인터를 사용하여 두 변수 a, b의 값을 교체하는 경우 빈 칸에 알맞은 코드는?**

```
int a = 10, b = 20, temp;
int *pa = &a;
int *pb = &b;
temp = *pa;

*pb = temp;
```

- ① b = &a;                      ② a = b;  
③ \*pb = \*pa;                  ④ \*pa = \*pb;

**13. C언어의 포인터 조작 연산에서 변수 pc에 대입되는 것과 같은 결과를 갖는 것은?**

```
char *pc, array1[100];
pc = array1;
```

- ① pc = &array1[0];              ② pc = &array1[2];  
③ pc = array1[10];              ④ pc = array1[1];

**14. 다음 C언어 코드의 결과 값은?**

```
void main (void)
{
    int x[ ] = {100, 200, 300, 400};
    int y;
    y = *(x+2)+50;
    printf("%d", y);
}
```

- ① 150                      ② 250  
③ 350                      ④ 450

**15. C 언어의 특징으로 옳지 않은 것은?**

- ① 이식성이 뛰어나 컴퓨터 기종에 관계없이 프로그램을 작성할 수 있다.  
② 융통성을 중요시하는 Interpreter 언어의 범주이다.  
③ UNIX 운영체제를 구성하는 시스템 프로그램이다.  
④ 포인터에 의한 번지 연산 등 다양한 연산 기능을 가진다.

**16. 객체지향(Object-Oriented)의 개념 설명 중 가장 옳지 않은 것은?**

- ① 클래스(Class) : 데이터값을 저장하는 필드와 이 필드에서 연산하는 메소드로 정의  
② 속성(Attribute) : 객체들이 갖고 있는 데이터의 값으로 파일 처리에서 객체는 레코드, 속성은 필드와 유사한 개념  
③ 객체(Object) : 데이터 구조와 이 구조 하에서 이루어진 연산들이 모여서 하나의 독립된 기능을 수행하는 것  
④ 메소드(Method) : 객체들 사이에서 정보를 교환하기 위한 수단

**17. 다음은 객체지향에 관한 설명이다. 옳바르지 못한 것은?**

- ① 객체지향의 특징은 추상화, 정보 은닉, 모듈화이다.  
② 객체의 동작 지시는 메시지에 의해 수행된다.  
③ 객체 중심에서는 재사용의 기능을 이용할 수 있다.  
④ 객체 중심은 구조적 코딩 기능을 극대화할 수 있다.

**18. 객체지향 시스템에서는 객체가 시스템을 구성하는 기본 단위인데, 이런 객체 중에는 같은 특성을 갖는 객체들이 많다. 이와 같이 같은 특성을 갖는 객체를 표현한 것을 무엇이라 하는가?**

- ① 속성                      ② 클래스  
③ 메시지                      ④ 인스턴스

**19. 다음 중 게시판 입력, 상품 검색, 회원 가입 등과 같은 데이터베이스 처리 작업을 수행하기 위해 사용하며, 웹 서버에서 작동하는 스크립트 언어들로만 모아 놓은 것은?**

- ① HTML, XML, SGML  
② Java, Java Applet, Java Script  
③ Java Script, VB Script,  
④ ASP, JSP, PHP

**20.** 다음 중 언어의 분류에 관한 설명으로 옳지 않은 것은?

- ① 논리형 언어는 기호 논리학을 기반으로 두고 있다.
- ② 함수형 언어는 적용형 언어라고도 한다.
- ③ 논리형 언어는 선언형 언어에 포함된다.
- ④ 함수형 언어는 폰노이만 구조에 개념적 기초를 두고 있다.

**21.** JAVA의 대표적인 표준 라이브러리에 대한 설명이 잘못된 것은?

- ① JAVA에서 패키지를 사용하려면 'import java.util'와 같이 import문을 이용해 선언한 후 사용해야 한다.
- ② import로 선언된 패키지 안에 있는 클래스의 메소드를 사용할 때는 클래스와 메소드를 콜론(:)으로 구분하여 'Math.abs()'와 같이 사용한다.
- ③ JAVA의 주요 패키지에는 java.lang, java.util, java.io, java.net 등이 있다.
- ④ 'java.lang' 패키지는 자바에 기본적으로 필요한 인터페이스, 자료형, 예외 처리 등에 관련된 기능을 제공한다.

**22.** 다음은 C언어 프로그램이다. 실행 결과는?

```
main()
{
    printf("%2.1f\n", 123.45);
}
```

- ① 123.4                      ② error
- ③ 123.5                      ④ 23.4

**23.** C언어에서 정수형 변수 a에 256이 저장되어 있다. 이를 7자리로 잡아 왼쪽으로 붙여 출력하려고 할 때, 적절한 printf() 내의 % 변환 문자 사용은?

- ① %7f                      ② %7d
- ③ %-7d                      ④ %-7i

**24.** 다음에서 C언어의 입·출력 명령이 잘못 표현된 것은?

- ① getchar("%c", i);
- ② printf("%3d", i);
- ③ putchar(a);
- ④ scanf("%5d", &i);

**25.** 다음 중 잘못 구성된 것은?

```
main()
{
    int i;
    float j;
    char string[10];
    scanf("%f %f\n", &i, &j);
    scanf("%s\n", string);
}
```

- ① float j;
- ② char string[ ];
- ③ scanf("%f %f\n", &i, &j);
- ④ scanf("%s\n", string);

**26.** 다음은 JAVA에서의 주요 예외 처리 객체에 대한 원인을 설명한 것이다. 객체에 대한 원인이 잘못 설명된 것은?

- ① ClassNotFoundException : 클래스를 찾지 못한 경우
- ② FileNotFoundException : 파일을 찾지 못한 경우
- ③ ArithmeticException : 숫자 형식으로 변환할 수 없는 문자열을 숫자 형식으로 변환하는 경우
- ④ ArrayIndexOutOfBoundsException : 배열의 범위를 벗어난 접근을 시도하는 경우

**1. Section 126**

- 정수 타입(Integer Type) : 정수, 즉 소수점이 없는 숫자를 저장할 때 사용함
- 문자 타입(Character Type) : 한 문자를 저장할 때 사용함
- 배열 타입(Array Type) : 같은 타입의 데이터 집합을 만들어 저장할 때 사용함

**2. Section 130**

- for문 : 초기값, 최종값, 증가값을 지정하는 수식을 이용해 정해진 횟수를 반복하는 제어문
- while문 : 조건이 참인 동안 실행할 문장을 반복 수행하는 제어문. 조건이 처음부터 거짓이면 한 번도 수행하지 않음
- switch문 : 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리할 수 있는 제어문

**3. Section 127**

C언어에서 사용하는 4가지 기억 클래스는 Auto(자동), Register(레지스터), Static(정적), Extern(외부)이다.

**4. Section 127**

변수명은 대·소문자를 구분한다.

**5. Section 128, 138**

```
1 int x = 4, y = 7;
2 int resultxy;
3 resultxy = x & y;
4 printf("%d ", resultxy);
```

- 1 정수형 변수 x에 4, y에 7을 저장한다.
- 2 정수형 변수 resultxy를 선언한다.
- 3 x와 y에 저장된 4와 7을 정수형인 4Byte 크기의 2진수로 변환한 후 비트별로 and 연산(&)한 결과를 resultxy에 저장한다.

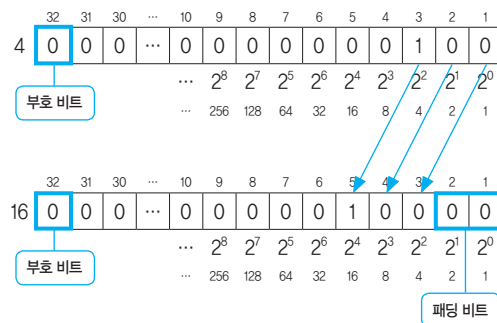
	0000	...	0000	0100(4)
AND(&)	0000	...	0000	0111(7)
	0000	...	0000	0100(4)

- 4 resultxy에 저장된 값 4를 10진수로 출력한다.

**6. Section 128, 138**

```
int main(void)
{
1 int x = 3;
2 int resultxy;
3 resultxy = 1 + x << 2;
4 printf("%d", resultxy);
5 return 0;
}
```

- 1 정수형 변수 x를 3으로 초기화한다.
- 2 정수형 변수 resultxy를 선언한다.
- 3 '1 + x'를 실행하면 4이고, 4를 왼쪽으로 2비트 시프트 연산(<<)을 실행한 후 그 결과를 resultxy에 저장한다.
  - 왼쪽 시프트(<<)는 왼쪽으로 1비트 시프트 할 때마다 2배씩 증가하므로, 2비트 시프트하면 원래의 값 4에  $2^2$ 를 곱한 값이 된다. 즉 16이 된다.
  - 다음과 같이 원래의 값을 정수형인 4Byte 크기의 2진수로 변환한 후 왼쪽으로 2비트 시프트한 다음 10진수로 변환해서 결과를 확인할 수 있다.
  - 부호를 제외한 전체 비트를 왼쪽으로 2비트 이동시키면, 빈 자리(패딩 비트)에는 0이 들어온다.



- 4 resultxy에 저장된 값 16을 10진수로 출력한다.
- 5 0을 반환하고 프로그램을 종료한다. 0을 반환하는 이유는 메인 함수가 문제없이 종료되었음을 나타내기 위함이다.





## 7. Section 128, 138

```
int main(void)
{
    ❶ int a = 3, b = 6;
    ❷ int c, d, e;
    ❸ c = a & b;
    ❹ d = a | b;
    ❺ e = a ^ b;
    ❻ printf("%d %d %d\n", c, d, e);
}
```

- ❶ 정수형 변수 a를 3으로, b를 6으로 초기화한다.
- ❷ 정수형 변수 c, d, e를 선언한다.
- ❸ a(3)와 b(6)를 정수형인 4Byte 크기의 2진수로 변환하고 비트별로 AND(&) 연산을 한 후 그 결과를 c에 저장한다.

0000	...	0011(3)
AND(&)	0000	0110(6)
	0000	0010(2)

- ❹ a(3)와 b(6)를 정수형인 4Byte 크기의 2진수로 변환하고 비트별로 OR(|) 연산을 한 후 그 결과를 d에 치환한다.

0000	...	0011(3)
OR( )	0000	0110(6)
	0000	0111(7)

- ❺ a(3)와 b(6)를 정수형인 4Byte 크기의 2진수로 변환하고 비트별로 XOR(^) 연산을 한 후 그 결과를 e에 치환한다.

0000	...	0011(3)
XOR(^)	0000	0110(6)
	0000	0101(5)

- ❻ c(2), d(7), e(5)의 값을 정수형 10진수로 출력한 후 커서를 다음 줄 앞으로 이동한다.

## 8. Section 128

- 오른쪽 시프트(>>)는 오른쪽으로 1비트 시프트 할 때마다 2배씩 감소하므로, 2비트 시프트하면 원래의 값 16을  $2^2$ 로 나눈 값이 된다. 즉 4가 된다.  
다음과 같이 원래의 값을 오른쪽으로 2비트 시프트한 후 10진수로 변환해서 결과를 확인할 수 있다.

원래의 값: 0 0 0 1 0 0 0 0 = 16

0 0 0 0 0 1 0 0 = 4

× × ×

$2^2$   $2^1$   $2^0$

4 + 0 + 0 = 4

- 왼쪽 시프트(<<)는 왼쪽으로 1비트 시프트 할 때마다 2배씩 증가하므로, 2비트 시프트하면 원래의 값 64에  $2^2$ 를 곱한 값이 된다. 즉 256이 된다.  
다음과 같이 원래의 값을 왼쪽으로 2비트 시프트한 후 10진수로 변환해서 결과를 확인할 수 있다.

원래의 값: 0 0 1 0 0 0 0 0

1 0 0 0 0 0 0 0

× × × × × × × ×

$2^8$   $2^7$   $2^6$   $2^5$   $2^4$   $2^3$   $2^2$   $2^1$   $2^0$

256 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 256

## 9. Section 129

a가 5일 때 b는 7, a가 7일 때 b는 7, a가 3일 때 b는 6이다. 문제에 사용된 코드의 의미는 다음과 같다.

```
❶ if (a >= 5)
❷ { b = 7;
❸   if (a > 10) b = b + 5;
❹ }
❺ else b = 6;
```

- ❶ a가 5보다 크거나 같으면 ❷~❸번을 수행하고 그렇지 않으면 ❹번을 수행한다.
- ❷ b에 7을 치환한다.
- ❸ a가 10보다 크면 b에 b+5를 치환한다.
- ❹ a가 5보다 작으면 b에 6을 치환한다.

## 10. Section 129

a에는 int, char, enum 형의 자료만 사용할 수 있다.



## 11. Section 129, 130

지문에 사용된 코드의 의미는 다음과 같다.

```
void main()
{
    ❶ int i = 10, hap = 0;
    ❷ while (i > 1)
    ❸ {
    ❹ i--;
    ❺ if (i % 3 == 1)
    ❻ hap += i;
    ❼ }
    ❽ printf("%d\n", hap);
}
```

- ❶ 정수형 i는 10, hap은 0으로 초기화 한다.
- ❷ i가 1보다 큰 동안 ❸~❹번 사이의 문장을 반복하여 수행한다.
- ❸ ❸~❹번까지가 반복문의 범위이다.
- ❹ i = i - 1;과 동일하다. i의 값을 1씩 감소시킨다.
- ❺ i를 3으로 나눈 나머지가 1이면 ❻번 문장을 수행하고, 아니면 ❼번 문장으로 이동한다.
- ❻ hap = hap + i;와 동일하다. i의 값을 hap에 누적시킨다.
- ❼ 반복문의 끝이다. ❷번으로 이동하여 조건을 판단한 다음 반복 여부를 결정한다.
- ❽ hap를 정수형 10진수로 출력한 후 커서를 다음 줄 맨 처음으로 이동한다.

반복문 실행에 따른 변수의 변화는 다음과 같다.

반복횟수	i	i % 3	hap
1	10		0
2	9	0	
3	8	2	
4	7	1	7
5	6	0	
6	5	2	
7	4	1	11
8	3	0	
9	2	2	
반복실행 안됨	1	1	12

※ i가 2가 되었을 때 i--를 수행하면 i는 1, i % 3은 1이 되므로 hap에 더하면 12가 되고, i가 1이 되었을 때 반복문을 종료한다.

## 12. Section 132

지문에 사용된 코드의 의미는 다음과 같다.

```
❶ int a = 10, b = 20, temp;
❷ int *pa = &a;
❸ int *pb = &b;
❹ temp = *pa;
❺ *pa = *pb
❻ *pb = temp;
```

- ❶ 정수형 변수 a, b, temp를 선언하면서, a에는 10, b에는 20을 저장한다.

주소      메모리      변수

1000	10	a
1004	20	b
1008		temp

- ❷ 정수형 변수의 주소를 저장할 포인터 변수 pa를 선언하면서 정수형 변수 a의 주소를 저장한다.

	주소	메모리	변수
*pa			
1000		1000	10 a
		1004	20 b
		1008	temp

- ❸ 포인터 변수 pb를 선언하면서 정수형 변수 b의 주소를 저장한다.

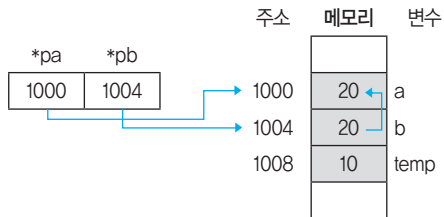
	주소	메모리	변수
*pa			
*pb			
1000		1000	10 a
		1004	20 b
		1008	temp

- ❹ 포인터 변수 pa가 가리키는 곳의 값을 temp에 저장한다.

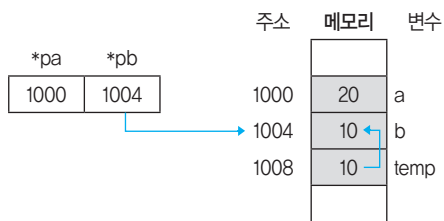
	주소	메모리	변수
*pa			
*pb			
1000		1000	10 a
		1004	20 b
		1008	10 temp



- ⑤ 포인터 변수 `pa`가 가리키는 곳에 `pb`가 가리키는 곳의 값을 저장한다.



- ⑥ 포인터 변수 `pb`가 가리키는 곳에 `temp`에 저장된 값을 저장한다.



### 13. Section 132

지문에 사용된 코드의 의미는 다음과 같다.

- ① `char *pc, array1[100];`
- ② `pc = array1;`

- ① 문자형 변수의 주소를 기억할 문자형 포인터 변수 `pc`와 100개의 요소를 갖는 문자형 배열 `array1`을 선언한다.
  - ② 배열의 대표명 `array1`는 배열의 주소이므로 포인터 변수 `pc`에는 배열 `array1`의 시작 위치가 기억된다.
- ※ 포인터 변수 `pc`에는 배열 `array1`의 시작 위치가 기억되므로 `array1` 배열 첫 번째 요소의 주소인 `&array1[0]`을 지정해도 같은 주소가 기억된다.
- 포인터 변수에 주소를 저장하기 위해 변수의 주소를 구할 때는 변수 앞에 `&`를 붙인다.
  - C언어에서 배열의 요소는 0부터 시작한다.

### 14. Section 132

```
void main (void)
{
    ① int x[ ] = {100, 200, 300, 400};
    ② int y;
    ③ y = *(x+2)+50;
    ④ printf("%d", y);
}
```

- ① 정수형 배열 `x`를 선언하고 다음과 같이 초기화한다.

x[0]	x[1]	x[2]	x[3]
100	200	300	400

※ 배열을 선언할 때 배열의 개수를 생각하고 초기값을 지정하면, 초기값으로 지정된 값의 수와 같은 크기의 배열이 선언된다.

- ② 정수형 변수 `y`를 선언한다.

- ③ `y`에 배열 `x`의 주소에서 2만큼 증가한 주소의 값(`*(x+2)`)에 50을 더한 후 그 값을 저장한다.

※ 배열의 대표명은 배열의 첫 번째 요소의 주소를 가리킨다. 즉 `x`는 `x[0]`의 주소를 말한다.

※ 배열 `x`가 가리키는 곳의 주소에서 2 증가한다는 것은 현재 `x`가 가리키고 있는 정수형 자료의 주소(`x[0]`)에서 정수형 자료의 크기만큼 2번 증가하는 것이므로 `x[2]`의 주소가 된다.

주소				
0				
...	x[0]	x[1]	x[2]	x[3]
200	100	200	300	400
	x	x+1	x+2	x+3
...	200	204	208	212

- ④ `y`에 저장된 값을 10진수로 출력한다.

### 15. Section 133

C 언어는 컴파일 과정을 거쳐야 실행할 수 있는 컴파일러 언어이다.

### 16. Section 134

메소드는 객체의 상태를 참조하거나 변경하기 위한 수단이고, 객체들 사이에서 정보를 교환하기 위한 수단은 메시지이다.

### 17. Section 134

객체지향 기법은 구조적 방법에서의 생산성 문제를 해결하기 위한 새로운 방법이다.

### 18. Section 134

- 속성 : 객체가 가지고 있는 정보로 객체의 상태, 분류 등을 나타냄
- 메시지 : 객체들 간 상호작용을 하는 데 사용되는 수단으로, 객체에게 어떤 행위를 하도록 지시하는 명령 또는 요구사항



- 인스턴스 : 클래스에 속한 각각의 객체

### 19. Section 135

- 서버용 스크립트 언어 : ASP, JSP, PHP 등
- 클라이언트용 스크립트 언어 : 자바 스크립트

### 20. Section 136

폰노이만 구조에 개념적 기초를 두고 있는 것은 명령형 언어이다.

### 21. Section 137

클래스의 메소드를 호출할 때는 클래스와 메소드를 마침표(.)로 구분하여 'Math.abs()'와 같이 사용한다.

### 22. Section 138

'%2.1f'는 2자리를 확보하여 소수점 이하 1자리까지 표시한다는 의미이다. 지정한 자릿수보다 출력할 값이 클 경우 정수 부분은 모두 표시하고, 소수점 이하 부분은 반올림하여 지정한 자릿수까지만 표시한다.

### 23. Section 138

서식 변환 문자임을 나타내려면 %, 왼쪽에 붙여 출력하려면 -, 7자리를 확보하려면 7, 정수를 출력하려면 d를 차례로 지정하면 된다.

### 24. Section 138

getchar() 함수는 서식 문자열을 사용할 수 없다.

### 25. Section 138

- 서식 문자열과 입력받은 값을 저장할 변수는 자료형이 같아야 한다.
- i는 정수형 변수이므로 '%f'가 아닌 '%d'로 서식 문자열을 지정해야 한다.

### 26. Section 139

- ArithmeticException 객체는 0으로 나누는 등의 산술 연산에 대한 예외가 발생한 경우에 사용된다.
- ③번은 NumberFormatException 객체에 대한 발생 원인이다.

# 3 장

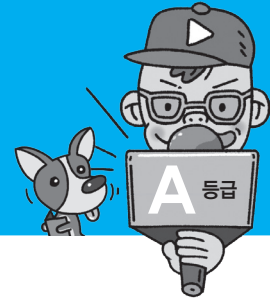
## 응용 SW 기초 기술 활용

- 141 운영체제의 개념 **A** 등급
- 142 Windows **C** 등급
- 143 UNIX / LINUX / MacOS **A** 등급
- 144 기억장치 관리의 개요 **A** 등급
- 145 주기억장치 할당 기법 **C** 등급
- 146 가상기억장치 구현 기법 / 페이지 교체 알고리즘 **A** 등급
- 147 가상기억장치 기타 관리 사항 **A** 등급
- 148 프로세스의 개요 **A** 등급
- 149 스케줄링 **B** 등급
- 150 환경 변수 **C** 등급
- 151 운영체제 기본 명령어 **B** 등급
- 152 인터넷 **A** 등급
- 153 OSI 참조 모델 **A** 등급
- 154 네트워크 관련 장비 **B** 등급
- 155 프로토콜의 개념 **B** 등급
- 156 TCP/IP **A** 등급



이 장에서 꼭 알아야 할 키워드 **Best 10**

1. 운영체제 2. UNIX 3. 배치 전략 4. FIFO 5. LRU 6. 프로세스 7. 스레드 8. IPv6 9. OSI 참조 모델  
10. TCP/IP



## 전문가의 조언

운영체제에 대한 기본적인 내용들입니다. 기초를 탄탄히 한다는 마음가짐으로 확실하게 숙지하고 넘어가세요.

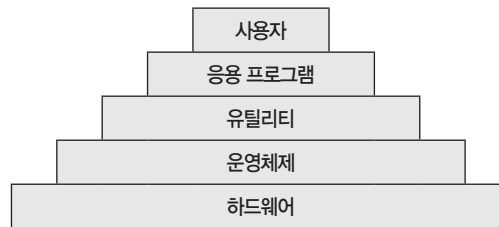
**자원**  
자원은 시스템에서 사용할 수 있는 CPU, 주기억장치, 보조기억장치, 프린터, 파일 및 정보 등을 의미합니다.

**스케줄링(Scheduling)**  
스케줄링은 어떤 자원을 누가, 언제, 어떤 방식으로 사용할지를 결정해 주는 것입니다.

## 1 운영체제(OS; Operating System)의 정의

운영체제는 컴퓨터 시스템의 자원\*들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임이다.

- 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어의 일종으로, 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 제공해준다.



## 2 운영체제의 목적

운영체제의 목적에는 처리 능력 향상, 사용 가능도 향상, 신뢰도 향상, 반환 시간 단축 등이 있다.

- 처리 능력, 반환 시간, 사용 가능도, 신뢰도는 운영체제의 성능을 평가하는 기준이 된다.

처리 능력(Throughput)	일정 시간 내에 시스템이 처리하는 일의 양
반환 시간(Turn Around Time)	시스템에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간
사용 가능도(Availability)	시스템을 사용할 필요가 있을 때 즉시 사용 가능한 정도
신뢰도(Reliability)	시스템이 주어진 문제를 정확하게 해결하는 정도

## 3 운영체제의 기능

- 프로세서(처리기, Processor), 기억장치(주기억장치, 보조기억장치), 입·출력장치, 파일 및 정보 등의 자원을 관리한다.
- 자원을 효율적으로 관리하기 위해 자원의 스케줄링\* 기능을 제공한다.
- 사용자와 시스템 간의 편리한 인터페이스를 제공한다.
- 시스템의 각종 하드웨어와 네트워크를 관리·제어한다.
- 데이터를 관리하고, 데이터 및 자원의 공유 기능을 제공한다.
- 시스템의 오류를 검사하고 복구한다.

- 자원 보호 기능을 제공한다.
- 입·출력에 대한 보조 기능을 제공한다.
- 가상 계산기\* 기능을 제공한다.

## 4 운영체제의 주요 자원 관리

운영체제에서는 다음과 같은 자원 관리 기능을 수행한다.

자원	기능
프로세스* 관리	<ul style="list-style-type: none"> <li>• 프로세스 스케줄링 및 동기화 관리 담당</li> <li>• 프로세스 생성과 제거, 시작과 정지, 메시지 전달 등의 기능 담당</li> </ul>
기억장치 관리	프로세스에게 메모리 할당 및 회수 관리 담당
주변장치 관리	입·출력장치 스케줄링 및 전반적인 관리 담당
파일 관리	파일의 생성과 삭제, 변경, 유지 등의 관리 담당

## 5 운영체제의 종류

운영체제의 종류에는 Windows, UNIX, LINUX, MacOS, MS-DOS 등이 있다.

운영체제	특징	인터페이스
Windows	마이크로소프트(Microsoft) 사가 개발한 운영체제	GUI*
UNIX	AT&T 벨(Bell) 연구소, MIT, General Electric이 공동 개발한 운영체제	CLI*
LINUX	<ul style="list-style-type: none"> <li>• UNIX와 호환이 가능한 커널(Kernel)*이며, 리누스 토발즈(Linus Torvalds)가 개발한 운영체제</li> <li>• 누구나 제한 없이 활용 및 재배포가 가능</li> </ul>	CLI
MacOS	애플(Apple) 사가 UNIX를 기반으로 개발한 운영체제	GUI
MS-DOS	Windows 이전에 사용되던 운영체제	CLI

- 단일 작업 처리 시스템\*에는 MS-DOS, 다중 작업 처리 시스템\*에는 Windows, UNIX, LINUX, MacOS 등이 사용된다.
- Windows, MacOS, MS-DOS는 개인용, UNIX, LINUX는 서버용 운영체제이다.

잠깐만요



단일 작업 처리 시스템 / 다중 작업 처리 시스템

### 단일 작업 처리 시스템(Single Tasking System)

- 컴퓨터 시스템을 한 개의 작업이 독점하여 사용하는 방식입니다.
- 예를 들어 DOS에서 워드 작업을 하다가 PC 통신을 하려면 워드 작업을 종료해야 하는 것을 의미합니다.

### 다중 작업 처리 시스템(Multi-Tasking System)

- 여러 개의 프로그램을 열어 두고 다양한 작업을 동시에 진행하는 방식입니다.
- 예를 들어 Windows에서 워드 작업을 하고 있는 상태에서 음악을 들으며 엑셀, 그림판 등의 프로그램을 실행시켜 놓고, 필요할 때마다 해당 프로그램으로 바로 바로 전환하여 사용할 수 있는 것을 의미합니다.

### 가상 계산기(Virtual Computer)

가상 계산기는 한 대의 컴퓨터를 여러 대의 컴퓨터처럼 보이게 하는 가상 컴퓨터 운영체제에 의해 만들어지며 사용자의 관점에서는 가상 컴퓨터가 실제 컴퓨터처럼 보일 수도 있고 아주 다르게 보일 수도 있습니다.

### 프로세스(Process)

프로세스는 일반적으로 실행중인 프로그램을 의미합니다.

### GUI(Graphic User Interface)

GUI는 키보드로 명령어를 직접 입력하지 않고, 마우스로 아이콘이나 메뉴를 선택하여 모든 작업을 수행하는 그래픽 사용자 인터페이스를 의미합니다.

### CLI(Command Line Interface)

CLI는 키보드로 명령어를 직접 입력하여 작업을 수행하는 사용자 인터페이스를 의미합니다. CUI(Character User Interface)라고도 합니다.

### 커널(Kernel)

커널은 운영체제의 가장 중요한 핵심 부분으로, 운영체제의 다른 모든 부분에 다양하고 기본적인 서비스를 제공하는 역할을 합니다.



## 기출문제 따라잡기

Section 141

이전기출

1. 컴퓨터 시스템을 계층적으로 묘사할 때 운영체제의 위치는 다음 그림의 어느 부분에 해당되는가?



- ① 사용자와 응용 프로그램 사이
- ② 응용 프로그램과 유틸리티 사이
- ③ 유틸리티와 하드웨어 사이
- ④ 하드웨어 아래

운영체제는 '사용자와 하드웨어 간의 인터페이스 제공' 잊지 않았죠? 운영체제가 위치할 하드웨어의 다음 계층이 어디인지 찾아보세요.

이전기출

2. 운영체제에 대한 설명으로 옳지 않은 것은?

- ① 다중 사용자와 다중 응용 프로그램 환경 하에서 자원의 현재 상태를 파악하고, 자원 분배를 위한 스케줄링을 담당한다.
- ② CPU, 메모리 공간, 기억장치, 입·출력장치 등의 자원을 관리한다.
- ③ 운영체제의 종류로는 매크로 프로세서, 어셈블러, 컴파일러 등이 있다.
- ④ 입·출력장치와 사용자 프로그램을 제어한다.

운영체제의 종류만 알면 풀 수 있겠죠? 매크로 프로세서, 어셈블러, 컴파일러는 운영체제와 같이 시스템 소프트웨어의 한 종류입니다.

이전기출

3. 운영체제의 설명으로 옳지 않은 것은?

- ① 운영체제는 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 일종의 하드웨어 장치다.
- ② 운영체제는 컴퓨터를 편리하게 사용하고 컴퓨터 하드웨어를 효율적으로 사용할 수 있도록 한다.
- ③ 운영체제는 스스로 어떤 유용한 기능도 수행하지 않고 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 마련하여 준다.
- ④ 운영체제는 중앙처리장치의 시간, 메모리 공간, 파일, 기억장치 등의 자원을 관리한다.

보기가 길어 어려워 보일 수 있지만 운영체제는 여러 프로그램이 모인 소프트웨어라는 것만 알면 쉽게 풀 수 있는 문제입니다. 기억해 두세요. 운영체제는 소프트웨어입니다.

이전기출

4. 운영체제의 기능으로 틀린 것은?

- ① 자원의 스케줄링 기능을 제공한다.
- ② 자원 보호 기능을 제공한다.
- ③ 사용자와 시스템 간의 편리한 인터페이스를 제공한다.
- ④ 목적 프로그램과 라이브러리, 실행 프로그램 등을 연결하여 실행 가능한 로드 모듈을 만든다.

운영체제의 기능을 묻는 문제가 자주 출제되고 있네요. ④번은 링커(Linker)의 기능입니다.

이전기출

5. 운영체제의 목적으로 거리가 먼 것은?

- ① 처리 능력의 향상
- ② 반환 시간의 최대화
- ③ 사용 가능성 증대
- ④ 신뢰도 향상

반환 시간은 작업을 요청하고 완료될 때까지의 시간을 말하는데, 이것이 최대화 되는 것을 목적으로 할까요?

이전기출

6. 운영체제의 성능 판단 요소로 거리가 먼 것은?

- ① 처리 능력
- ② 비용
- ③ 신뢰도
- ④ 사용 가능성

운영체제의 성능 판단 기준에는 처리 능력, 반환 시간, 사용 가능성, 신뢰도가 있습니다. 네 가지 판단 기준을 기억하세요.

이전기출

7. 운영체제의 성능평가 요인 중 다음 설명에 해당하는 것은?

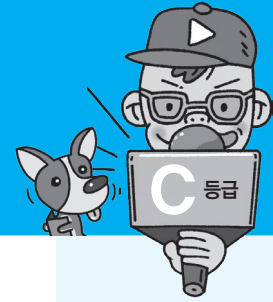
이것은 컴퓨터 시스템 내의 한정된 각종 자원을 여러 사용자가 요구할 때, 어느 정도 신속하고 충분히 지원해 줄 수 있는지의 정도이다. 이는 사용 가능한 하드웨어 자원의 수나 다중 프로그래밍 정도 등의 요소가 좌우하는 것으로 같은 종류의 시스템 자원수가 많을 경우에는 이것이 높아질 수 있다.

- ① Throughput
- ② Availability
- ③ Turn Around Time
- ④ Reliability

지문의 내용을 요약하면 '자원이 필요할 때 즉시 사용할 수 있는 정도'입니다. 이와 관련된 성능평가 기준을 찾아보세요.

▶ 정답 : 1. ③ 2. ③ 3. ① 4. ④ 5. ② 6. ② 7. ②





## 1 Windows의 개요

Windows는 1990년대 마이크로소프트(Microsoft) 사가 개발한 운영체제이다.

- Windows의 버전에는 95, 98, me, XP, Vista, 7, 8, 10 등이 있다.
- Windows의 주요 특징에는 GUI, 선점형 멀티태스킹, OLE, PnP 등이 있다.

## 2 그래픽 사용자 인터페이스(GUI; Graphic User Interface)

그래픽 사용자 인터페이스는 키보드로 명령어를 직접 입력하지 않고, 마우스로 아이콘이나 메뉴를 선택하여 모든 작업을 수행하는 방식을 말한다.

- 초보자도 쉽게 사용할 수 있는 그래픽 사용자 인터페이스(GUI)를 채용하였다.

## 3 선점형 멀티태스킹(Preemptive Multi-Tasking)

선점형 멀티태스킹은 동시에 여러 개의 프로그램을 실행하는 멀티태스킹\*을 하면서 운영체제가 각 작업의 CPU 이용 시간을 제어하여 응용 프로그램 실행중 문제가 발생하면 해당 프로그램을 강제 종료시키고 모든 시스템 자원을 반환하는 방식을 말한다.

- 하나의 응용 프로그램이 CPU를 독점하는 것을 방지할 수 있어 시스템 다운 현상 없이 더욱 안정적인 작업을 할 수 있다.

## 4 PnP(Plug and Play, 자동 감지 기능)

PnP는 컴퓨터 시스템에 프린터나 사운드 카드 등의 하드웨어를 설치했을 때, 해당 하드웨어를 사용하는 데 필요한 시스템 환경을 운영체제가 자동으로 구성해 주는 기능이다.

- 운영체제가 하드웨어의 규격을 자동으로 인식하여 동작하게 해주므로 PC 주변장치를 연결할 때 사용자가 직접 환경을 설정하지 않아도 된다.
- PnP 기능을 활용하기 위해서는 하드웨어와 소프트웨어 모두 PnP를 지원하여야 한다.

## 5 OLE(Object Linking and Embedding)

OLE는 다른 여러 응용 프로그램에서 작성된 문자나 그림 등의 개체(Object)를 현재 작성 중인 문서에 자유롭게 연결(Linking)하거나 삽입(Embedding)하여 편집할 수 있게 하는 기능이다.



### 전문가의 조언

Windows의 특징에는 어떤 것들이 있는지 정리하세요.

### 멀티태스킹(Multi-Tasking, 다중 작업)

멀티태스킹은 여러 개의 프로그램을 동시에 열어 두고 다양한 작업을 동시에 진행하는 것을 말합니다.

예 MP3 음악을 들으면서 워드프로세서 작업을 하다 인터넷에서 파일을 다운로드하는 것

- OLE로 연결된 이미지를 원본 프로그램에서 수정하거나 편집하면 그 내용이 그대로 해당 문서에 반영된다.

## 6 255자의 긴 파일명

Windows에서는 파일 이름을 지정할 때 VFAT(Virtual File Allocation Table)를 이용하여 최대 255자까지 지정할 수 있다.

- 파일 이름으로는 W / : \* ? " < > | 를 제외한 모든 문자 및 공백을 사용할 수 있으며, 한글의 경우 127자까지 지정할 수 있다.

## 7 Single-User 시스템

컴퓨터 한 대를 한 사람만이 독점해서 사용한다.\*

### Single-User / Multi-User

Windows 10과 같은 개인용은 하나의 컴퓨터를 한 사람이 사용하는 Single-User 시스템이고, UNIX, LINUX, Windows NT와 같은 서버용은 하나의 컴퓨터를 동시에 여러 사람이 사용하는 Multi-User 시스템입니다.

### 따라잡기



### 기출문제 따라잡기

### Section 142

#### 이전기술

1. 윈도우에서 사용자가 사용하기 원하는 하드웨어를 시스템에 부착하면 자동으로 인식하여 동작하게 해주는 기능은?

- ① Folding                      ② Plug and Play
- ③ Coalescing                ④ Naming

Windows의 주요 특징을 정리했나요? 확실하지 않으면 다시 한 번 확인해 보세요.

#### 이전기술

2. 윈도우에서 지원하는 OLE의 기능은?

- ① 디스크의 효율적 관리
- ② 모니터, 화질의 개선
- ③ 효율적인 메모리 관리
- ④ 응용 프로그램 간의 자료 공유

OLE는 Object Linking and Embedding(개체의 연결과 삽입)이란 뜻입니다. 이제 답을 찾아보세요.

#### 이전기술

3. Windows에 대한 설명 중 옳지 않은 것은?

- ① PnP 기능을 지원한다.
- ② 멀티태스킹을 지원한다.
- ③ 네트워크 기능이 강화되었다.
- ④ 멀티 유저 시스템이다.

혼동하지 말아야 합니다. Windows는 Single-User, Multi-Tasking, UNIX는 Multi-User, Multi-Tasking!

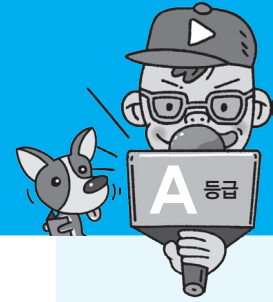
#### 이전기술

4. 컴퓨터의 윈도우 창에 여러 윈도우를 열어 놓고 작업하는 것을 기역장치 처리 방법으로 무엇이라 하는가?

- ① 보조 프로그램
- ② 멀티 프로세싱
- ③ 멀티 프로그래밍
- ④ 리얼 타임 프로그래밍

동시에 여러 개의 프로그램을 실행하는 것을 멀티 태스킹 또는 멀티 프로그래밍이라고 합니다.

▶ 정답 : 1. ② 2. ④ 3. ④ 4. ③



## 1 UNIX의 개요 및 특징

UNIX는 1960년대 AT&T 벨(Bell) 연구소, MIT, General Electric이 공동 개발한 운영체제이다.

- 시분할 시스템(Time Sharing System)을 위해 설계된 대화식 운영체제로, 소스가 공개된 개방형 시스템(Open System)이다.
- 대부분 C 언어로 작성되어 있어 이식성이 높으며 장치, 프로세스 간의 호환성이 높다.
- 크기가 작고 이해하기가 쉽다.
- 다중 사용자(Multi-User), 다중 작업(Multi-Tasking)을 지원한다.
- 많은 네트워킹 기능을 제공하므로 통신망(Network) 관리용 운영체제로 적합하다.
- 트리 구조의 파일 시스템을 갖는다.
- 전문적인 프로그램 개발에 용이하다.
- 다양한 유틸리티 프로그램들이 존재한다.

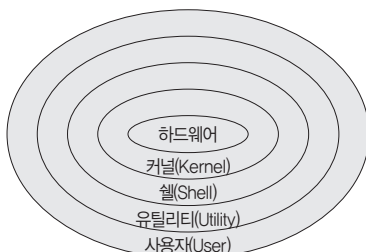
잠깐만요



### 다중 사용자(Multi-User), 다중 작업(Multi-Tasking)

- 다중 사용자(Multi-User)는 여러 사용자가 동시에 시스템을 사용하는 것이고, 다중 작업(Multi-Tasking)은 여러 개의 작업이나 프로그램을 동시에 수행하는 것을 의미합니다.
- 하나 이상의 작업을 백그라운드\*에서 수행하므로 여러 작업을 동시에 처리할 수 있습니다.

## 2 UNIX 시스템의 구성



시스템의 구성

### 전문가의 조언

UNIX의 특징을 정확히 알아두고, UNIX 시스템의 구성 중 커널과 셸의 기능을 구분할 수 있도록 정리하세요.

### 포그라운드 작업과 백그라운드 작업

여러 개의 작업이 동시에 실행될 때 전면에서 실행되는 우선순위가 높은 작업을 포그라운드 작업이라고 하고, 같은 상황에서 우선 순위가 낮아 화면에 보이지 않고 실행되는 프로그램을 백그라운드 작업이라 합니다.

#### 파이프라인

파이프라인은 둘 이상의 명령을 함께 묶어 처리한 결과를 다른 명령의 입력으로 전환하는 기능입니다.

#### 프로세스 간의 통신(IPC; Inter Process Communication)

프로세스 간의 통신은 여러 프로세스, 즉 프로그램 간에 서로 데이터를 주고받는 것을 의미합니다.

#### 시스템 호출

시스템 호출은 프로세스가 커널에 접근하기 위한 인터페이스를 제공하는 명령어입니다.

## 커널(Kernel)

- UNIX의 가장 핵심적인 부분이다.
- 컴퓨터가 부팅될 때 주기억장치에 적재된 후 상주하면서 실행된다.
- 하드웨어를 보호하고, 프로그램과 하드웨어 간의 인터페이스 역할을 담당한다.
- 프로세스(CPU 스케줄링) 관리, 기억장치 관리, 파일 관리, 입·출력 관리, 프로세스 간 통신, 데이터 전송 및 변환 등 여러 가지 기능을 수행한다.

## 셸(Shell)

- 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기이다.
- 시스템과 사용자 간의 인터페이스를 담당한다.
- DOS의 COMMAND.COM과 같은 기능을 수행한다.
- 주기억장치에 상주하지 않고, 명령어가 포함된 파일 형태로 존재하며 보조 기억장치에서 교체 처리가 가능하다.
- 파이프라인\* 기능을 지원하고 입·출력 재지정을 통해 출력과 입력의 방향을 변경할 수 있다.
- 공용 Shell(Bourne Shell, C Shell, Korn Shell)이나 사용자 자신이 만든 Shell을 사용할 수 있다.

## Utility Program

- 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용한다.
- DOS에서의 외부 명령어에 해당된다.
- 유틸리티 프로그램에는 에디터, 컴파일러, 인터프리터, 디버거 등이 있다.

#### 잠깐만요



#### UNIX에서의 프로세스 간 통신\*

각 프로세스는 시스템 호출\*을 통해 커널의 기능을 사용하며, 프로세스 간 통신은 시그널, 파이프, 소켓 등을 사용합니다.

- 시그널(Signal) : 간단한 메시지를 이용하여 통신하는 것으로 초기 UNIX 시스템에서 사용됨
- 파이프(Pipe) : 한 프로세스의 출력이 다른 프로세스의 입력으로 사용되는 단방향 통신 방식
- 소켓(Socket) : 프로세스 사이의 대화를 가능하게 하는 쌍방향 통신 방식

## 3 LINUX의 개요 및 특징

LINUX는 1991년 리누스 토발즈(Linus Torvalds)가 UNIX를 기반으로 개발한 운영 체제이다.

- 프로그램 소스 코드가 무료로 공개되어 있기 때문에 프로그래머가 원하는 기능을 추가할 수 있고, 다양한 플랫폼에 설치하여 사용이 가능하며, 재배포가 가능하다.
- UNIX와 완벽하게 호환된다.
- 대부분의 특징이 UNIX와 동일하다.

## 4 MacOS의 개요 및 특징

MacOS는 1980년대 애플(Apple) 사가 UNIX를 기반으로 개발한 운영체제이다.

- 아이맥(iMac)과 맥북(MacBook) 등 애플사에서 생산하는 제품에서만 사용이 가능하다.
- 드라이버 설치 및 install과 uninstall의 과정이 단순하다.



### 전문가의 조언

MacOS에 대해 공부할 내용이 많지 않으니 편안한 마음으로 가볍게 읽어보고 넘어가세요.



### 기출문제 따라잡기

Section 143

#### 이전기술

#### 1. UNIX의 특징이 아닌 것은?

- ① 트리 구조의 파일 시스템을 갖는다.
- ② Multi-User는 지원하지만 Multi-Tasking은 지원하지 않는다.
- ③ 대화식 운영체제이다.
- ④ 이식성이 높으며, 장치, 프로세스 간의 호환성이 높다.

UNIX의 특징을 묻는 문제가 자주 출제됩니다. UNIX하면 대화식 운영체제, C언어, 트리 구조, Multi-User와 Multi-Tasking 지원, 높은 이식성과 호환성이 바로 떠오를 수 있도록 기억해 두세요.

#### 이전기술

#### 2. UNIX에서 명령어를 백그라운드로 수행시킬 때 가장 큰 장점은?

- ① 기억장치를 작게 차지한다.
- ② CPU를 독점적으로 사용할 수 있다.
- ③ 해당 명령문의 수행 시간을 단축할 수 있다.
- ④ 수행중인 명령문이 끝나기 전에 다른 명령문을 줄 수 있다.

포그라운드 작업과 백그라운드 작업에 대해 배웠죠? 백그라운드 작업이 가능한 것은 UNIX가 다중 작업이 가능하기 때문입니다. 이와 관련된 내용을 찾아보세요.

#### 이전기술

#### 3. UNIX에서 커널에 대한 설명으로 옳지 않은 것은?

- ① UNIX 시스템의 중심부에 해당한다.
- ② 사용자의 명령을 수행하는 명령어 해석기이다.
- ③ 프로세스 관리, 기억장치 관리 등을 담당한다.
- ④ 컴퓨터 부팅시 주기억장치에 적재되어 상주하면서 실행된다.

'명령어 해석'하면 셸의 기능입니다. 커널과 셸의 기능은 확실히 구분할 수 있어야 합니다.

#### 이전기술

#### 4. UNIX에서 두 프로세스를 연결하여 프로세스 간 통신을 가능하게 하며, 한 프로세스의 출력이 다른 프로세스의 입력으로 사용됨으로써 프로세스 간 정보 교환이 가능하도록 하는 것은?

- ① pipe
- ② signal
- ③ fork
- ④ preemption

UNIX에서 사용하는 프로세스 간 통신 방법! 기억하고 있겠죠? 잊었으면 다시 한번 보고 오세요.

#### 이전기술

#### 5. UNIX의 셸(Shell)에 대한 설명으로 옳지 않은 것은?

- ① 시스템과 사용자 간의 인터페이스를 담당한다.
- ② 프로세스 관리, 파일 관리, 입·출력 관리, 기억장치 관리 등의 기능을 수행한다.
- ③ 명령어 해석기 역할을 한다.
- ④ 사용자의 명령어를 인식하여 프로그램을 호출한다.

3번 문제를 풀었다면 아주 쉽게 풀 수 있는 문제네요. 자원 관리하면 커널, 명령어 해석하면 셸 꼭 기억하세요.

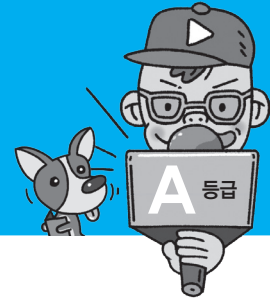
#### 이전기술

#### 6. UNIX 시스템에서 사용자와 운영체제 서비스를 연결해 주는 인터페이스로, 상위 수준의 소프트웨어가 커널의 기능을 이용할 수 있도록 지원해 주는 것은?

- ① 시스템 호출
- ② 하드웨어 제어 루틴
- ③ 프로세스 제어 서브시스템
- ④ 파일 서브시스템

"각 프로세스는 시스템 호출을 통해 커널의 기능을 사용하며..." 앞에서 본 것 같죠. 생소하면 다시 한번 공부하고 오세요.

▶ 정답: 1. ② 2. ④ 3. ② 4. ① 5. ② 6. ①



## 전문가의 조언

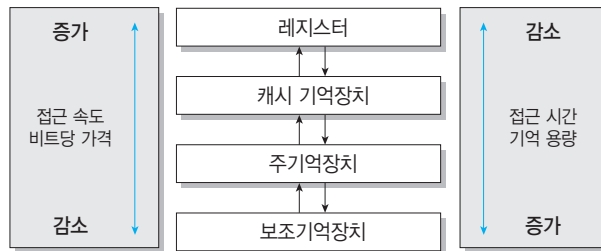
운영체제가 기억장치를 어떻게 관리하는지를 배우기 전에 기억장치 계층 구조의 특징을 파악하면 각 섹션을 공부하는 데 도움이 됩니다.

## 전문가의 조언

기억장치의 관리 전략에는 반입(Fetch), 배치(Placement), 교체(Replacement) 전략이 있다는 것과 각각의 의미, 그리고 배치 전략의 종류를 알아두세요.

## 1 기억장치 계층 구조의 특징

기억장치는 레지스터, 캐시 기억장치, 주기억장치, 보조기억장치를 다음과 같이 계층 구조로 분류할 수 있다.



- 계층 구조에서 상위의 기억장치일수록 접근 속도와 접근 시간이 빠르지만, 기억 용량이 적고 고가이다.
- 주기억장치는 각기 자신의 주소를 갖는 워드 또는 바이트들로 구성되어 있으며, 주소를 이용하여 액세스할 수 있다.
- 레지스터, 캐시 기억장치, 주기억장치의 프로그램과 데이터는 CPU가 직접 액세스 할 수 있으나 보조기억장치에 있는 프로그램이나 데이터는 직접 액세스할 수 없다.
- 보조기억장치에 있는 데이터는 주기억장치에 적재된 후 CPU에 의해 액세스될 수 있다.

## 2 기억장치의 관리 전략의 개요

기억장치의 관리 전략은 보조기억장치의 프로그램이나 데이터를 주기억장치에 적재시키는 시기, 적재 위치 등을 지정하여 한정된 주기억장치의 공간을 효율적으로 사용하기 위한 것으로 반입(Fetch) 전략, 배치(Placement) 전략, 교체(Replacement) 전략이 있다.

## 3 반입(Fetch) 전략

반입 전략은 보조기억장치에 보관중인 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지를 결정하는 전략이다.

- **요구 반입(Demand Fetch)** : 실행중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재하는 방법이다.
- **예상 반입(Anticipatory Fetch)** : 실행중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법이다.

## 4 배치(Placement) 전략

배치 전략은 새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략이다.

- **최초 적합(First Fit)** : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법
- **최적 적합(Best Fit)** : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화\*를 가장 작게 남기는 분할 영역에 배치시키는 방법
- **최악 적합(Worst Fit)** : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법

**예제** 기억장치 상태가 다음 표와 같다. 기억장치 관리 전략으로 First Fit, Best Fit, Worst Fit 방법을 사용하려 할 때, 각 방법에 대하여 10K의 프로그램이 할당받게 되는 영역의 번호는?

영역 번호	영역 크기	상태
1	5K	공백
2	14K	공백
3	10K	사용 중
4	12K	공백
5	16K	공백

- ① 먼저 10K가 적재될 수 있는지 각 영역의 크기를 확인한다.
- ② First Fit : 빈 영역 중에서 10K의 프로그램이 들어갈 수 있는 첫 번째 영역은 2번이다.
- ③ Best Fit : 빈 영역 중에서 10K 프로그램이 들어가고 단편화를 가장 작게 남기는 영역은 4번이다.
- ④ Worst Fit : 빈 영역 중에서 10K 프로그램이 들어가고 단편화를 가장 많이 남기는 영역은 5번이다.

## 5 교체(Replacement) 전략

교체 전략은 주기억장치의 모든 영역이 이미 사용중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략이다.

- 교체 전략에는 FIFO, OPT, LRU, LFU, NUR, SCR 등이 있다.

### 단편화

단편화는 주기억장치의 분할된 영역에 프로그램이나 데이터를 할당할 경우, 분할된 영역이 프로그램이나 데이터보다 작거나 커서 생기는 빈 기억 공간을 의미합니다.



### 전문가의 조언

주기억장치와 가상기억장치 관리 전략은 그 의미와 방식이 동일하게 사용됩니다. 교체 전략은 주로 가상기억장치와 함께 사용됩니다. 교체 전략에서 사용되는 기법은 648쪽에서 자세히 배웁니다.



## 기출문제 따라잡기

Section 144

이전기출

1. 기억장치 관리 전략은 주기억장치 자원을 가장 잘 사용하도록 설계되어야 하는데, 다음 중 주기억장치 관리 전략과 거리가 먼 것은?

- ① Fetch 전략
- ② Placement 전략
- ③ Paging 전략
- ④ Replacement 전략

주기억장치 관리 전략에는 반입(Fetch), 배치(Placement), 교체(Replacement) 전략이 있다는 것을 알아두라고 했죠! Paging 전략은 가상기억장치에서 사용되는 기법입니다.

이전기출

2. 기억장치의 관리 전략 중 반입(Fetch) 전략의 설명으로 맞는 것은?

- ① 프로그램/데이터를 주기억장치로 가져오는 시기를 결정하는 전략
- ② 프로그램/데이터의 주기억장치 내의 위치를 정하는 전략
- ③ 주기억장치 내의 빈 공간 확보를 위해 제거할 프로그램/데이터를 선택하는 전략
- ④ 프로그램/데이터의 위치를 이동시키는 전략

기억장치 관리 전략의 종류뿐만 아니라 의미도 정확히 알아야 한다고 했죠? ②번은 배치 전략, ③번은 교체 전략을 의미합니다.

이전기출

3. 새로 들어온 프로그램과 데이터를 주기억장치 내의 어디에 놓을 것인가를 결정하기 위한 주기억장치 배치 전략에 해당하지 않는 것은?

- ① Best Fit
- ② Worst Fit
- ③ First Fit
- ④ Last Fit

배치 전략에는 최초(First), 최적(Best), 최악(Worst)이 있지요! 각 단어의 의미를 생각하면 이것이 배치 전략의 종류라는 것을 쉽게 구분할 수 있습니다.

이전기출

4. 저장장치의 배치 전략 중에서 작업의 배치 결정을 가장 빨리 내릴 수 있는 방식은?

- ① Best Fit
- ② First Fit
- ③ Worst Fit
- ④ Fast Fit

이 문제를 보면서 무엇일까 고민하고 계신 분 있나요? 단순히 생각하면 됩니다. 최적이나 최악 적합은 단편화 크기 때문에 각 영역을 일일이 계산해 봐야 하잖아요. 하지만 최초 적합은 프로그램이 들어갈 수 있는 영역 중 첫 번째 영역에 배치하는 것이니 당연히 빠를 수 밖에요.

이전기출

5. 주기억장치 관리 기법인 First Fit, Best Fit, Worst Fit 방법에 대해서 10K 프로그램이 할당될 부분으로 옳은 것은 어느 것인가?

영역1	9K
영역2	15K
영역3	10K
영역4	30K

- ① 2-3-4
- ② 2-2-3
- ③ 2-3-2
- ④ 2-1-4

First Fit은 프로그램이 들어갈 수 있는 크기 중 처음 영역, Best Fit은 단편화가 가장 적어 최적인 영역, Worst Fit은 단편화가 가장 커서 최악인 영역입니다.

이전기출

6. 그림과 같이 저장장치가 배치되어 있을 때 13K의 작업이 공간의 할당을 요구하여 최악 적합(Worst-Fit) 전략을 사용한다면 어느 주소에 배치되는가?

a	OS 사용 공간
b	16K 공백
c	사용중
d	14K 공백
e	사용중
f	5K 공백
g	사용중
h	30K 공백

- ① b
- ② d
- ③ f
- ④ h

최악 적합은 단편화가 가장 많이 남은 영역에 할당하는 것이죠? 사용되지 않은 영역 중 크기가 가장 큰 영역을 찾아보세요.

이전기출

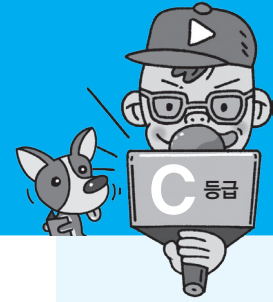
7. 가상기억장치의 페이지 교체(Replacement) 알고리즘이 아닌 것은?

- ① FIFO(First In First Out)
- ② LRU(Least Recently Used)
- ③ SSTF(Shortest Seek Time First)
- ④ LFU(Least Frequently Used)

페이지 교체 알고리즘에는 OPT, FIFO, LRU, LFU, NUR, SCR 등이 있습니다.

▶ 정답 : 1. ③ 2. ① 3. ④ 4. ② 5. ① 6. ④ 7. ③





## 1 주 기억 장치 할당의 개념

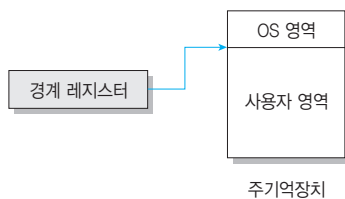
주 기억 장치 할당 기법은 프로그램이나 데이터를 실행시키기 위해 주 기억 장치에 어떻게 할당할 것인지에 대한 내용이며, 연속 할당 기법과 분산 할당 기법으로 분류할 수 있다.

연속 할당 기법	<p>프로그램을 주 기억 장치에 연속으로 할당하는 기법으로, 단일 분할 할당 기법과 다중 분할 할당 기법이 있다.</p> <ul style="list-style-type: none"> <li>• 단일 분할 할당 기법 : 오버레이, 스와핑</li> <li>• 다중 분할 할당 기법 : 고정 분할 할당 기법, 동적 분할 할당 기법</li> </ul>
분산 할당 기법*	<p>프로그램을 특정 단위의 조각으로 나누어 주 기억 장치 내에 분산하여 할당하는 기법으로, 페이징 기법과 세그멘테이션 기법으로 나눌 수 있다.</p>

## 2 단일 분할 할당 기법

단일 분할 할당 기법은 주 기억 장치를 운영체제 영역과 사용자 영역으로 나누어 한 순간에는 오직 한 명의 사용자만이 주 기억 장치의 사용자 영역을 사용하는 기법이다.

- 가장 단순한 기법으로 초기의 운영체제에서 많이 사용하던 기법이다.
- 운영체제를 보호하고, 프로그램이 사용자 영역만을 사용하기 위해 운영체제 영역과 사용자 영역을 구분하는 경계 레지스터(Boundary Register)\*가 사용된다.
- 프로그램의 크기가 작을 경우 사용자 영역이 낭비될 수 있다.
- 초기에는 주 기억 장치보다 큰 사용자 프로그램은 실행할 수 없었으나 오버레이 기법을 사용하면서 이 문제가 해결되었다.



### 오버레이(Overlay) 기법\*

오버레이 기법은 주 기억 장치보다 큰 사용자 프로그램을 실행하기 위한 기법이다.

- 보조 기억 장치에 저장된 하나의 프로그램을 여러 개의 조각으로 분할한 후 필요한 조각을 차례로 주 기억 장치에 적재하여 프로그램을 실행한다.
- 프로그램이 실행되면서 주 기억 장치의 공간이 부족하면 주 기억 장치에 적재된 프로

### 전문가의 조언

고정 분할 할당과 동적 분할 할당 기법으로 나누는 것은 다중 분할 할당 기법이며, 이것은 연속 할당 기법에 해당된다는 것을 기억하세요.

### 분산 할당 기법

분산 할당 기법은 가상 기억 장치의 내용을 주 기억 장치에 할당하기 위한 기법으로, 가상 기억 장치 관리 기법이라고도 합니다. Section 146 가상 기억 장치 구현 기법에서 자세히 다룹니다.

### 전문가의 조언

단일 분할 할당 기법의 개념과 특징을 숙지하고, 오버레이 기법과 스와핑 기법은 각각의 특징을 구분할 수 있도록 정리하세요.

### 경계 레지스터 (Boundary Register)

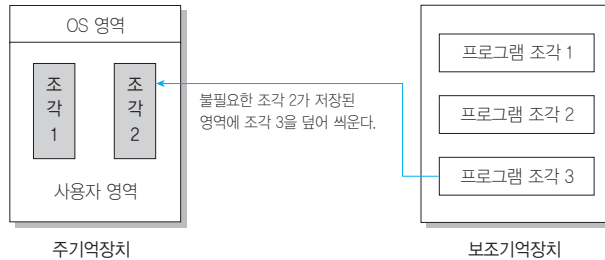
경계 레지스터는 사용자 영역에 있는 사용자 프로그램이 운영체제 영역에 접근하지 못하도록 보호하는 레지스터로, 사용자 영역이 시작되는 주소를 기억하고 있습니다.

### 오버레이 기법이 가능한 이유

하나의 프로그램을 여러 개의 조각으로 분할하여 처리할 수 있는 것은 프로그램의 모든 부분이 동시에 실행되는 것이 아니기 때문입니다. 예를 들어 한글을 사용할 때 한글의 모든 기능을 동시에 사용하는 것이 아니라 파일을 여는 기능, 저장하는 기능, 색을 지정하는 기능 등이 분할되어 따로 실행될 수 있습니다.

그램의 조각 중 불필요한 조각이 위치한 장소에 새로운 프로그램의 조각을 중첩(Overlay)하여 적재한다.

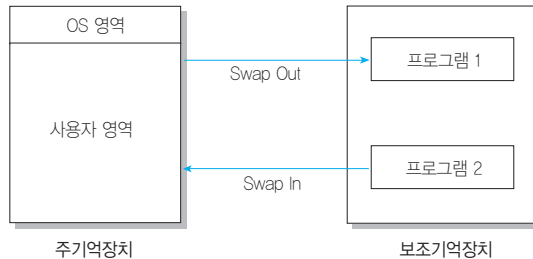
- 프로그램을 여러 개의 조각으로 분할하는 작업은 프로그래머가 수행해야 하므로 프로그래머는 시스템 구조나 프로그램 구조를 알아야 한다.



### 스와핑(Swapping) 기법

스와핑 기법은 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다 필요에 따라 다른 프로그램과 교체하는 기법이다.

- 주기억장치에 있는 프로그램이 보조기억장치로 이동되는 것을 Swap Out, 보조기억장치에 있는 프로그램이 주기억장치로 이동되는 것을 Swap In이라고 한다.
- 하나의 사용자 프로그램이 완료될 때까지 교체 과정을 여러 번 수행할 수 있다.
- 가상기억장치의 페이징 기법으로 발전되었다.



#### 전문가의 조언

고정 분할 할당 기법에 대해서는 프로그램 전체가 주기억장치에 위치해야 한다는 것과 주기억장치 크기를 고정적으로 분할한다는 것을 기억하고, 가변 분할 할당 기법에 대해서는 주기억장치를 프로그램에 맞게 분할한다는 것을 기억해 두세요.

#### 내부 단편화 및 외부 단편화

- **내부 단편화** : 분할된 영역이 할당될 프로그램의 크기보다 크기 때문에 프로그램이 할당된 후 사용되지 않고 남아 있는 빈 공간
- **외부 단편화** : 분할된 영역이 할당될 프로그램의 크기보다 작기 때문에 프로그램이 할당될 수 없어 사용되지 않고 빈 공간으로 남아 있는 분할된 전체 영역

## 3 다중 분할 할당 기법

### 고정 분할 할당(Multiple contiguous Fixed parTition allocation, MFT) 기법

#### = 정적 할당(Static Allocation) 기법

고정 분할 할당은 프로그램을 할당하기 전에 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할하고 준비상태 큐에서 준비중인 프로그램을 각 영역에 할당하여 수행하는 기법이다.

- 프로그램을 실행하려면 프로그램 전체가 주기억장치에 위치해야 한다.
- 프로그램이 분할된 영역보다 커서 영역 안에 들어갈 수 없는 경우가 발생할 수 있다.
- 일정한 크기의 분할 영역에 다양한 크기의 프로그램이 할당되므로 내부 단편화 및 외부 단편화\*가 발생하여 주기억장치의 낭비가 많다.

- 실행할 프로그램의 크기를 미리 알고 있어야 한다.
- 다중 프로그래밍을 위해 사용되었으나 현재는 사용되지 않는다.

잠깐만요

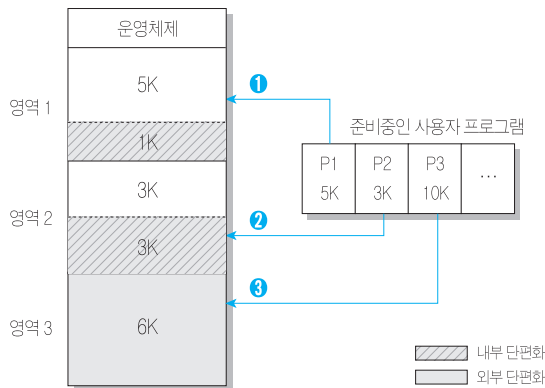


#### 절대 번역과 적재, 재배치 번역과 적재

고정 분할 할당 기법은 다음과 같이 절대 번역과 적재, 재배치 번역과 적재로 구분할 수 있습니다.

- **절대 번역과 적재** : 프로그램이 할당될 분할 영역을 어셈블러나 컴파일러가 지정하는 방식으로, 각 프로그램은 분할된 각 영역의 준비상태 큐에서 기다리며, 다른 분할 영역이 비어 있다 하더라도 지정된 분할 영역만을 사용해야 합니다.
- **재배치 번역과 적재** : 프로그램이 할당될 영역이 미리 지정되지 않고, 하나의 준비상태 큐에서 기다린 순서대로 분할 영역에 할당되는 방식입니다.

**예제** 18K 크기의 사용자 영역이 6K씩 3개의 영역으로 분할된 주기억장치에 준비중인 5K, 3K, 10K 프로그램을 고정 분할 기법으로 할당하면 다음과 같다.



- ① 준비상태 큐에서 대기중인 P1(5K)를 영역 1에 할당시키면 1K의 내부 단편화가 발생한다.
- ② 준비상태 큐에서 대기중인 P2(3K)를 영역 2에 할당시키면 3K의 내부 단편화가 발생한다.
- ③ 준비상태 큐에서 대기중인 P3(10K)를 영역 3에 할당시키면 6K의 외부 단편화가 발생한다.

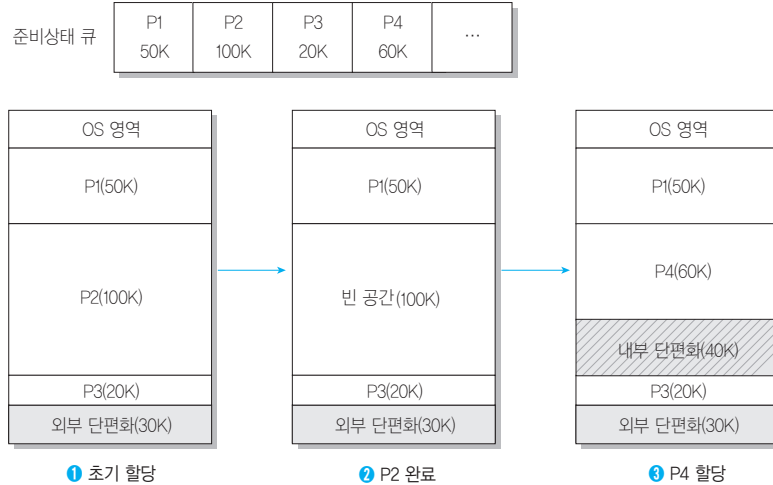
#### 가변 분할 할당(Multiple contiguous Variable parTition allocation, MVT) 기법

##### = 동적 할당(Dynamic Allocation) 기법

고정 분할 할당 기법의 단편화를 줄이기 위한 것으로, 미리 주기억장치를 분할해 놓는 것이 아니라 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할하는 기법이다.

- 주기억장치를 효율적으로 사용할 수 있으며, 다중 프로그래밍의 정도를 높일 수 있다.
- 고정 분할 할당 기법에 비해 실행될 프로세스 크기에 대한 제약이 적다.
- 단편화를 상당 부분 해결할 수 있으나 영역과 영역 사이에 단편화가 발생될 수 있다.

**예제** 200K 크기의 사용자 영역으로 구성된 주기억장치에 준비중인 다음과 같은 프로그램들을 가변 분할 할당 기법으로 할당하면 다음과 같다.



- ❶ P1, P2, P3이 차례로 주기억장치의 사용자 영역에 할당되며 30K(200-50-100-20)의 외부 단편화가 발생된다.
- ❷ P2의 수행이 완료되면 100K의 빈 공간이 생긴다.
- ❸ 준비상태 큐에서 기다리던 P4는 P2가 사용했던 공간에 할당되며 40K(100-60)의 내부 단편화가 발생된다.



### 기출문제 따라잡기

Section 145

**이전기술**

1. 주기억장치의 사용자 영역을 일정 수의 고정된 크기로 분할하여 준비상태 큐에서 준비 중인 프로그램을 각 영역에 할당하여 수행하는 기법은?

- ❶ 가변 분할 기억장치 할당
- ❷ 고정 분할 기억장치 할당
- ❸ 교체 기법
- ❹ 오버레이 기법

문제에 답이 숨어 있네요. 잘 찾아보세요. “고정된 크기로 분할하여 ~ 각 영역에 할당하여~”

**이전기술**

2. 실기억장치를 사용하는 시스템에서 주기억장치를 고정된 크기로 분할하여 사용하는 경우에 대한 설명 중 가장 거리가 먼 것은?

- ❶ 내부 조각(Internal Fragmentation) 현상이 발생한다.
- ❷ 주기억장치와 보조기억장치 간에 데이터의 이동이 빈번히 발생한다.

- ❸ 프로그램이 주어진 분할 안에 다 들어갈 수 없는 경우가 생길 수 있다.
- ❹ 프로그램이 실행되기 위해서는 그 전체가 주기억장치에 위치해야 한다.

고정 분할 할당 기법은 실행할 프로그램 전체가 주기억장치에 적재된 후 실행하기 때문에 주기억장치와 보조기억장치 간에 데이터 이동은 자주 발생하지 않습니다.

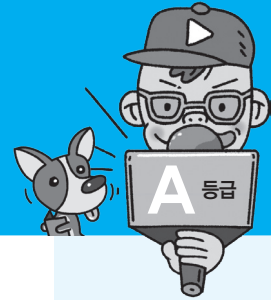
**이전기술**

3. 다음 기억공간 관리 중 고정 분할 할당과 동적 분할 할당으로 나누어 관리되는 기법은?

- ❶ 연속 로딩 기법
- ❷ 분산 로딩 기법
- ❸ 페이징(Paging)
- ❹ 세그먼트(Segment)

고정 분할 할당과 동적 분할 할당은 다중 분할 할당 기법에 해당된다고 했죠? 그리고 다중 분할 할당 기법은 어느 할당 기법에 해당된다고 했죠? 로딩(Loding)을 한글로 번역하면 ‘할당’입니다.

▶ 정답 : 1. ❷ 2. ❷ 3. ❶



## 1 가상기억장치의 개요

가상기억장치는 보조기억장치(하드디스크)의 일부를 주기억장치처럼 사용하는 것으로, 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법이다.

- 프로그램을 여러 개의 작은 블록\* 단위로 나누어서 가상기억장치에 보관해 놓고, 프로그램 실행 시 요구되는 블록만 주기억장치에 불연속적으로 할당하여 처리한다.
- 주기억장치의 용량보다 큰 프로그램을 실행하기 위해 사용한다.
- 주기억장치의 이용률과 다중 프로그래밍의 효율을 높일 수 있다.
- 가상기억장치에 저장된 프로그램을 실행하려면 가상기억장치의 주소를 주기억장치의 주소로 바꾸는 주소 변환\* 작업이 필요하다.
- 블록 단위로 나누어 사용하므로 연속 할당 방식에서 발생할 수 있는 단편화를 해결할 수 있다.
- 가상기억장치의 일반적인 구현 방법에는 블록의 종류에 따라 페이징 기법과 세그먼테이션 기법으로 나눌 수 있다.

## 2 페이징(Paging) 기법

페이징 기법은 가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나뉜 프로그램(페이지)을 동일하게 나뉜 주기억장치의 영역(페이지 프레임)에 적재시켜 실행하는 기법이다.

- 프로그램을 일정한 크기로 나눈 단위를 페이지(Page)\*라고 하고, 페이지 크기로 일정하게 나뉜 주기억장치의 단위를 페이지 프레임(Page Frame)이라고 한다.
- 외부 단편화는 발생하지 않으나 내부 단편화는 발생할 수 있다.\*
- 주소 변환을 위해서 페이지의 위치 정보를 가지고 있는 페이지 맵 테이블(Page Map Table)이 필요하다.
- 페이지 맵 테이블 사용으로 비용이 증가되고, 처리 속도가 감소된다.

## 3 세그먼테이션(Segmentation) 기법

세그먼테이션 기법은 가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 기법이다.

- 프로그램을 배열이나 함수 등과 같은 논리적인 크기로 나눈 단위를 세그먼트(Segmentation)라고 하며, 각 세그먼트는 고유한 이름과 크기를 갖는다.
- 기억장치의 사용자 관점을 보존하는 기억장치 관리 기법이다.

### 전문가의 조언

가상기억장치의 특징을 파악해 두고, 가상기억장치 구현 기법에는 페이징 기법과 세그먼테이션 기법이 있다는 것을 기억하세요.

#### 블록

블록은 보조기억장치와 주기억장치 간에 전송되는 데이터의 최소 단위입니다.

#### 주소 변환

주소 변환은 가상기억장치에 있는 프로그램이 주기억장치에 적재되어 실행될 때 논리적인 가상주소를 물리적인 실기억주소로 변환하는 것으로, 주소 사상 또는 주소 매핑(Mapping)이라고도 합니다. 이때 연속적인 가상주소가 반드시 연속적인 실기억주소로 변환되지 않아도 되는데, 이를 인위적 연속성(Artificial Contiguity)이라고 합니다.

### 전문가의 조언

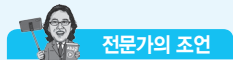
페이징 기법과 세그먼테이션 기법의 가장 큰 차이점은, 페이징 기법에서 사용하는 페이지 크기가 동일하지만, 세그먼테이션 기법에서 사용하는 세그먼트는 크기가 일정하지 않다는 것입니다. 이로 인해 발생할 수 있는 차이점들은 어떤 것들이 있는지 파악해 두세요.

#### 페이지의 크기

일반적으로 페이지의 크기는 1~4KB입니다.

#### 내부 단편화는 발생할 수 있다?

페이지 크기가 4KB이고, 사용할 프로그램이 17KB라면 프로그램은 페이지 단위로 4KB씩 나누어지게 됩니다. 이때 마지막 페이지의 실제 용량은 1KB(17KB-16KB)가 되고, 이것이 주기억장치에 적재되면 3KB의 내부 단편화가 발생합니다.



페이지 교체 알고리즘들의 개별적인 특징과 교체 원리를 학습하세요. 특히 FIFO와 LRU의 알고리즘은 페이지 부재수를 구하는 방법을 확실히 숙지하세요. 영문 약어를 풀어서 이해하면 기억하기 쉽습니다.

#### 페이지 부재(Page Fault)

페이지 부재는 CPU가 액세스한 가상 페이지가 주기억장치에 없는 경우를 말합니다. 페이지 부재가 발생하면 해당 페이지를 디스크에서 주기억장치로 가져와야 합니다.

- 세그먼테이션 기법을 이용하는 궁극적인 이유는 기억공간을 절약하기 위해서이다.
- 주소 변환을 위해서 세그먼트가 존재하는 위치 정보를 가지고 있는 세그먼트 맵 테이블(Segment Map Table)이 필요하다.
- 세그먼트가 주기억장치에 적재될 때 다른 세그먼트에게 할당된 영역을 침범할 수 없으며, 이를 위해 기억장치 보호키(Storage Protection Key)가 필요하다.
- 내부 단편화는 발생하지 않으나 외부 단편화는 발생할 수 있다.

## 4 페이지 교체 알고리즘

페이지 교체 알고리즘은 페이지 부재(Page Fault)\*가 발생했을 때 가상기억장치의 필요한 페이지를 주기억장치에 적재해야 하는데, 이때 주기억장치의 모든 페이지 프레임이 사용중이면 어떤 페이지 프레임을 선택하여 교체할 것인지를 결정하는 기법이다.

- 페이지 교체 알고리즘에는 OPT, FIFO, LRU, LFU, NUR, SCR 등이 있다.

### OPT(OPTimal replacement, 최적 교체)

- OPT는 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법이다.
- 벨레이디(Belady)가 제안한 것으로, 페이지 부재 횟수가 가장 적게 발생하는 가장 효율적인 알고리즘이다.

### FIFO(First In First Out)

- FIFO는 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법이다.
- 이해하기 쉽고, 프로그래밍 및 설계가 간단하다.

**예제** 다음의 참조 페이지를 세 개의 페이지 프레임에 가진 기억장치에서 FIFO 알고리즘을 사용하여 교체했을 때 페이지 부재의 수는? (단, 초기 페이지 프레임은 모두 비어 있는 상태이다.)

참조 페이지	2	3	2	1	5	2	3	5
페이지 프레임	2	2	2	2	5	5	5	5
		3	3	3	3	2	2	2
				1	1	1	3	3
부재 발생	●	●		●	●	●	●	

부재 수 = 6

①                      ②                      ③

- ① 참조 페이지를 각 페이지 프레임에 차례로 적재시키되 이미 적재된 페이지는 해당 위치의 페이지 프레임을 사용한다.
- ② 사용할 페이지 프레임이 없을 경우 가장 먼저 들어와서 오래 있었던 페이지 2를 제거한 후 5를 적재한다.
- ③ 그 다음에 적재된 페이지 3을 제거한 후 2를 적재하며, 같은 방법으로 나머지 참조 페이지를 수행한다.

### LRU(Least Recently Used)

- LRU는 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법이다.
- 각 페이지마다 계수기(Counter)\*나 스택(Stack)을 두어 현 시점에서 가장 오랫동안 사용하지 않은, 즉 가장 오래 전에 사용된 페이지를 교체한다.

**예제** 다음의 참조 페이지를 세 개의 페이지 프레임에 가진 기억장치에서 LRU 알고리즘을 사용하여 교체했을 때 페이지 부재의 수는? (단, 초기 페이지 프레임은 모두 비어 있는 상태이다.)

참조 페이지	2	3	2	1	5	2	3	5
페이지 프레임	2	2	2	2	2	2	2	2
		3	3	3	5	5	5	5
				1	1	1	3	3
부재 발생	●	●		●	●		●	

부재 수 = 5

- 1 참조 페이지를 각 페이지 프레임에 차례로 적재시키되 이미 적재된 페이지는 해당 위치의 페이지 프레임을 사용한다.
- 2 사용할 페이지 프레임이 없을 경우 현재 시점에서 가장 오랫동안 사용되지 않은 페이지 3을 제거한 후 5를 적재한다.
- 3 같은 방법으로 나머지 참조 페이지를 수행한다.

### LFU(Least Frequently Used)

- LFU는 사용 빈도가 가장 적은 페이지를 교체하는 기법이다.
- 활발하게 사용되는 페이지는 사용 횟수가 많아 교체되지 않고 사용된다.

### NUR(Not Used Recently)

- NUR은 LRU와 비슷한 알고리즘으로, 최근에 사용하지 않은 페이지를 교체하는 기법이다.
- 최근에 사용되지 않은 페이지는 향후에도 사용되지 않을 가능성이 높다는 것을 전제로, LRU에서 나타나는 시간적인 오버헤드를 줄일 수 있다.
- 최근의 사용 여부를 확인하기 위해서 각 페이지마다 두 개의 비트, 즉 참조 비트(Reference Bit)\*와 변형 비트(Modified Bit, Dirty Bit)\*가 사용된다.
- 다음과 같이 참조 비트와 변형 비트의 값에 따라 교체될 페이지의 순서가 결정된다.

참조 비트	0	0	1	1
변형 비트	0	1	0	1
교체 순서	1	2	3	4

### SCR(Second Chance Replacement, 2차 기회 교체)

SCR은 가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로, FIFO 기법의 단점을 보완하는 기법이다.

#### 계수기(Counter)

계수기는 각 페이지별로 존재하는 논리적 시계(Logical Clock)로, 해당 페이지가 사용될 때마다 0으로 클리어 시킨 후 시간을 증가시켜서 시간이 가장 오래된 페이지를 교체합니다.

#### 참조 비트(Reference Bit)

참조 비트는 페이지가 호출되었을 때는 0, 호출되었을 때는 1로 지정됩니다.

#### 변형 비트(Modified Bit, Dirty Bit)

변형 비트는 페이지 내용이 변경되지 않았을 때는 0, 변경되었을 때는 1로 지정됩니다.





이전기출

**1. 가상기억장치에 대한 설명으로 옳지 않은 것은?**

- ① 연속 배당 방식에서의 기억장소 단편화 문제를 적극적으로 해결할 수 있다.
- ② 기억장치의 이용률과 다중 프로그래밍의 효율을 높일 수 있다.
- ③ 가상기억장치의 일반적인 구현 방법에는 페이징 기법과 세그먼테이션 기법이 있다.
- ④ 주기억장소의 물리적 공간보다 큰 프로그램은 실행될 수 없다.

가상기억장치를 사용하는 주된 이유는 주기억장치보다 큰 프로그램을 수행하기 위해서라는 것 잊으면 안 됩니다.

이전기출

**2. 페이징 기법과 세그먼테이션 기법에 대한 설명으로 옳지 않은 것은?**

- ① 페이징 기법에서는 주소 변환을 위한 페이지 맵 테이블이 필요하다.
- ② 프로그램을 일정한 크기로 나눈 단위를 페이지라고 한다.
- ③ 세그먼테이션 기법에서는 하나의 작업을 크기가 각각 다른 여러 논리적인 단위로 나누어 사용한다.
- ④ 세그먼테이션 기법에서는 내부 단편화가, 페이징 기법에서는 외부 단편화가 발생할 수 있다.

페이징 기법에서는 내부 단편화, 세그먼테이션 기법에서는 외부 단편화가 발생할 수 있다는 것 알고 있죠? 모르면 다시 한 번 공부하고 오세요.

이전기출

**3. 다음 설명의 (A)와 (B)에 들어갈 내용으로 옳은 것은?**

가상기억장치의 일반적인 구현 방법에는 프로그램을 고정된 크기의 일정한 블록으로 나누는 (A) 기법과 가변적인 크기의 블록으로 나누는 (B) 기법이 있다.

- ① (A) : Virtual Address (B) : Paging
- ② (A) : Paging (B) : Segmentation
- ③ (A) : Segmentation (B) : Fragmentation
- ④ (A) : Segmentation (B) : Compaction

가상기억장치의 두 가지 구현 기법을 기억하고 있죠? 고정된 크기면 페이징, 가변적 크기면 세그먼테이션 잊지마세요.

이전기출

**4. 세그먼테이션 기법에 대한 설명으로 옳지 않은 것은?**

- ① 각 세그먼트는 고유한 이름과 크기를 갖는다.
- ② 세그먼트 맵 테이블이 필요하다.
- ③ 프로그램을 일정한 크기로 나눈 단위를 세그먼트라고 한다.
- ④ 기억장치 보호키가 필요하다.

2, 3번 문제와 크게 다르지 않죠? 세그먼트는 가변적인 크기입니다.

이전기출

**5. 가상기억장치 구현에서 세그먼테이션(Segmentation) 기법의 설명으로 옳지 않은 것은?**

- ① 페이지 맵 테이블(Page Map Table)이 필요하다.
- ② 세그먼테이션은 프로그램을 여러 개의 블록으로 나누어 수행한다.
- ③ 각 세그먼트는 고유한 이름과 크기를 갖는다.
- ④ 기억장치 보호키가 필요하다.

가상기억장치 구현 기법에는 페이징 기법과 세그먼테이션 기법이 있죠! 페이징 기법에는 페이지 맵 테이블이, 세그먼테이션 기법에는 세그먼트 맵 테이블이 필요합니다. 혼동하지 마세요.

이전기출

**6. 페이징 기법에 대한 설명으로 옳지 않은 것은?**

- ① 동적 주소 변환 기법을 사용하여 다중 프로그래밍의 효과를 증진시킨다.
- ② 내부 단편화가 발생하지 않는다.
- ③ 프로그램을 동일한 크기로 나눈 단위를 페이지라고 하며, 이 페이지를 블록으로 사용하는 기법이다.
- ④ 페이지 맵 테이블이 필요하다.

2번 문제를 풀었다면 어렵지 않은 문제죠? 페이징 기법에서는 내부 단편화가 발생할 수 있다는 것 잊지마세요!

이전기출

**7. 가상기억장치에서 주기억장치로 페이지를 옮겨 넣을 때 주소를 조정해 주어야 하는데 이를 무엇이라 하는가?**

- ① 매핑(Mapping)                      ② 스케줄링(Scheduling)
- ③ 매칭(Matching)                    ④ 로딩>Loading)

가상주소로부터 물리주소를 찾는 것을 사상(Mapping)이라고 합니다.





## 기출문제 따라잡기

Section 146

이전기출

### 8. 가상기억장치에 대한 설명 중 옳지 않은 것은?

- ① 동적 주소 변환(DAT) 기법은 프로세스가 수행될 때 가상주소를 실주소로 바꾸어 준다.
- ② 크기가 고정된 블록을 페이지라 하며, 크기가 변할 수 있는 블록을 세그먼트라 한다.
- ③ 인위적 연속성(Artificial Contiguity)이란 가상주소 공간상의 연속적인 주소가 주기억장치에서도 인위적으로 연속성을 보장해야 하는 성질을 말한다.
- ④ 세그먼트 기법에서 한 프로세스의 세그먼트들은 동시에 모두 기억장치 내에 있을 필요가 없으며, 연속적일 필요도 없다.

인위적 연속성의 의미를 떠올려보세요. 인위적 연속성은 가상주소를 실주소로 바꾸어 줄 때 프로세스가 갖는 가상주소 공간 상의 연속적인 주소가 실저장장치에서 연속적일 필요가 없다는 성질을 의미합니다.

이전기출

### 9. 요구 페이지 기법 중 가장 오랫동안 사용되지 않았던 페이지를 먼저 교체하는 기법에 해당되는 것은?

- ① FIFO
- ② LFU
- ③ LRU
- ④ NUR

FIFO(First In First Out)는 먼저 들어온 것, LFU(Least Frequently Used)는 사용 횟수가 적은 것, NUR(Not Used Recently)은 최근에 사용되지 않은 것 쉽게 구분할 수 있겠지요.

이전기출

### 10. 선입선출(FIFO) 교체 알고리즘을 사용하고 참조하는 페이지 번호의 순서는 다음과 같다. 할당된 페이지 프레임의 수가 4개이고, 이들 페이지 프레임은 모두 비어 있다고 가정할 경우 몇 번의 페이지 부재가 발생하는가?

참조 페이지 번호 : 0 1 2 3 0 1 4 0 1 2 3 4

- ① 7
- ② 8
- ③ 9
- ④ 10

참조 페이지가 페이지 프레임에 없을 경우 페이지 결함(부재)이 발생합니다. 초기에는 모든 페이지 프레임이 다 비어 있으므로 처음 0, 1, 2, 3 페이지 적재 시 페이지 결함이 발생합니다. FIFO는 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법이므로, 참조 페이지 4를 참조할 때에는 0을 제거한 후 4를 가져오게 됩니다. 이와 같은 방식으로 모든 페이지 요청을 처리하고 나면 총 페이지 결함 발생 수는 10번입니다.

참조 페이지	0	1	2	3	0	1	4	0	1	2	3	4
페이지 프레임	0	0	0	0	0	0	4	4	4	4	3	3
		1	1	1	1	1	1	0	0	0	0	4
			2	2	2	2	2	2	1	1	1	1
				3	3	3	3	3	3	2	2	2
부재 발생	●	●	●	●			●	●	●	●	●	●

이전기출

### 11. 주기억장치에 완전히 비어 있는 3개의 페이지가 있다. 페이지 교체 방법으로 LRU를 사용할 때 요청된 페이지 번호의 순서가 0, 1, 2, 3, 0, 1, 4, 0 인 경우 페이지 부재(Fault)는 몇 번 발생하는가?

- ① 5
- ② 6
- ③ 7
- ④ 8

10번과 비슷한 문제죠! LRU 기법을 사용하여 페이지를 참조하는 그림은 아래와 같습니다.

참조 페이지	0	1	2	3	0	1	4	0
페이지 프레임	0	0	0	3	3	3	4	4
		1	1	1	0	0	0	0
			2	2	2	1	1	1
부재 발생	●	●	●	●	●	●	●	

이전기출

### 12. 페이지 교체 기법 중 매 페이지마다 두 개의 하드웨어 비트, 즉 참조 비트와 변형 비트가 필요한 기법은?

- ① FIFO
- ② LRU
- ③ LFU
- ④ NUR

'하드웨어 비트가 필요한 기법'하면 'NUR'입니다. 참조 비트와 변형 비트가 필요하죠.

이전기출

### 13. NUR 기법은 호출 비트와 변형 비트를 가진다. 다음 중 가장 나중에 교체될 페이지는?

- ① 호출 비트 : 0, 변형 비트 : 0
- ② 호출 비트 : 0, 변형 비트 : 1
- ③ 호출 비트 : 1, 변형 비트 : 0
- ④ 호출 비트 : 1, 변형 비트 : 1

참조(호출)비트와 변형비트에 따른 교체 순서를 기억하라고 했죠? NUR 기법은 최근에 사용되지 않은 페이지를 교체하는 것이므로, 최근에 참조되고 변형된 것을 찾으면 되겠네요.

▶ 정답 : 1. ④ 2. ④ 3. ② 4. ③ 5. ① 6. ② 7. ① 8. ③ 9. ③ 10. ④ 11. ③ 12. ④ 13. ④



## 전문가의 조언

페이지 크기가 작으면 페이지의 수는 늘어나고, 페이지 크기가 크면 페이지의 수는 줄어듭니다. 이 점을 염두에 두고 각각의 특징을 읽어보세요.



## 전문가의 조언

Locality의 개념을 파악하고, 시간 구역성과 공간 구역성에 해당하는 기억장소를 정확하게 구분할 수 있어야 합니다.

## 1 페이지 크기

페이징 기법을 사용하면 프로그램을 페이지 단위로 나누게 되는데, 페이지의 크기에 따라 시스템에 미치는 영향이 다르다. 페이지 크기에 따른 특징은 다음과 같다.

### 페이지 크기가 작을 경우

- 페이지 단편화가 감소되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 줄어든다.
- 불필요한 내용이 주기억장치에 적재될 확률이 적으므로 효율적인 워킹 셋을 유지할 수 있다.
- Locality에 더 일치할 수 있기 때문에 기억장치 효율이 높아진다.
- 페이지 정보를 갖는 페이지 맵 테이블의 크기가 커지고, 매핑 속도가 늦어진다.
- 디스크 접근 횟수가 많아져서 전체적인 입·출력 시간은 늘어난다.

### 페이지 크기가 클 경우

- 페이지 정보를 갖는 페이지 맵 테이블의 크기가 작아지고, 매핑 속도가 빨라진다.
- 디스크 접근 횟수가 줄어들어 전체적인 입·출력의 효율성이 증가된다
- 페이지 단편화가 증가되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 늘어난다.
- 프로세스(프로그램) 수행에 불필요한 내용까지도 주기억장치에 적재될 수 있다.

## 2 Locality

Locality(국부성, 지역성, 구역성, 국소성)는 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론이다.

- 스래싱을 방지하기 위한 워킹 셋 이론의 기반이 되었다.
- 프로세스가 집중적으로 사용하는 페이지를 알아내는 방법 중 하나로, 가상기억장치 관리의 이론적인 근거가 된다.
- 데닝(Denning) 교수에 의해 구역성의 개념이 증명되었으며 캐시 메모리 시스템의 이론적 근거이다.
- Locality의 종류에는 시간 구역성(Temporal Locality)과 공간 구역성(Spatial Locality)이 있다.

### 시간 구역성(Temporal Locality)

- 시간 구역성은 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상이다.
- 한 번 참조한 페이지는 가까운 시간 내에 계속 참조할 가능성이 높음을 의미한다.
- **시간 구역성이 이루어지는 기억 장소** : Loop(반복, 순환), 스택(Stack), 부 프로그램(Sub Routine), Counting(1씩 증감), 집계(Totaling)에 사용되는 변수(기억장소)

### 공간 구역성(Spatial Locality)

- 공간 구역성은 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상이다.
- 어느 하나의 페이지를 참조하면 그 근처의 페이지를 계속 참조할 가능성이 높음을 의미한다.
- **공간 구역성이 이루어지는 기억장소** : 배열 순회(Array Traversal, 배열 순례), 순차적 코드의 실행, 프로그래머들이 관련된 변수(데이터를 저장할 기억장소)들을 서로 근처에 선언하여 할당되는 기억장소, 같은 영역에 있는 변수를 참조할 때 사용

## 3 워킹 셋(Working Set)

워킹 셋은 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합이다.

- 데닝(Denning)이 제안한 프로그램의 움직임에 대한 모델로, 프로그램의 Locality 특징을 이용한다.
- 자주 참조되는 워킹 셋을 주기억장치에 상주시킴으로써 페이지 부재 및 페이지 교체 현상이 줄어들어 프로세스의 기억장치 사용이 안정된다.
- 시간이 지남에 따라 자주 참조하는 페이지들의 집합이 변화하기 때문에 워킹 셋은 시간에 따라 변경된다.

## 4 페이지 부재 빈도 방식

페이지 부재(Page Fault)는 프로세스 실행 시 참조할 페이지가 주기억장치에 없는 현상이며, 페이지 부재 빈도(PFF; Page Fault Frequency)는 페이지 부재가 일어나는 횟수를 의미한다.

- 페이지 부재 빈도 방식은 페이지 부재율(Page Fault Rate)에 따라 주기억장치에 있는 페이지 프레임의 수를 늘리거나 줄여 페이지 부재율을 적정 수준으로 유지하는 방식이다.
- 운영체제는 프로세스 실행 초기에 임의의 페이지 프레임을 할당하고, 페이지 부재율을 지속적으로 감시하고 있다가 부재율이 상한선을 넘어가면 좀더 많은 페이지 프레임을 할당하고, 부재율이 하한선을 넘어가면 페이지 프레임을 회수하는 방식을 사용한다.



#### 전문가의 조언

워킹 셋은 '자주 참조하는 페이지의 집합'이라는 것을 중심으로 개념과 특징에 대해 정리하세요.



#### 전문가의 조언

654쪽의 그래프 그림을 참조하여 페이지 부재 빈도 방식의 개념을 이해하세요.

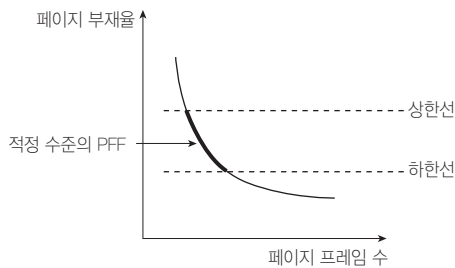


#### 전문가의 조언

스래싱의 개념과 스래싱 현상 방지 방법을 파악하고, 그래프를 통해 다중 프로그래밍의 정도에 따른 CPU의 이용률을 정리하세요.

#### 다중 프로그래밍의 정도

얼마나 많은 프로그램을 동시에 수행하느냐를 나타내는 것으로, 다중 프로그래밍의 정도를 높인다는 것은 동시에 수행하는 프로그램의 수를 늘린다는 것입니다.



## 5 프리페이징(Prepaging)

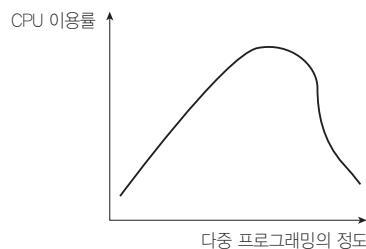
프리페이징은 처음의 과도한 페이지 부재를 방지하기 위해 필요할 것 같은 모든 페이지를 한꺼번에 페이지 프레임에 적재하는 기법이다.

- 기억장치에 들어온 페이지들 중에서 사용되지 않는 페이지가 많을 수도 있다.

## 6 스래싱(Thrashing)

스래싱은 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상이다.

- 다중 프로그래밍 시스템이나 가상기억장치를 사용하는 시스템에서 하나의 프로세스 수행 과정중 자주 페이지 부재가 발생함으로써 나타나는 현상으로, 전체 시스템의 성능이 저하된다.
- 다중 프로그래밍의 정도\*가 높아짐에 따라 CPU의 이용률은 어느 특정 시점까지는 높아지지만, 다중 프로그래밍의 정도가 더욱 커지면 스래싱이 나타나고, CPU의 이용률은 급격히 감소하게 된다.



#### 스래싱 현상 방지 방법

- 다중 프로그래밍의 정도를 적정 수준으로 유지한다.
- 페이지 부재 빈도(Page Fault Frequency)를 조절하여 사용한다.
- 워킹 셋을 유지한다.
- 부족한 자원을 증설하고, 일부 프로세스를 중단시킨다.
- CPU 성능에 대한 자료의 지속적 관리 및 분석으로 임계치를 예상하여 운영한다.



## 기출문제 따라잡기

Section 147

이전기술

## 1. 페이지(Page) 크기에 대한 설명으로 옳은 것은?

- ① 페이지 크기가 작을 경우, 동일한 크기의 프로그램에 더 많은 수의 페이지가 필요하게 되어 주소 변환에 필요한 페이지 사상표의 공간은 더 작게 요구된다.
- ② 페이지 크기가 작을 경우, 페이지 단편화를 감소시키고 특정한 참조 지역성만을 포함하기 때문에 기억장치 효율은 좋을 수 있다.
- ③ 페이지 크기가 클 경우 페이지 단편화로 인해 많은 기억 공간을 낭비하고 페이지 사상표의 크기도 늘어난다.
- ④ 페이지 크기가 클 경우, 디스크와 기억장치 간에 대량의 바이트 단위로 페이지가 이동하기 때문에 디스크 접근 시간 부담이 증가되어 페이지 이동 호출이 나빠진다.

페이지 크기가 큰 것이나 작은 것 중 하나의 특징을 정확히 알아두면 쉽게 맞힐 수 있습니다. 무조건 외우지 말고 페이지 크기가 작으면 페이지의 수가 늘어나므로 페이지 정보를 갖는 페이지 사상 테이블 크기가 커진다는 식으로 연관지어 기억하면 쉽습니다.

이전기술

## 2. 구역성(Locality)에 대한 설명으로 옳지 않은 것은?

- ① Denning에 의해 증명된 이론으로 어떤 프로그램의 참조 영역은 지역화 된다는 것이다.
- ② 워킹 셋(Working Set) 이론의 바탕이 되었다.
- ③ 시간 구역성은 어떤 프로세스가 최근에 참조한 기억 장소의 특정 부분은 그 후에도 계속 참조할 가능성이 높음을 의미한다.
- ④ 부 프로그램이나 서브루틴, 순환 구조를 가진 루틴, 스택 등의 프로그램 구조나 자료 구조는 공간 구역성의 특성을 갖는다.

시간 구역성과 공간 구역성에 해당하는 기억장소를 정확하게 구분할 수 있어야 한다고 했죠?

이전기술

## 3. 시간적 구역성(Temporal Locality)의 예가 아닌 것은?

- ① 루프
- ② 서브루틴
- ③ 프로그램의 순차적 수행
- ④ 스택

시간 구역성과 공간 구역성을 정확히 구분할 수 있어야 합니다. 배열 순회, 순차적 코드의 실행, 프로그래머들이 관련된 변수들을 서로 근처에 선언하여 활용되는 기억장소 등은 공간 구역성에 해당됩니다.

이전기술

## 4. 프로세스들이 국부적인 부분만을 집중적으로 참조하는 구역성에는 시간 구역성과 공간 구역성이 있는데, 다음 중 공간 구역성의 경우는?

- ① 순환(Looping)
- ② 배열 순회(Array Traversal)
- ③ 스택(Stack)
- ④ 집계(Totaling)에 사용되는 변수

순환, 스택, 집계에 사용되는 변수는 어느 특정한 시기를 기준으로 하죠? 그럼, 공간 구역성은? 네, 맞습니다. 특정 공간을 기준으로 합니다.

이전기술

## 5. Denning이 제안한 프로그램의 움직임에 관한 모델로 프로세스를 효과적으로 실행하기 위하여 주기억장치에 유지되어야 하는 페이지들의 집합을 의미하는 것은?

- ① Locality
- ② Working set
- ③ Overlay
- ④ Mapping

'페이지 집합'이란 말이 나오면 워킹 셋(Working Set)이라고 했죠? 자주 사용되는 것을 모아 놓은 것입니다.

이전기술

## 6. 페이지 오류율(Page Fault Ratio)과 스래싱(Thrashing)에 대한 설명으로 옳은 것은?

- ① 페이지 오류율이 크면 스래싱이 많이 발생한 것이다.
- ② 페이지 오류율과 스래싱은 전혀 관계가 없다.
- ③ 스래싱이 많이 발생하면 페이지 오류율이 감소한다.
- ④ 다중 프로그래밍의 정도가 높을수록 페이지 오류율과 스래싱이 감소한다.

페이지 오류율은 페이지 부재가 발생하는 비율을 의미하는 것으로, 페이지 부재가 많이 발생되면 페이지 교체가 자주 발생되겠죠? 즉 페이지 오류율이 높을수록 스래싱이 많이 발생합니다.

이전기술

7. Working set  $W(t, w)$ 는  $t-w$  시간부터  $t$ 까지 참조된 page들의 집합을 말한다. 그 시간에 참조된 페이지가 {2, 3, 5, 5, 6, 3, 7}라면 Working set은?

- ① {3, 5}
- ② {2, 6, 7}
- ③ {2, 3, 5, 6, 7}
- ④ {2, 7}

워킹 셋은 프로세스가 일정 시간 동안 참조하는 페이지들의 집합이므로 참조된 페이지에서 중복을 제거하면 됩니다.

▶ 정답 : 1. ② 2. ④ 3. ③ 4. ② 5. ② 6. ① 7. ③



이전기출

8. 하나의 프로세스가 작업을 수행하는 과정에서 기억장치 접근에서 지나치게 페이지 폴트가 발생하여 프로세스 수행에 소요되는 시간보다 페이지 이동에 소요되는 시간이 더 커지는 현상은?

- ① 스래싱(Thrashing)
- ② 워킹셋(Working set)
- ③ 세마포어(Semaphore)
- ④ 교환(Swapping)

스래싱(Thrashing)은 페이지 부재로 인해 페이지 교체가 자주 일어나는 현상입니다.

이전기출

9. 워킹 셋(Working Set)에 대한 설명으로 옳지 않은 것은?

- ① 프로세스가 실행하는 과정에서 시간이 지남에 따라 자주 참조하는 페이지들의 집합이 변화하기 때문에 워킹 셋은 시간에 따라 바뀌게 된다.
- ② 프로그램의 구성성(Locality) 특징을 이용한다.
- ③ 워킹 셋에 속한 페이지를 참조하면 프로세스의 기억장치 사용은 안정상태가 된다.
- ④ 페이지 이동에 소요되는 시간과 프로세스 수행에 소요되는 시간의 차이를 의미한다.

워킹 셋의 개념을 기억하라고 했죠? 워킹 셋은 '자주 참조하는 페이지의 집합'이란 것 기억하세요.

이전기출

10. 페이지징 기법과 관련된 설명으로 옳지 않은 것은?

- ① 어떤 프로세스가 프로그램 실행에 사용하는 시간보다 페이지 적재/대체에 소비하는 시간이 더 큰 경우에 스래싱이 발생한다.
- ② 페이지 크기가 작을 경우 페이지 테이블의 공간이 많이 요구된다.
- ③ 작업 셋(Working Set) 방식은 스래싱을 방지하는 방법 중의 하나이다.
- ④ 다중 프로그래밍의 정도가 높을수록 스래싱의 발생 빈도는 낮아진다.

다중 프로그래밍 정도가 높아지면 어느 시점까지는 CPU의 이용률이 높고 스래싱의 발생 빈도가 낮아지지만 어느 시점을 넘어서면 CPU의 이용률이 낮아지고 스래싱의 발생 빈도가 높아집니다. 다중 프로그래밍 정도가 높다고 무조건 좋은 것은 아닙니다.

이전기출

11. 시스템을 설계할 때 최적의 페이지 크기에 관한 결정이 이루어져야만 한다. 페이지 크기에 관한 설명으로 옳지 않은 것은?

- ① 페이지 크기가 크면 페이지 테이블 공간은 증가한다.
- ② 입·출력 전송 시 큰 페이지가 더 효율적이다.
- ③ 페이지 크기가 클수록 디스크 접근 시간 부담이 감소된다.
- ④ 페이지 크기가 작아도 페이지 테이블의 단편화는 발생한다.

1번 문제에서 페이지 크기가 큰 것이나 작은 것 중 하나의 특징만이라도 제대로 알아두면 쉽다고 했죠? 페이지 크기가 커지면 페이지의 수가 적어지므로 페이지 정보를 갖는 페이지 테이블 공간은 줄어듭니다.

이전기출

12. 실행중인 프로세스는 일정 시간에 메모리의 일정 부분만을 집중적으로 참조한다는 개념을 의미하는 것은?

- ① Locality
- ② Monitor
- ③ Spooling
- ④ Fragmentation

이 섹션에서 나오는 용어는 모두 정확하게 암기해야 합니다. 이미 다 배운 내용이나 모르겠으면 앞 페이지를 다시 펼쳐보세요.

이전기출

13. 페이지 대체 문제에 관련된 사항 중 잘못된 것은?

- ① 스래싱(Thrashing) 현상이 일어나면 시스템의 처리율이 증가한다.
- ② 시간 지역성이란 최근에 참조한 기억장소가 다시 참조될 가능성이 높다는 것이다.
- ③ 공간 지역성이란 참조된 기억장소에 대해 근처의 기억장소가 다시 참조될 가능성이 높다는 것이다.
- ④ 어떤 프로세스가 빈번하게 참조하는 페이지들의 집합을 작업 셋이라 한다.

스래싱은 프로세스 처리 시간보다 페이지 교체 시간이 더 많아지는 현상이죠! 그럼 스래싱이 발생하면 시스템의 처리율이 증가할까요? 감소할까요?

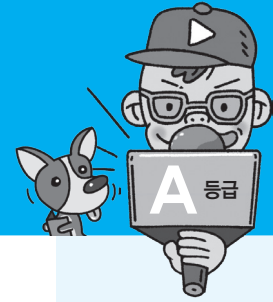
이전기출

14. 스래싱(Thrashing) 현상의 해결 조치로 틀린 것은?

- ① 부족한 자원을 증설한다.
- ② 일부 프로세스를 중단시킨다.
- ③ 성능 자료의 지속적 관리 및 분석으로 임계치를 예상하여 운영한다.
- ④ 다중 프로그래밍의 정도를 높여준다.

이 문제 틀렸으면 반성하고 10번 문제의 해설을 다시 한 번 읽어보세요.

▶ 정답 : 8. ① 9. ④ 10. ④ 11. ① 12. ① 13. ① 14. ④



## 1 프로세스(Process)의 정의

프로세스는 일반적으로 프로세서(처리기, CPU)에 의해 처리되는 사용자 프로그램, 시스템 프로그램, 즉 실행중인 프로그램을 의미하며, 작업(Job), 태스크(Task)라고도 한다.

- 프로세스는 다음과 같이 여러 형태로 정의할 수 있다.
  - PCB를 가진 프로그램
  - 실기억장치에 저장된 프로그램
  - 프로세서가 할당되는 실체로서, 디스패치가 가능한 단위
  - 프로시저\*가 활동중인 것
  - 비동기적 행위\*를 일으키는 주체
  - 지정된 결과를 얻기 위한 일련의 계통적 동작
  - 목적 또는 결과에 따라 발생하는 사건들의 과정
  - 운영체제가 관리하는 실행 단위

## 2 PCB(Process Control Block, 프로세스 제어 블록)

PCB는 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓는 곳으로, Task Control Block 또는 Job Control Block이라고도 한다.

- 각 프로세스가 생성될 때마다 고유의 PCB가 생성되고, 프로세스가 완료되면 PCB는 제거된다.
- PCB에 저장되어 있는 정보는 다음과 같다.

저장 정보	설명
프로세스의 현재 상태	준비, 대기, 실행 등의 프로세스 상태
포인터	<ul style="list-style-type: none"> <li>• 부모 프로세스*에 대한 포인터 : 부모 프로세스의 주소 기억</li> <li>• 자식 프로세스*에 대한 포인터 : 자식 프로세스의 주소 기억</li> <li>• 프로세스가 위치한 메모리에 대한 포인터 : 현재 프로세스가 위치한 주소 기억</li> <li>• 할당된 자원에 대한 포인터 : 프로세스에 할당된 각 자원에 대한 주소 기억</li> </ul>
프로세스 고유 식별자	프로세스를 구분할 수 있는 고유의 번호
스케줄링 및 프로세스의 우선순위	스케줄링 정보 및 프로세스가 실행될 우선순위
CPU 레지스터 정보	Accumulator(누산기), 인덱스 레지스터, 범용 레지스터, 프로그램 카운터(PC) 등에 대한 정보

### 전문가의 조언

다양하게 표현되는 프로세스의 정의를 모두 파악해 두세요.

### 프로시저

한 프로그램은 여러 개의 작은 프로그램으로 분할될 수 있는데, 이때 분할된 작은 프로그램을 의미하며, 부프로그램이라고도 합니다.

### 비동기적 행위

다수의 프로세스가 서로 규칙적이거나 연속적이지 않고 독립적으로 실행되는 것을 말합니다.

### 전문가의 조언

PCB에 저장되는 정보와 저장되지 않는 정보를 명확히 구분할 수 있도록 정리하세요.

### 부모 프로세스 / 자식 프로세스

하나의 프로세스로 다른 프로세스를 생성할 수 있는데, 이때 생성되는 프로세스를 자식 프로세스라 하고, 기존에 있는 프로세스를 부모 프로세스라고 합니다.



#### 기준 레지스터(Base Register)

기준 레지스터는 주기억장치가 분할된 영역으로 나뉘어 관리될 때, 프로그램이 한 영역에서 다른 영역으로 옮겨지더라도 명령의 주소 부분을 바꾸지 않고 정상적으로 수행될 수 있도록 하기 위한 레지스터입니다.

#### 페이지 테이블(Page Table)

페이지 테이블은 페이징 기법에서 주소 변환을 위해 페이지가 존재하는 주기억장치의 위치 정보를 가지고 있는 테이블을 의미합니다. 페이징 기법은 146 Section에서 자세히 다룹니다.

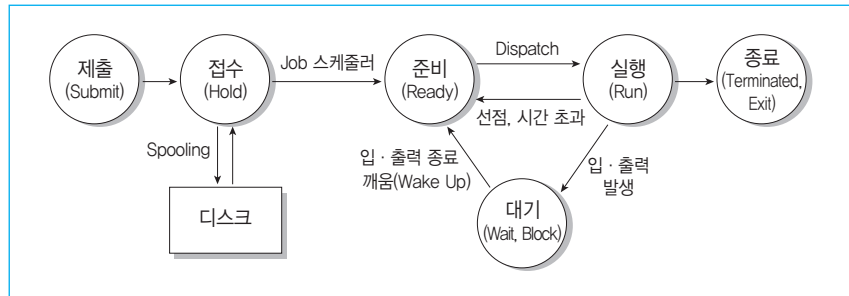
#### 준비상태 큐

준비상태 큐는 여러 프로세스가 프로세서를 할당 받기 위해 기다리는 장소입니다.

주기억장치 관리 정보	기준 레지스터(Base Register)*, 페이지 테이블(Page Table)*에 대한 정보
입 · 출력 상태 정보	입 · 출력장치, 개방된 파일 목록
계정 정보	CPU 사용 시간, 실제 사용 시간, 한정된 시간

### 3 프로세스 상태 전이

프로세스 상태 전이는 프로세스가 시스템 내에 존재하는 동안 프로세스의 상태가 변하는 것을 의미하며, 프로세스의 상태를 다음과 같이 상태 전이도로 표시할 수 있다.



- 프로세스의 상태는 제출, 접수, 준비, 실행, 대기 상태로 나눌 수 있으며, 이 중 주요 세 가지 상태는 준비, 실행, 대기 상태이다.
- **제출(Submit)** : 작업을 처리하기 위해 사용자가 작업을 시스템에 제출한 상태이다.
- **접수(Hold)** : 제출된 작업이 스폴 공간인 디스크의 할당 위치에 저장된 상태이다.
- **준비(Ready)**
  - 프로세스가 프로세서를 할당받기 위해 기다리고 있는 상태이다.
  - 프로세스는 준비상태 큐\*에서 실행을 준비하고 있다.
  - 접수 상태에서 준비 상태로의 전이는 Job 스케줄러에 의해 수행된다.
- **실행(Run)**
  - 준비상태 큐에 있는 프로세스가 프로세서를 할당받아 실행되는 상태이다.
  - 프로세스 수행이 완료되기 전에 프로세스에게 주어진 프로세서 할당 시간이 종료(Timer Run Out)되면 프로세스는 준비 상태로 전이된다.
  - 실행중인 프로세스에 입 · 출력(I/O) 처리가 필요하면 실행중인 프로세스는 대기 상태로 전이된다.
  - 준비 상태에서 실행 상태로의 전이는 CPU(프로세서) 스케줄러에 의해 수행된다.
- **대기(Wait, 보류, 블록(Block))** : 프로세스에 입 · 출력 처리가 필요하면 현재 실행중인 프로세스가 중단되고, 입 · 출력 처리가 완료될 때까지 대기하고 있는 상태이다.
- **종료(Terminated, Exit)** : 프로세스의 실행이 끝나고 프로세스 할당이 해제된 상태이다.



## 4 프로세스 상태 전이 관련 용어

- **Dispatch** : 준비 상태에서 대기하고 있는 프로세스 중 하나가 프로세서를 할당받아 실행 상태로 전이되는 과정이다.
- **Wake Up** : 입·출력 작업이 완료되어 프로세스가 대기 상태에서 준비 상태로 전이 되는 과정이다.
- **Spooling** : 입·출력장치의 공유 및 상대적으로 느린 입·출력장치의 처리 속도를 보완하고 다중 프로그래밍 시스템의 성능을 향상시키기 위해 입·출력할 데이터를 직접 입·출력장치에 보내지 않고 나중에 한꺼번에 입·출력하기 위해 디스크에 저장하는 과정이다.
- **교통량 제어기(Traffic Controller)** : 프로세스의 상태에 대한 조사와 통보를 담당한다.

## 5 스레드(Thread)

스레드는 프로세스 내에서의 작업 단위로서 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위이다.

- 하나의 프로세스에 하나의 스레드가 존재하는 경우에는 단일 스레드, 하나 이상의 스레드가 존재하는 경우에는 다중 스레드라고 한다.
- 프로세스의 일부 특성을 갖고 있기 때문에 경량(Light Weight) 프로세스라고도 한다.
- 스레드 기반 시스템에서 스레드는 독립적인 스케줄링의 최소 단위로서 프로세스의 역할을 담당한다.
- 동일 프로세스 환경에서 서로 독립적인 다중 수행이 가능하다.
- **스레드의 분류**

사용자 수준의 스레드	<ul style="list-style-type: none"> <li>• 사용자가 만든 라이브러리를 사용하여 스레드를 운용한다.</li> <li>• 속도는 빠르지만 구현이 어렵다.</li> </ul>
커널 수준의 스레드	<ul style="list-style-type: none"> <li>• 운영체제의 커널에 의해 스레드를 운용한다.</li> <li>• 구현이 쉽지만 속도가 느리다</li> </ul>

### • 스레드 사용의 장점

- 하나의 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진시킬 수 있다.
- 하드웨어, 운영체제의 성능과 응용 프로그램의 처리율을 향상시킬 수 있다.
- 응용 프로그램의 응답 시간(Response Time)을 단축시킬 수 있다.
- 실행 환경을 공유시켜 기억장소의 낭비가 줄어든다.
- 프로세스들 간의 통신이 향상된다.
- 스레드는 공통적으로 접근 가능한 기억장치를 통해 효율적으로 통신한다.



### 전문가의 조언

스레드의 개념을 기억하고, 특징은 구분할 수 있을 정도로만 정리해 두세요.



## 기출문제 따라잡기

Section 148

이전기출

1. 프로세스(Process)의 정의에 대한 설명 중 옳지 않은 것은?

- ① 동기적 행위를 일으키는 주체
- ② 실행중인 프로그램
- ③ 프로시저의 활동
- ④ 운영체제가 관리하는 실행 단위

프로세스의 정의를 여러 가지 형태로 암기해야 한다고 했죠? 프로세스는 비동기적 행위를 일으키는 주체입니다. 정확히 기억하세요.

이전기출

2. 다음은 무엇에 관한 정의인가?

- 실행중인 프로그램
- 프로시저가 활동 중인 것
- 비동기적 행위를 일으키는 주체
- PCB의 존재로서 명시되는 것

- ① 페이지
- ② 프로세스
- ③ 모니터
- ④ 세그멘테이션

1번과 문제의 형태가 약간 다르지만 프로세스 정의를 기억한다면 쉽게 풀 수 있겠네요. 1번과 2번 문제를 통해 여러 형태의 프로세스 정의를 정확히 파악하세요.

이전기출

3. 프로세스 제어 블록(Process Control Block)에 대한 설명으로 옳지 않은 것은?

- ① 프로세스에 할당된 자원에 대한 정보를 갖고 있다.
- ② 프로세스의 우선순위에 대한 정보를 갖고 있다.
- ③ 부모 프로세스와 자식 프로세스는 PCB를 공유한다.
- ④ 프로세스의 현 상태를 알 수 있다.

PCB가 무엇이지만 알면 풀 수 있는 문제죠? PCB는 프로세스에 대한 고유 정보입니다. 따져봅시다. 우리 엄마에 대한 정보와 나에 대한 정보가 있는데 그걸 공동으로 사용하는 정보라고 같이 사용할 수 있을까요? 프로세스마다 고유의 PCB를 가지고 있다는 것 잊지마세요.

이전기출

4. PCB(Process Control Block)가 갖고 있는 정보가 아닌 것은?

- ① 프로세스 상태
- ② 프로그램 카운터
- ③ 처리기 레지스터
- ④ 할당되지 않은 주변장치의 상태 정보

PCB가 가지고 있는 정보는 모두 다 알아두세요.

이전기출

5. 실행중인 프로세스가 CPU 할당 시간을 다 사용한 후, 어떤 상태로 전이하는가?

- ① Ready 상태
- ② Running 상태
- ③ Block 상태
- ④ Suspended 상태

프로세스 완료 전 할당된 시간 종료로 인해 전이되는 상태는 '준비', I/O 요구에 의해 전이되는 상태는 대기! 정확하게 구분하세요.

이전기출

6. 준비 상태에서 대기하고 있는 프로세스 중 하나가 스케줄링되어 중앙처리장치를 할당받아 실행 상태로 전이되는 과정을 무엇이라 하는가?

- ① 실행(Run)
- ② 준비(Ready)
- ③ 대기(Wait)
- ④ 디스패치(Dispatch)

문제를 잘 읽어보고 답을 찾으세요. '준비 상태에서 실행 상태로 전이되는 과정'을 묻는 문제입니다. 그렇다면 실행 상태도 아니고, 준비 상태 혹은 대기 상태도 아니겠네요! 디스패치라는 용어, 정확히 알아두세요.

이전기출

7. 스레드(Thread)에 대한 설명으로 옳지 않은 것은?

- ① 한 개의 프로세스는 여러 개의 스레드를 가질 수 없다.
- ② 커널 스레드의 경우 운영체제에 의해 스레드를 운용한다.
- ③ 사용자 스레드의 경우 사용자가 만든 라이브러리를 사용하여 스레드를 운용한다.
- ④ 스레드를 사용함으로써 하드웨어, 운영체제의 성능과 응용 프로그램의 처리율을 향상시킬 수 있다.

하나의 프로세스에 하나 이상의 스레드가 존재하는 경우에는 다중 스레드라고 했죠? 그럼 답이 보이네요.

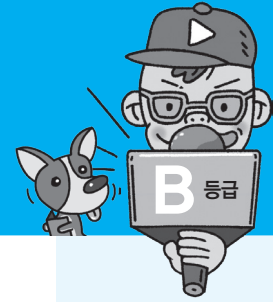
이전기출

8. 스레드(Thread)에 관한 설명으로 옳지 않은 것은?

- ① 스레드는 하나의 프로세스 내에서 병행성을 증대시키기 위한 메커니즘이다.
- ② 스레드는 프로세스의 일부 특성을 갖고 있기 때문에 경량(Light Weight) 프로세스라고도 한다.
- ③ 스레드는 동일 프로세스 환경에서 서로 독립적인 다중 수행이 불가능하다.
- ④ 스레드 기반 시스템에서 스레드는 독립적인 스케줄링의 최소 단위로서 프로세스의 역할을 담당한다.

①번의 '병행성'이라는 단어에 대해 생각해 보세요. 그렇다면 답이 보이는 문제입니다.

▶ 정답 : 1. ① 2. ② 3. ③ 4. ④ 5. ① 6. ④ 7. ① 8. ③



## 1 스케줄링(Scheduling)의 개요

스케줄링은 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업을 의미한다.

- 프로세스가 생성되어 완료될 때까지 프로세스는 여러 종류의 스케줄링 과정을 거치게 된다.
- 스케줄링의 종류에는 장기 스케줄링, 중기 스케줄링, 단기 스케줄링이 있다.

장기 스케줄링	<ul style="list-style-type: none"> <li>• 어떤 프로세스가 시스템의 자원을 차지할 수 있도록 할 것인가를 결정하여 준비상태 큐로 보내는 작업을 의미한다.</li> <li>• 작업 스케줄링(Job Scheduling), 상위 스케줄링이라고도 하며, 작업 스케줄러에 의해 수행된다.</li> </ul>
중기 스케줄링	<ul style="list-style-type: none"> <li>• 어떤 프로세스들이 CPU를 할당받을 것인지 결정하는 작업을 의미한다.</li> <li>• CPU를 할당받으려는 프로세스가 많을 경우 프로세스를 일시 보류시킨 후 활성화해서 일시적으로 부하를 조절한다.</li> </ul>
단기 스케줄링	<ul style="list-style-type: none"> <li>• 프로세스가 실행되기 위해 CPU를 할당받는 시기와 특정 프로세스를 지정하는 작업을 의미한다.</li> <li>• 프로세서 스케줄링(Processor Scheduling), 하위 스케줄링이라고도 한다.</li> <li>• 프로세서 스케줄링 및 문맥 교환은 프로세서 스케줄러에 의해 수행된다</li> </ul>

잠깐만요



### 문맥 교환(Context Switching)

문맥 교환은 하나의 프로세스에서 다른 프로세스로 CPU가 할당되는 과정에서 발생하는 것으로 새로운 프로세스에 CPU를 할당하기 위해 현재 CPU가 할당된 프로세스의 상태 정보를 저장하고, 새로운 프로세스의 상태 정보를 설정한 후 CPU를 할당하여 실행되도록 하는 작업을 의미합니다.

## 2 스케줄링의 목적

스케줄링은 CPU나 자원을 효율적으로 사용하기 위한 정책으로, 다음과 같은 목적을 가지고 있다.

- **공정성** : 모든 프로세스에 공정하게 할당한다.
- **처리율(량) 증가** : 단위 시간당 프로세스를 처리하는 비율(양)을 증가시킨다.
- **CPU 이용률 증가** : 프로세스 실행 과정에서 주기억장치를 액세스한다든지, 입·출력 명령 실행 등의 원인에 의해 발생할 수 있는 CPU의 낭비 시간을 줄이고, CPU가 순수하게 프로세스를 실행하는 데 사용되는 시간 비율을 증가시킨다.



### 전문가의 조언

스케줄링과 문맥 교환이 무엇인지 개념을 파악하고, 스케줄링의 종류를 알아두세요.



### 전문가의 조언

스케줄링은 CPU나 자원을 효율적으로 사용하기 위한 정책이란 것을 중심으로 스케줄링의 목적을 정리하세요.

#### 우선순위

우선순위는 시스템에 의해 자동으로 또는 외부 상황에 의해 결정되기도 합니다. 시간 제한, 기억장치 요구, 개방된 파일 수, 평균 입·출력 실행 시간 등을 이용한 내부적 우선순위와 작업을 지원하는 정책, 부서 등의 외부적 우선순위가 있습니다.



#### 전문가의 조언

선점 스케줄링과 비선점 스케줄링의 의미 및 특징을 파악하고, 각각의 스케줄링에는 어떤 알고리즘들이 있는지 알아두세요.

#### 비선점의 영어 표현

비선점은 Non-Preemption이나 Non-Preemptive로 표현할 수 있습니다.

#### 인터럽트용 타이머 클럭

인터럽트용 타이머 클럭은 하나의 시스템 내에서 동작하는 장치들을 감시하기 위해 주기적인 신호를 발생하는 것으로, 하나의 프로세스가 자원을 독점하지 못하도록 방지하기 위해 사용됩니다.

- **우선순위\* 제도** : 우선순위가 높은 프로세스를 먼저 실행한다.
- **오버헤드 최소화** : 오버헤드를 최소화한다.
- **응답 시간(Response Time, 반응 시간) 최소화** : 작업을 지시하고, 반응하기 시작하는 시간을 최소화한다.
- **반환 시간(Turn Around Time) 최소화** : 프로세스를 제출한 시간부터 실행이 완료될 때까지 걸리는 시간을 최소화한다.
- **대기 시간 최소화** : 프로세스가 준비상태 큐에서 대기하는 시간을 최소화한다.
- **균형 있는 자원의 사용** : 메모리, 입·출력장치 등의 자원을 균형 있게 사용한다.
- **무한 연기 회피** : 자원을 사용하기 위해 무한정 연기되는 상태를 회피한다.

#### 잠깐만요



#### 스케줄링의 성능 평가 기준

스케줄링의 목적 중 CPU 이용률, 처리율, 반환 시간, 대기 시간, 응답 시간은 여러 종류의 스케줄링 성능을 비교하는 기준이 됩니다.

### 3 프로세스 스케줄링의 기법

#### 비선점(Non-Preemptive)\* 스케줄링

- 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법이다.
- 프로세스가 CPU를 할당받으면 해당 프로세스가 완료될 때까지 CPU를 사용한다.
- 모든 프로세스에 대한 요구를 공정하게 처리할 수 있다.
- 프로세스 응답 시간의 예측이 용이하며, 일괄 처리 방식에 적합하다.
- 중요한 작업(짧은 작업)이 중요하지 않은 작업(긴 작업)을 기다리는 경우가 발생할 수 있다.
- 비선점 스케줄링의 종류에는 FCFS, SJF, 우선순위, HRN, 기한부 등의 알고리즘이 있다.

#### 선점(Preemptive) 스케줄링

- 하나의 프로세스가 CPU를 할당받아 실행하고 있을 때 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법이다.
- 우선순위가 높은 프로세스를 빠르게 처리할 수 있다.
- 주로 빠른 응답 시간을 요구하는 대화식 시분할 시스템에 사용된다.
- 많은 오버헤드(Overhead)를 초래한다.
- 선점이 가능하도록 일정 시간 배당에 대한 인터럽트용 타이머 클럭(Clock)\*이 필요하다.
- 선점 스케줄링의 종류에는 Round Robin, SRT, 선점 우선순위, 다단계 큐, 다단계 피드백 큐 등의 알고리즘이 있다.



## Section 149

이전기출

### 1. 선점(Preemptive) 기법의 스케줄링에 해당하는 것은?

- ① FIFO                      ② SJF  
③ HRN                     ④ RR

선점 스케줄링과 비선점 스케줄링을 구분할 수 있어야 합니다. 확실히 파악해 두  
세요.

이전기출

2. 필요한 하드웨어 레지스터를 설정함으로써 프로세스에게 CPU를 할당하고, 문맥 교환을 하는 프로세스 관리 기능은?

- ① Traffic Controller      ② I/O Scheduling  
③ Processor Scheduler      ④ Dispatcher

프로세서(단기) 스케줄링과 문맥 교환 등은 프로세서 스케줄러에 의해 수행됩니다. Dispatcher는 선택된 프로세스에게 CPU 제어를 넘기는 프로그램입니다.

이전기출

### 3. 가장 바람직한 스케줄링 정책은?

- ① CPU 이용률을 줄이고 반환 시간을 늘린다.
- ② 응답 시간을 줄이고 CPU 이용률을 늘린다.
- ③ 대기 시간을 늘리고 반환 시간을 줄인다.
- ④ 반환 시간과 처리율을 늘린다.

보기만 잘 읽어봐도 답이 보이는 문제입니다. 무슨 일을 하든 효율적이고 이익이 되는 것을 목적으로 하겠죠? 어떤 것이 그런지 찾아 보세요.

이전기출

4. 선점 기법과 대비하여 비선점 스케줄링 기법에 대한 설명으로 옳지 않은 것은?

- ① 모든 프로세스들에 대한 요구를 공정히 처리한다.
- ② 응답 시간의 예측이 용이하다.
- ③ 많은 오버헤드(Overhead)를 초래할 수 있다.
- ④ CPU의 사용 시간이 짧은 프로세스들이 사용 시간이 긴 프로세스들로 인하여 오래 기다리는 경우가 발생할 수 있다.

선점 스케줄링은 우선순위가 높은 프로세스를 찾아서 CPU를 할당하는 작업을 반복하므로 이런 과정이 없는 비선점 스케줄링에 비해 오버헤드가 많습니다.

이전기출

5. 선점(Preemptive) 스케줄링 방식에 대한 설명으로 옳지 않은 것은?

- ① 대화식 시분할 시스템에 적합하다.
- ② 긴급하고 높은 우선순위의 프로세스들이 빠르게 처리될 수 있다.

- ③ 일단 CPU를 할당받으면 다른 프로세스가 CPU를 강제로 빼앗을 수 없는 방식이다.
- ④ 선점을 위한 시간 배당에 대한 인터럽트용 타이머 클럭(Clock)이 필요하다.

CPU를 할당받았을 경우 다른 프로세스가 CPU를 차지할 수 있다는 의미겠조?  
구분할 수 있어야 합니다.

이전기출

6. CPU 스케줄링을 평가하는 기준으로 가장 거리가 먼 것은?

- ① 처리량(Throughput)
- ② 대기 시간(Waiting Time)
- ③ 균형 있는 자원 이용
- ④ 오류 복구 시간

스케줄링의 성능 평가 기준에는 CPU 이용률, 처리량, 반환 시간, 대기 시간, 응답 시간, 균형 있는 자원 이용 등이 있습니다. 기억해 두세요.

이전기출

7. 비선점(Non-preemptive) 스케줄링 방식에 해당하는 것으로만 짝지어진 것은?

- ① FCFS(First Come First Service), SJF(Shortest Job First)
- ② RR(Round-Robin), SRT(Shortest Remaining Time)
- ③ SRT(Shortest Remaining Time), SJF(Shortest Job First)
- ④ MQ(Multi-level Queue), FCFS(First Come First Service)

1번 문제를 풀었기 때문에 어렵지 않게 답을 찾을 수 있겠죠.

이전기출

8. 다중 프로그래밍 시스템에서 운영체제에 의해 중앙처리장치가 할당되는 프로세스를 변경하기 위하여 현재 중앙처리장치를 사용하여 실행되고 있는 프로세스의 상태 정보를 저장하고, 앞으로 실행될 프로세스의 상태 정보를 설정한 다음 중앙처리장치를 할당하여 실행이 되도록 하는 작업을 의미하는 것은?

- ① Context Switching
- ② Interrupt
- ③ Semaphore
- ④ Dispatching

무맥 교화의 의미에 대해 파악해 두라고 했죠? 다시 한 번 확인해 보세요

▶ 정답 : 1. ④ 2. ③ 3. ② 4. ③ 5. ③ 6. ④ 7. ① 8. ①



## 전문의가의 조언

- 환경 변수의 개념 및 특징을 정리하고, Windows와 UNIX/LINUX에서 사용하는 환경 변수를 구분하여 알아두세요.
- 환경 변수를 이용하는 소프트웨어를 만든 경우 소프트웨어가 실행될 때 시스템에 저장된 환경 변수를 불러와 사용하게 됩니다. 예를 들어 'USERNAME'이라는 환경 변수를 이용하여 "[USERNAME]님 안녕하세요"라는 메시지를 출력하는 소프트웨어를 만든 경우, 'USERNAME'에 'Sinagong'이 저장된 PC에서 소프트웨어를 실행하면 "Sinagong님 안녕하세요"가 출력되고, 'Gilbut'이 저장된 PC에서 실행하면 "Gilbut님 안녕하세요"가 출력됩니다.

### 변수(Variable)

변수는 컴퓨터가 명령을 처리하는 도중 발생하는 값을 저장하기 위한 공간으로, 변할 수 있는 값을 의미합니다.

### 시스템 환경 변수

시스템 환경 변수는 적용되는 범위가 모듈이나 실행 프로세스 내인 경우 내부 환경 변수, 모듈 외인 경우 외부 환경 변수로 구분합니다.

## 전문의가의 조언

Windows의 명령 프롬프트에서 set을 입력하면 모든 환경 변수와 값을 표시합니다.

## 1 환경 변수의 개요

환경 변수\*(Environment Variable)란 시스템 소프트웨어의 동작에 영향을 미치는 동적인 값들의 모임을 의미한다.

- 환경 변수는 변수명과 값으로 구성된다.
- 환경 변수는 시스템의 기본 정보를 저장한다.
- 환경 변수는 자식 프로세스에 상속된다.
- 환경 변수는 시스템 전반에 걸쳐 적용되는 시스템 환경 변수\*와 사용자 계정 내에 서만 적용되는 사용자 환경 변수로 구분된다.

## 2 Windows의 주요 환경 변수

Windows에서 환경 변수를 명령어나 스크립트에서 사용하려면 변수명 앞뒤에 '%'를 입력해야 한다.

환경 변수명	내용
%ALLUSERPROFILE%	모든 사용자의 프로필이 저장된 폴더
%APPDATA%	설치된 프로그램의 필요 데이터가 저장된 폴더
%ComSpec%	기본 명령 프롬프트로 사용할 프로그램명
%HOMEDRIVE%	로그인한 계정의 정보가 저장된 드라이브
%HOMEPATH%	로그인한 계정의 기본 폴더
%LOGONSERVER%	로그인한 계정이 접속한 서버명
%PATH%	실행 파일을 찾는 경로
%PATHEXT%	cmd에서 실행할 수 있는 파일의 확장자 목록
%PROGRAMFILES%	기본 프로그램의 설치 폴더
%SYSTEMDRIVE%	Windows가 부팅된 드라이브
%SYSTEMROOT%	부팅된 운영체제가 들어 있는 폴더
%TEMP% 또는 %TMP%	임시 파일이 저장되는 폴더
%USERDOMAIN%	로그인한 시스템의 도메인명
%USERNAME%	로그인한 계정 이름
%USERPROFILE%	로그인한 유저의 프로필이 저장된 폴더명

### 3 UNIX / LINUX의 주요 환경 변수

UNIX나 LINUX에서 환경 변수를 명령어나 스크립트에서 사용하려면 변수명 앞에 '\$'를 입력해야 한다.

환경 변수명	내용
\$DISPLAY	현재 X 윈도우* 디스플레이 위치
\$HOME	사용자의 홈 디렉터리
\$LANG	프로그램 사용 시 기본적으로 지원되는 언어
\$MAIL	메일을 보관하는 경로
\$PATH	실행 파일을 찾는 경로
\$PS1	셸 프롬프트 정보
\$PWD	현재 작업하는 디렉터리
\$TERM	로긴 터미널 타입
\$USER	사용자의 이름



## 전문가의 조언

UNIX와 LINUX에서 set, env, printenv 중 하나를 입력하면 모든 환경 변수와 값을 표시합니다.

X 원도

X 윈도는 UNIX 계열의 운영체제  
에서 사용되는 GUI(Graphical User  
Interface) 기반의 시스템 소프트웨어를  
의미합니다.

※ GUI(Graphic User Interface) :  
키보드로 명령어를 직접 입력  
하지 않고, 마우스로 아이콘이  
나 메뉴를 선택하여 모든 작업  
을 수행하는 방식



## 기출문제 따라잡기

## Section 150

## 출제예상

1. 다음 중 환경 변수에 대한 설명으로 가장 옳지 않은 것은?

- ① 시스템의 기본 정보를 저장한다.
- ② 기본적으로 부모 프로세스에서 상속받아 사용한다.
- ③ 변수명과 값으로 구성된다.
- ④ 종류로는 시스템 환경 변수, 사용자 환경 변수, 인터페이스 환경 변수가 있다.

환경 변수의 종류는 크게 시스템 환경 변수와 사용자 환경 변수로 구분됩니다.

## 출제예상

2. Windows에서 환경 변수를 명령어나 스크립트에서 사용하기 위해 변수명 앞뒤에 입력해야 하는 특수문자는?

- [illegible]

Windows에서는 변수명 앞뒤에 %, UNIX/LINUX에서는 변수명 앞에 \$를 입력한다는 것을 기억해 두세요.

## 출제예상

### 3. UNIX에서 모든 환경 변수와 값을 표시하기 위한 명령어가 아닌 것은?

- ① set                      ② view  
③ env                     ④ printenv

UNIX, LINUX에서 모든 환경 변수와 값을 표시하려면 set, env, printenv을, Windows에서는 set을 입력해야 합니다.

## 출제예상

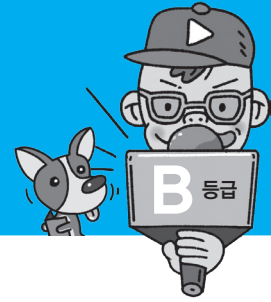
4. 다음 중 Windows에서 사용하는 환경 변수가 아닌 것은?

- ① PATH                      ② SystemDrive  
③ PWD                      ④ TEMP

PWD는 UNIX/LINUX에서 사용하는 환경 변수입니다.

▶ 정답: 1. ④ 2. ① 3. ② 4. ③





## 전문의가의 조언

Windows와 UNIX/LINUX에서 사용하는 명령어를 구분할 수 있어야 하고, 각 명령어들의 개별적인 기능을 알고 있어야 합니다.

### 명령 프롬프트 창 실행 방법

- 실행(윈도우 로고 키+R) 창 또는 프로그램 및 파일 검색란에 'cmd'를 입력한 후 [Enter] 누름
- Windows 7 이하 버전은 보조프로그램에서 '명령 프롬프트' 선택
- Windows 8 이상 버전은 Windows 시스템에서 '명령 프롬프트' 선택

## 1 운영체제 기본 명령어의 개요

- 운영체제를 제어하는 방법은 크게 CLI와 GUI로 구분할 수 있다.
- CLI(Command Line Interface)는 키보드로 명령어를 직접 입력하여 작업을 수행하는 사용자 인터페이스를 의미한다.
- GUI(Graphic User Interface)는 키보드로 명령어를 직접 입력하지 않고, 마우스로 아이콘이나 메뉴를 선택하여 작업을 수행하는 그래픽 사용자 인터페이스를 의미한다.

## 2 Windows 기본 명령어

- CLI 기본 명령어 : 명령 프롬프트(Command) 창\*에 명령어를 입력하여 작업을 수행하는 것으로, 주요 기본 명령어는 다음과 같다.

명령어	기능
DIR	파일 목록을 표시한다.
COPY	파일을 복사한다.
TYPE	파일의 내용을 표시한다.
REN	파일의 이름을 변경한다.
DEL	파일을 삭제한다.
MD	디렉토리를 생성한다.
CD	디렉터리의 위치를 변경한다.
CLS	화면의 내용을 지운다.
ATTRIB	파일의 속성을 변경한다.
FIND	파일을 찾는다.
CHKDSK	디스크 상태를 점검한다.
FORMAT	디스크 표면을 트랙과 섹터로 나누어 초기화한다.
MOVE	파일을 이동한다.

- GUI 기본 명령어 : 바탕 화면이나 Windows 탐색기에서 마우스로 아이콘을 더블 클릭하여 프로그램 실행, 파일의 복사 및 이동, 제어판의 기능 실행 등 모든 작업이 GUI 명령어에 해당한다.



### 3 UNIX / LINUX 기본 명령어

- CLI 기본 명령어 : 셸(Shell)\*에 명령어를 입력하여 작업을 수행하는 것으로, UNIX/LINUX의 주요 기본 명령어는 다음과 같다.

명령어	기능
cat	파일 내용을 화면에 표시한다.
chdir	현재 사용할 디렉터리의 위치를 변경한다.
chmod	파일의 보호 모드를 설정하여 파일의 사용 허가를 지정한다.
chown	소유자를 변경한다.
cp	파일을 복사한다.
exec	새로운 프로세스를 수행한다.
find	파일을 찾는다.
fork	새로운 프로세스를 생성*한다(하위 프로세스 호출, 프로세스 복제 명령).
fsck	파일 시스템을 검사하고 보수한다.
getpid	자신의 프로세스 아이디를 얻는다.
getppid	부모 프로세스 아이디를 얻는다.
ls	현재 디렉터리 내의 파일 목록을 확인한다.
mount/unmount	파일 시스템을 마운팅*한다/마운팅 해제한다.
rm	파일을 삭제한다.
wait	fork 후 exec에 의해 실행되는 프로세스의 상위 프로세스가 하위 프로세스 종료 등의 event를 기다린다.

- GUI 기본 명령어 : UNIX와 LINUX는 기본적으로 CLI를 기반으로 운영되는 시스템이지만 X Window라는 별도의 프로그램을 설치하여 GUI 방식으로 운영할 수 있다.

#### 셸(Shell)

셸은 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기입니다.

#### 프로세스 생성

UNIX와 LINUX에서 새로운 프로세스를 생성한다는 것은 기존 프로세스를 복제한다는 것을 의미합니다. 프로세스가 생성되면 기존 프로세스는 상위(부모) 프로세스가 되고, 생성된 프로세스는 하위(자식) 프로세스가 됩니다.

#### 파일 시스템 마운팅

파일 시스템 마운팅은 새로운 파일 시스템을 기존 파일 시스템의 서버 디렉터리에 연결하는 것을 의미합니다.



#### 기출문제 따라잡기

#### Section 151

##### 이전기출

##### 1. UNIX에서 프로세스를 생성하는 시스템 호출문은?

- ① exec
- ② fork
- ③ ls
- ④ cat

프로세스 생성, 프로세스 복제, 하위 프로세스 호출! 이런 기능을 수행하는 명령은 'fork'입니다.

##### 이전기출

##### 2. Windows와 UNIX에서 사용되는 명령어 중 서로 관련이 없는 것으로 짝지어진 것은?

- ① DIR - ls
- ② COPY - cp
- ③ TYPE - cat
- ④ CD - chmod

Windows에서 CD는 디렉터리의 위치를 변경하는 것으로 UNIX에서는 chdir을 사용합니다. UNIX에서 chmod는 파일의 속성을 변경하는 것으로 Windows에서는 ATTRIB을 사용합니다.

▶ 정답 : 1. ② 2. ④



이전기출

3. 유닉스에서 파일 내용을 화면에 표시하는 명령과 파일의 보호 모드를 설정하여 파일의 사용 허가를 지정하는 명령을 순서적으로 옳게 나열한 것은?

- ① cp, rm                      ② fsck, cat  
③ cat, chmod                ④ find, cp

답을 찾지 못하겠다면 앞쪽의 표를 다시 한 번 공부하고 오세요.

이전기출

4. UNIX에서 기존 파일 시스템에 새로운 파일 시스템을 서브 디렉터리에 연결할 때 사용하는 명령은?

- ① mount                      ② find  
③ fsck                        ④ wait

'find'는 '찾다', 'is'는 '파일 시스템', 'ck'는 '체크하다, 검사하다', 'wait'는 '기다리다'라는 의미입니다. 무조건 암기하려 하지 말고 명령어의 의미를 파악해 보세요.

이전기출

5. UNIX에서 파일의 조작을 위한 명령어가 아닌 것은?

- ① cp                      ② cat  
③ ls                     ④ rm

'cp'는 파일 복사, 'cat'은 파일 내용 화면 표시, 'rm'은 파일 삭제, 'ls'는 현재 디렉터리의 파일 목록을 확인하는 명령입니다. 이제 답을 찾아보세요.

이전기출

6. 자식 프로세스의 하나가 종료될 때까지 부모 프로세스를 일시 중지시키는 유닉스 명령어는?

- ① find( )                      ② fork( )  
③ exec( )                      ④ wait( )

'find'는 파일 찾기, 'fork'는 프로세스 생성, 'exec'는 새로운 프로세스 수행을 위한 명령어입니다. 각 명령어의 기능을 꼭 기억해 두세요.

출제예상

7. Windows에서 디스크의 상태를 확인하는 명령어는?

- ① MD                      ② CLS  
③ CHKDSK                ④ MOVE

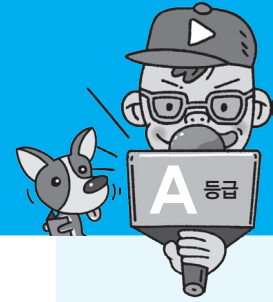
디스크(DISK)의 상태를 확인(CHECK)하는 명령어! 어렵지 않게 답을 찾을 수 있  
겠죠.

출제예상

8. 디스크 표면을 트랙과 섹터로 나누어 초기화하는 Windows 명령은?

- ① FORMAT                      ② COPY  
③ DEL                          ④ CLS

디스크를 초기화하여 사용 가능한 상태로 만들어 주는 작업을 포맷(Format)이라고 합니다.



## 1 인터넷(Internet)의 개요

인터넷이란 TCP/IP\* 프로토콜을 기반으로 하여 전 세계 수많은 컴퓨터와 네트워크들이 연결된 광범위한 컴퓨터 통신망이다.

- 인터넷은 미 국방성의 ARPANET에서 시작되었다.
- 인터넷은 유닉스 운영체제를 기반으로 한다.
- 통신망과 컴퓨터가 있는 곳이라면 시간과 장소에 구애받지 않고 정보를 교환할 수 있다.
- 인터넷에 연결된 모든 컴퓨터는 고유한 IP 주소를 갖는다.
- 컴퓨터 또는 네트워크를 서로 연결하기 위해서는 브리지, 라우터, 게이트웨이\*가 사용된다.
- 다른 네트워크 또는 같은 네트워크를 연결하여 중추적 역할을 하는 네트워크로, 보통 인터넷의 주가 되는 기간망을 일컫는 용어를 백본(Backbone)이라고 한다.

## 2 IP 주소(Internet Protocol Address)

IP 주소는 인터넷에 연결된 모든 컴퓨터 자원을 구분하기 위한 고유한 주소이다.

- 숫자로 8비트씩 4부분, 총 32비트로 구성되어 있다.
- IP 주소는 네트워크 부분의 길이에 따라 다음과 같이 A 클래스에서 E 클래스까지 총 5단계로 구성되어 있다.

A Class	<ul style="list-style-type: none"> <li>• 국가나 대형 통신망에 사용(0~127로 시작*)</li> <li>• <math>2^{24} = 16,777,216</math>개의 호스트 사용 가능</li> </ul>	
B Class	<ul style="list-style-type: none"> <li>• 중대형 통신망에 사용(128~191로 시작)</li> <li>• <math>2^{16} = 65,536</math>개의 호스트 사용 가능</li> </ul>	
C Class	<ul style="list-style-type: none"> <li>• 소규모 통신망에 사용(192~223으로 시작)</li> <li>• <math>2^8 = 256</math>개의 호스트 사용 가능</li> </ul>	
D Class	멀티캐스트*용으로 사용(224~239로 시작)	
E Class	실험적 주소이며 공용되지 않음	



### 전문가의 조언

이번 섹션에서 학습하는 내용은 모두 중요합니다. 특히 IPv6 주소의 개념과 구성을 확실히 숙지하세요.

### TCP/IP

TCP/IP는 인터넷의 표준 프로토콜입니다. TCP/IP에 대한 자세한 내용은 Section 156을 참조하세요.

### 브리지, 라우터, 게이트웨이

브리지, 라우터, 게이트웨이는 네트워크를 연결하기 위한 장비로, 자세한 내용은 Section 154를 참조하세요.

### A Class의 실질적인 네트워크 주소

A Class의 네트워크 주소는 0~127로 시작하지만, 0번과 127번은 예약된 주소이므로 실질적으로는 1~126으로 시작합니다.

### 멀티캐스트(Multicast)

멀티캐스트란 한 명 이상의 송신자들이 특정한 한 명 이상의 수신자들에게 데이터를 전송하는 방식으로, 인터넷 화상 회의 등에서 사용됩니다.

- 인증성 : 사용자의 식별과 접근 권한 검증
- 기밀성 : 시스템 내의 정보와 자원은 인가된 사용자에게만 접근 허용
- 무결성 : 시스템 내의 정보는 인가된 사용자만 수정 가능
- Traffic Class(트래픽 클래스) : IPv6 패킷의 클래스나 우선순위를 나타내는 필드
- Flow Label(플로우 레이블) : 네트워크 상에서 패킷들의 흐름에 대한 특성을 나타내는 필드

### 3 서브네팅(Subnetting)

서브네팅은 할당된 네트워크 주소를 다시 여러 개의 작은 네트워크로 나누어 사용하는 것을 말한다.

- 4바이트의 IP 주소 중 네트워크 주소와 호스트 주소를 구분하기 위한 비트를 서브넷 마스크(Subnet Mask)라고 하며, 이를 변경하여 네트워크 주소를 여러 개로 분할하여 사용한다.
- 서브넷 마스크는 각 클래스마다 다르게 사용된다.

### 4 IPv6(Internet Protocol version 6)의 개요

IPv6은 현재 사용하고 있는 IP 주소 체계인 IPv4의 주소 부족 문제를 해결하기 위해 개발되었다.

- 128비트의 긴 주소를 사용하여 주소 부족 문제를 해결할 수 있으며, IPv4에 비해 자료 전송 속도가 빠르다.
- 인증성\*, 기밀성\*, 데이터 무결성\*의 지원으로 보안 문제를 해결할 수 있다.
- IPv4와 호환성이 뛰어나다.
- 주소의 확장성, 융통성, 연동성이 뛰어나며, 실시간 흐름 제어로 향상된 멀티미디어 기능을 지원한다.
- Traffic Class\*, Flow Label\*을 이용하여 등급별, 서비스별로 패킷을 구분할 수 있어 품질 보장이 용이하다.

### 5 IPv6의 구성

- 16비트씩 8부분, 총 128비트로 구성되어 있다.
- 각 부분을 16진수로 표현하고, 콜론(:)으로 구분한다.
- IPv6은 다음과 같이 세 가지 주소 체계로 나누어진다.

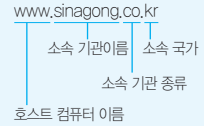
유니캐스트(Unicast)	단일 송신자와 단일 수신자 간의 통신(1 대 1 통신에 사용)
멀티캐스트(Multicast)	단일 송신자와 다중 수신자 간의 통신(1 대 다 통신에 사용)
애니캐스트(Anycast)	단일 송신자와 가장 가까이 있는 단일 수신자 간의 통신(1 대 1 통신에 사용)

## 6 도메인 네임(Domain Name)

도메인 네임은 숫자로 된 IP 주소를 사람이 이해하기 쉬운 문자 형태로 표현한 것이다.

- 호스트 컴퓨터 이름, 소속 기관 이름, 소속 기관의 종류, 소속 국가명 순으로 구성\*되며, 왼쪽에서 오른쪽으로 갈수록 상위 도메인을 의미한다.
- 문자로 된 도메인 네임을 컴퓨터가 이해할 수 있는 IP 주소로 변환하는 역할을 하는 시스템을 DNS(Domain Name System)라고 하며 이런 역할을 하는 서버를 DNS 서버라고 한다.

### 도메인 네임의 구성



### 기출문제 따라잡기

Section 152

#### 이전기술

1. IP Address에 관한 설명으로 옳지 않은 것은?

- ① 5개의 클래스(A, B, C, D, E)로 분류되어 있다.
- ② A, B, C 클래스만이 네트워크 주소와 호스트 주소 체계의 구조를 가진다.
- ③ D 클래스 주소는 멀티캐스팅(Multicasting)을 사용하기 위해 예약되어 있다.
- ④ E 클래스는 특수 목적 주소로 공용으로 사용된다.

D 클래스는 멀티캐스팅, E 클래스는 실험용으로 사용된다는 것! 아직 잊지 않았죠?

#### 이전기술

2. IPv6에 대한 설명으로 틀린 것은?

- ① 더 많은 IP 주소를 지원할 수 있도록 주소의 크기는 64 비트이다.
- ② 프로토콜의 확장을 허용하도록 설계되었다.
- ③ 확장 헤더로 이동성을 지원하고, 보안 및 서비스 품질 기능 등이 개선되었다.
- ④ 유니캐스트, 멀티캐스트, 애니캐스트를 지원한다.

IPv6은 16비트씩 8부분으로 구성되어 있습니다.

#### 이전기술

3. IP 주소와 호스트 이름 간의 변환을 제공하는 분산 데이터베이스를 무엇이라고 하는가?

- ① DNS
- ② NFS
- ③ 라우터
- ④ 웹 서버

문자로 된 도메인 네임을 컴퓨터가 이해할 수 있는 IP 주소로 변환하는 역할을 하는 시스템을 DNS(Domain Name System)라고 합니다.

#### 이전기술

4. 다음은 인터넷의 도메인에 대한 설명이다. 옳지 않은 것은?

www.hankook.co.kr

- ① www : 호스트 컴퓨터 이름
- ② hankook : 소속 기관
- ③ co : 소속 기관의 서버 이름
- ④ kr : 소속 국가

도메인 네임은 호스트 컴퓨터 이름, 소속 기관 이름, 소속 기관의 종류, 소속 국가명 순으로 구성됩니다. 'co'는 소속 기관의 종류를 의미하겠죠?

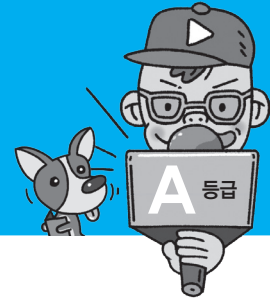
#### 이전기술

5. IP Address에서 네트워크 ID와 호스트 ID를 구별하는 방식은?

- ① 서브넷 마스크
- ② 클래스 E
- ③ 클래스 D
- ④ IPv6

네트워크 ID와 호스트 ID를 구별하는 이유는 네트워크 주소를 여러 개의 서브 네트워크로 나누어 사용하기 위한 것입니다.

▶ 정답 : 1. ④ 2. ① 3. ① 4. ③ 5. ①



## 전문가의 조언

OSI 7계층의 전체적인 순서와 하위 계층 또는 상위 계층을 구분할 수 있어야 합니다. 꼭 외우세요. 하위 계층부터 '물 → 데 → 네 → 전 → 세 → 표 → 응'

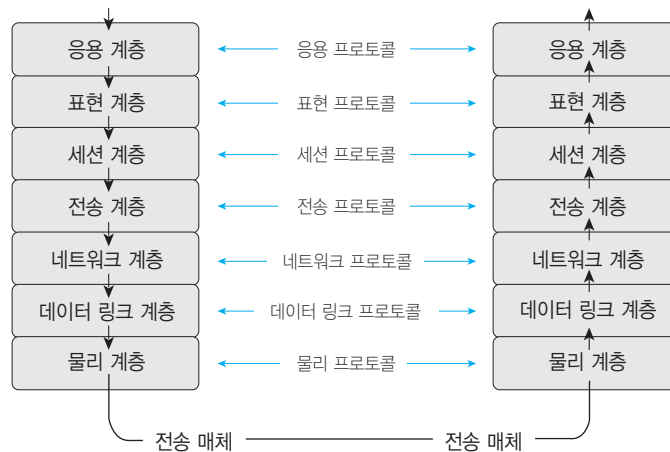
## OSI 참조 모델의 기본 개념

OSI 참조 모델은 특정 시스템에 대한 프로토콜의 의존도를 줄이고, 장래의 기술 진보 등에 따른 프로토콜의 확장성을 고려해 보편적인 개념과 용어를 사용하여 컴퓨터 통신망의 논리 구조를 규정하고 있습니다. 이 개념에 따라 OSI 참조 모델은 통신 회선(물리 매체)에 결합된 하나 이상의 개방형 시스템이, 통신망 상에서 특정한 업무를 분산하여 수행하기 위한 시스템 간의 협동적인 동작에 대하여 규정하고 있습니다. 이 협동적인 동작에는 프로세스 간의 통신, 데이터의 기억, 프로세스 및 자원의 관리, 안전 보호 및 프로그램의 지원 등이 있습니다.

## 1 OSI(Open System Interconnection) 참조 모델의 개요

OSI 참조 모델\*은 다른 시스템 간의 원활한 통신을 위해 ISO(국제표준화기구)에서 제안한 통신 규약(Protocol)이다.

- 개방형 시스템(Open System) 간의 데이터 통신 시 필요한 장비 및 처리 방법 등을 7단계로 표준화하여 규정했다.
- OSI 7계층은 1~3 계층을 하위 계층, 4~7 계층을 상위 계층이라고 한다.
  - 하위 계층 : 물리 계층 → 데이터 링크 계층 → 네트워크 계층
  - 상위 계층 : 전송 계층 → 세션 계층 → 표현 계층 → 응용 계층



## 2 OSI 참조 모델의 목적

- 서로 다른 시스템 간을 상호 접속하기 위한 개념을 규정한다.
- OSI 규격을 개발하기 위한 범위를 정한다.
- 관련 규정의 적합성을 조절하기 위한 공통적 기반을 제공한다.

## 3 OSI 참조 모델에서의 데이터 단위

프로토콜 데이터 단위(PDU; Protocol Data Unit)

프로토콜 데이터 단위는 동일 계층 간에 교환되는 정보의 단위이다.

- 물리 계층 : 비트
- 데이터 링크 계층 : 프레임
- 네트워크 계층 : 패킷
- 전송 계층 : 세그먼트
- 세션, 표현, 응용 계층 : 메시지

#### 서비스 데이터 단위(SDU; Service Data Unit)

서비스 데이터 단위는 서비스 접근점(SAP)\*을 통해 상·하위 계층끼리 주고받는 정보의 단위이다.

### 4 물리 계층(Physical Layer)

물리 계층은 전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의한다.

- 물리적 전송 매체와 전송 신호 방식을 정의하며, RS-232C, X.21 등의 표준이 있다.
- 관련 장비 : 리피터, 허브

### 5 데이터 링크 계층(Data Link Layer)

데이터 링크 계층\*은 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 한다.

- 송신 측과 수신 측의 속도 차이를 해결하기 위한 흐름 제어 기능을 한다.
- 프레임의 시작과 끝을 구분하기 위한 프레임의 동기화 기능을 한다.
- 오류의 검출과 회복을 위한 오류 제어 기능을 한다.
- 프레임의 순서적 전송을 위한 순서 제어 기능을 한다.
- HDLC, LAPB, LLC, MAC, LAPD, PPP 등의 표준이 있다.
- 관련 장비 : 랜카드, 브리지, 스위치

### 6 네트워크 계층(Network Layer, 망 계층)

네트워크 계층은 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 기능을 한다.

- 네트워크 연결을 설정, 유지, 해제하는 기능을 한다.
- 경로 설정(Routing), 데이터 교환 및 중계, 트래픽 제어, 패킷 정보 전송을 수행한다.
- X.25, IP 등의 표준이 있다.
- 관련 장비 : 라우터

#### 서비스 접근점(SAP)

- 상위 계층이 자신의 하위 계층으로부터 서비스를 제공받는 점(Point)을 말합니다.
- OSI 7계층의 각 계층들은 자신의 하위 계층으로부터 서비스를 제공합니다. 이때 하위 계층과 상위 계층의 통신 경계점을 서비스 접근점(SAP)이라고 합니다.



#### 전문가의 조언

OSI 7계층 가운데 어떤 계층을 설명한 것인지를 구분할 수 있어야 합니다. 각 계층의 정의를 바탕으로 주요 기능을 암기하세요.

#### 데이터 링크 계층의 주요 기능

- 흐름 제어
- 프레임 동기화
- 오류 제어
- 순서 제어

#### 전송 계층의 서비스 등급

전송 계층은 네트워크의 형(Type)을 A형, B형, C형의 3개로 나누고, 서비스의 등급(Class)을 0~4까지 5등급으로 나누어, 네트워크형에 따라 다양한 서비스의 품질(QoS)을 제공합니다.

#### 세션(Session)

세션이란 두 이용자 사이의 연결을 의미합니다. 세션 계층은 연결을 원하는 두 이용자 사이의 세션 설정 및 유지를 가능하게 해 줌으로써 두 이용자 간의 대화를 관리하고, 파일 복구 등의 기능을 지원합니다.

#### 소동기점(Minor Synchronization Point)

소동기점은 하나의 대화 단위 내에서 데이터의 전달을 제어하는 역할을 하며, 수신 측으로부터 확인 신호(ACK)를 받지 않습니다.

#### 대동기점(Major Synchronization Point)

대동기점은 전송하는 각 데이터의 처음과 끝에 사용하여 전송하는 데이터 단위를 대화 단위로 구성하는 역할을 하며, 수신 측으로부터 반드시 전송한 데이터에 대한 확인 신호(ACK)를 받습니다.

## 7 전송 계층(Transport Layer)

전송 계층\*은 논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단 시스템(End-to-End) 간에 투명한 데이터 전송을 가능하게 한다.

- OSI 7계층 중 하위 3계층과 상위 3계층의 인터페이스(Interface)를 담당한다.
- 종단 시스템(End-to-End) 간의 전송 연결 설정, 데이터 전송, 연결 해제 기능을 한다.
- 주소 설정, 다중화(분할 및 재조립), 오류 제어, 흐름 제어를 수행한다.
- TCP, UDP 등의 표준이 있다.
- 관련 장비 : 게이트웨이

## 8 세션 계층(Session Layer)

세션\* 계층은 송·수신 측 간의 관련성을 유지하고 대화 제어를 담당한다.

- 대화(회화) 구성 및 동기 제어, 데이터 교환 관리 기능을 한다.
- 송·수신 측 간의 대화 동기를 위해 전송하는 정보의 일정한 부분에 체크점을 두어 정보의 수신 상태를 체크하며, 이때의 체크점을 동기점(Synchronization Point)이라고 한다.
- 동기점은 오류가 있는 데이터의 회복을 위해 사용하는 것으로, 종류에는 소동기점\*과 대동기점\*이 있다.

## 9 표현 계층(Presentation Layer)

표현 계층은 응용 계층으로부터 받은 데이터를 세션 계층에 보내기 전에 통신에 적합한 형태로 변환하고, 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능을 한다.

- 서로 다른 데이터 표현 형태를 갖는 시스템 간의 상호 접속을 위해 필요한 계층이다.
- 코드 변환, 데이터 암호화, 데이터 압축, 구문 검색, 정보 형식(포맷) 변환, 문맥 관리 기능을 한다.

## 10 응용 계층(Application Layer)

응용 계층은 사용자(응용 프로그램)가 OSI 환경에 접근할 수 있도록 서비스를 제공한다.

- 응용 프로세스 간의 정보 교환, 전자 사서함, 파일 전송, 가상 터미널 등의 서비스를 제공한다.





## 기출문제 따라잡기

Section 153

이전기술

1. OSI 참조 모델 계층 구조로 최하위 계층부터 최상위 계층의 순서가 옳은 것은?

- ① 물리 → 네트워크 → 트랜스포트 → 데이터 링크 → 세션 → 프레젠테이션 → 응용
- ② 물리 → 네트워크 → 데이터 링크 → 트랜스포트 → 세션 → 프레젠테이션 → 응용
- ③ 물리 → 데이터 링크 → 네트워크 → 트랜스포트 → 세션 → 프레젠테이션 → 응용
- ④ 물리 → 데이터 링크 → 네트워크 → 트랜스포트 → 프레젠테이션 → 세션 → 응용

OSI 계층은 이렇게 외우면 쉬워요. 물, 데, 네, 전(트랜스포트), 세, 표(프레젠테이션), 응.

이전기술

2. OSI 참조 모델의 데이터 링크 계층에 대한 설명으로 틀린 것은?

- ① 물리적 계층의 신뢰도를 높여 주고 링크의 확립 및 유지할 수 있는 수단을 제공한다.
- ② 에러 제어, 흐름 제어 등의 기능을 수행한다.
- ③ 종단 시스템 간에 End-to-End 통신을 지원한다.
- ④ 대표적 프로토콜로는 HDLC, BSC 등이 있다.

양단 간(End-to-End)의 통신(전송)을 지원하는 기능은 전송(Transport) 계층의 대표적인 기능입니다.

이전기술

3. OSI-7계층 중 프로세스 간의 대화 제어(Dialogue Control) 및 동기점(Synchronization Point)을 이용한 효율적인 데이터 복구를 제공하는 계층은?

- ① Data Link Layer                      ② Network Layer
- ③ Transport Layer                      ④ Session Layer

이 문제에서의 핵심 단어는 "대화 제어와 동기점"입니다.

이전기술

4. OSI 계층 모델 중 각 계층의 기능을 설명한 것으로 옳지 않은 것은?

- ① 물리 계층 - 전기적, 기능적, 절차적 기능 정의
- ② 데이터 링크 계층 - 흐름 제어, 에러 제어
- ③ 네트워크 계층 - 경로 설정 및 네트워크 연결 관리
- ④ 전송 계층 - 코드 변환, 구문 검색

코드 변환, 구문 검색은 표현 계층의 기능입니다.

이전기술

5. OSI 참조 모델의 계층에서 통신 시스템 간의 경로 배정, 패킷 전달, 주소 설정, 통신량의 폭주 제어 등의 기능을 수행하는 것은?

- ① 물리 계층                              ② 데이터링크 계층
- ③ 네트워크 계층                      ④ 전송 계층

이 문제의 핵심은 경로 배정 기능입니다.

이전기술

6. OSI 7계층 중 암호화, 코드 변환, 데이터 압축 등의 역할을 담당하는 계층은?

- ① Data link Layer                      ② Application Layer
- ③ Presentation Layer                      ④ Session Layer

암호화, 코드 변환, 데이터 압축은 표현(Presentation) 계층의 대표적인 기능입니다.

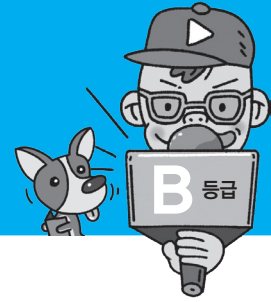
이전기술

7. OSI 7계층에서 단말기 사이에 오류 수정과 흐름 제어를 수행하여 신뢰성 있고 명확한 데이터를 전달하는 계층은?

- ① 전송 계층                              ② 응용 계층
- ③ 세션 계층                              ④ 표현 계층

'오류 수정과 흐름 제어, 신뢰성 있는 데이터 전송? 아~ 데이터링크 계층! 어려? 답이 없네'라고 생각하신 분들 손들어 보세요. 아쉽지만 틀렸습니다. 이 문제에서의 핵심 문구는 '단말기 사이에'입니다. 다른 말로 종단 간(End-to-End)이라고도 하죠. 데이터링크가 답이 되려면 '두 개의 인접한 통신 시스템'이라는 문구가 포함되어야 합니다.

▶ 정답 : 1. ③ 2. ③ 3. ④ 4. ④ 5. ③ 6. ③ 7. ①



## 전문가의 조언

네트워크 구축에 필요한 장비들의 개별적인 기능이 중요합니다. 어떤 네트워크를 연결하는가와, OSI 참조 모델의 어떤 계층에서 동작하는 장비인지를 중심으로 각 장비의 특징을 확실하게 알고 넘어가야 합니다.

### 1 네트워크 인터페이스 카드(NIC; Network Interface Card)

네트워크 인터페이스 카드는 컴퓨터와 컴퓨터 또는 컴퓨터와 네트워크를 연결하는 장치로, 정보 전송 시 정보가 케이블을 통해 전송될 수 있도록 정보 형태를 변경한다.

- 이더넷 카드(LAN 카드) 혹은 네트워크 어댑터라고도 한다.

### 2 허브(Hub)

허브는 한 사무실이나 가까운 거리의 컴퓨터들을 연결하는 장치로, 각 회선을 통합적으로 관리하며, 신호 증폭 기능을 하는 리피터의 역할도 포함한다.

- 허브의 종류에는 더미 허브, 스위칭 허브가 있다.
- 더미 허브(Dummy Hub)
  - 네트워크에 흐르는 모든 데이터를 단순히 연결하는 기능만을 제공한다.
  - LAN이 보유한 대역폭을 컴퓨터 수만큼 나누어 제공한다.
- 예 100MB의 대역폭을 5개의 컴퓨터에 제공한다면 각 컴퓨터는 20MB(100/5)의 대역폭을 사용하게 된다.
- 네트워크에 연결된 각 노드를 물리적인 성형 구조로 연결한다.
- 스위칭 허브(Switching Hub)
  - 네트워크상에 흐르는 데이터의 유무 및 흐름을 제어하여 각각의 노드가 허브의 최대 대역폭을 사용할 수 있는 지능형 허브이다.
  - 최근에 사용되는 허브는 대부분 스위칭 허브이다.

### 3 리피터(Repeater)

리피터는 전송되는 신호가 전송 선로의 특성 및 외부 충격 등의 요인으로 인해 원래의 형태와 다르게 왜곡되거나 약해질 경우 원래의 신호 형태로 재생하여 다시 전송하는 역할을 수행한다.

- OSI 참조 모델의 물리 계층에서 동작하는 장비이다.
- 근접한 네트워크 사이에 신호를 전송하는 역할로, 전송 거리의 연장 또는 배선의 자유도를 높이기 위한 용도로 사용한다.

## 4 브리지(Bridge)

브리지는 LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹(세그먼트)을 연결하는 기능을 수행한다.

- 데이터 링크 계층 중 MAC(Media Access Control) 계층\*에서 사용되므로 MAC 브리지라고도 한다.
- 네트워크 상의 많은 단말기들에 의해 발생하는 트래픽 병목 현상을 줄일 수 있다.
- 네트워크를 분산적으로 구성할 수 있어 보안성을 높일 수 있다.
- 브리지를 이용한 서브넷(Subnet) 구성 시 전송 가능한 회선 수는 브리지가  $n$ 개일 때,  $n(n-1)/2$ 개이다.

## 5 스위치(Switch)

스위치는 브리지와 같이 LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치이다.

- 하드웨어를 기반으로 처리하므로 전송 속도가 빠르다.
- 포트마다 각기 다른 전송 속도를 지원하도록 제어할 수 있고, 수십에서 수백 개의 포트를 제공한다.
- OSI 참조 모델의 데이터 링크 계층에서 사용된다.

## 6 라우터(Router)

라우터는 브리지와 같이 LAN과 LAN의 연결 기능에 데이터 전송의 최적 경로를 선택할 수 있는 기능이 추가된 것으로, 서로 다른 LAN이나 LAN과 WAN의 연결도 수행한다.

- OSI 참조 모델의 네트워크 계층에서 동작하는 장비이다.
- 접속 가능한 경로에 대한 정보를 라우팅 제어표(Routing Table)에 저장하여 보관한다.
- 3계층(네트워크 계층)까지의 프로토콜 구조가 다른 네트워크 간의 연결을 위해 프로토콜 변환 기능을 수행한다.

## 7 게이트웨이(Gateway)

게이트웨이는 전 계층(1~7계층)의 프로토콜 구조가 다른 네트워크의 연결을 수행한다.

- 세션 계층, 표현 계층, 응용 계층 간을 연결하여 데이터 형식 변환, 주소 변환, 프로토콜 변환 등을 수행한다.
- LAN에서 다른 네트워크에 데이터를 보내거나 다른 네트워크로부터 데이터를 받아들이는 출입구 역할을 한다.

### MAC 계층

LAN에서 데이터 링크 계층은 LLC(Logical Link Control) 계층과 MAC(Media Access Control) 계층으로 나누어지는데, 브리지는 이중 MAC(Media Access Control) 계층에서 동작합니다.



### 전문가의 조언

스위치는 Section 168에서 자세히 공부하니 여기에서는 다른 장치와 구분할 정도만 알아두세요.

## 8 네트워크 장비 설치 시 고려 사항

- 네트워크에 설치된 장비를 최대한 활용해야 한다.
- 이후 시스템 확장이나 증설 등을 고려하여 설계한다.
- 하드웨어와 소프트웨어는 최신 버전을 선정한다.
- 트래픽을 분산시킬 수 있도록 설계한다.
- 네트워크 관리나 유지 보수가 용이하게 설계한다.
- 장애 발생 시 즉시 조치할 수 있도록 여유 포트를 고려하여 설계한다.
- 신기술 도입 시 연동할 수 있는 미래 지향적인 네트워크 시스템을 구축한다.



### 기출문제 따라잡기

Section 154

이전기술

1. 두 개의 LAN이 데이터 링크 계층에서 서로 결합되어 있는 경우에 이들을 연결하는 요소를 무엇이라 하는가?

- ① 브리지(Bridge)
- ② 허브(Hub)
- ③ 게이트웨이(Gateway)
- ④ 모뎀(Modem)

각 장비가 동작하는 위치를 OSI 7계층으로 구분하면 브리지는 데이터 링크 계층, 라우터는 네트워크 계층, 게이트웨이는 세션 계층, 표현 계층, 응용 계층입니다.

이전기술

2. 두 개의 서로 다른 형태의 네트워크를 상호 접속하는 3계층 장비를 무엇이라고 하는가?

- ① 허브
- ② 리피터
- ③ 브리지
- ④ 라우터

3계층 장비라면 네트워크 계층에서 동작하는 장비겠죠? 네트워크 계층에서 동작하는 장비는 라우터입니다.

이전기술

3. 인터-네트워킹(Inter-Networking)을 위해 사용되는 네트워크 장비로 가장 거리가 먼 것은?

- ① 리피터(Repeater)
- ② 게이트웨이(Gateway)
- ③ 라우터(Router)
- ④ 증폭기(Amplifier)

증폭기(Amplifier)는 아날로그 신호 증폭을 위한 장비로 인터-네트워킹과는 무관합니다.

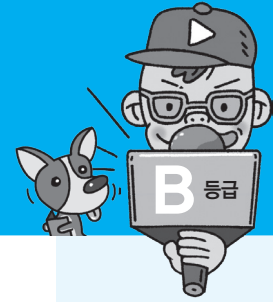
이전기술

4. 적절한 전송 경로를 선택하고 이 경로로 데이터를 전달하는 인터네트워킹(Internetworking) 장비는?

- ① 리피터(Repeater)
- ② 브리지(Bridge)
- ③ 라우터(Router)
- ④ 게이트웨이(Gateway)

각 장비의 기능을 다시 한 번 정확히 기억하세요. 리피터는 신호 재생, 브리지는 동종의 LAN 연결, 라우터는 서로 다른 LAN의 연결과 경로 설정, 게이트웨이는 프로토콜이 다른 네트워크 사이를 연결합니다.

▶ 정답 : 1. ① 2. ④ 3. ④ 4. ③



## 1 프로토콜(Protocol)의 정의

프로토콜은 서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약이다.

- 통신을 제어하기 위한 표준 규칙과 절차의 집합으로 하드웨어와 소프트웨어, 문서를 모두 규정한다.

## 2 프로토콜의 기본 요소

- 구문(Syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
- 의미(Semantics) : 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
- 시간(Timing) : 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정

## 3 프로토콜의 기능

단편화와 재결합	<ul style="list-style-type: none"> <li>• 송신 측에서 전송할 데이터를 전송에 알맞은 일정 크기의 작은 블록으로 자르는 작업을 단편화(Fragmentation)라 하고, 수신 측에서 단편화된 블록을 원래의 데이터로 모으는 것을 재결합(Reassembly)이라 한다.</li> <li>• 단편화를 통해 세분화된 데이터 블록을 프로토콜 데이터 단위*(PDU; Protocol Data Unit)라고 한다.</li> <li>• 데이터를 단편화하여 전송하면 전송 시간이 빠르고, 통신중의 오류를 효과적으로 제어할 수 있다.</li> <li>• 너무 작은 블록으로 단편화할 경우 재결합 시 처리 시간이 길어지고, 실제 데이터 외에 부수적인 데이터가 많아지므로 비효율적이다.</li> </ul>
캡슐화	<ul style="list-style-type: none"> <li>• 캡슐화(Encapsulation)는 단편화된 데이터에 송·수신지 주소, 오류 검출 코드, 프로토콜 기능을 구현하기 위한 프로토콜 제어 정보 등의 정보를 추가하는 것으로, 요약화라고도 한다.</li> <li>• 대표적인 예가 데이터 링크 제어 프로토콜의 HDLC 프레임이다.</li> <li>• 정보 데이터를 오류 없이 정확하게 전송하기 위해 캡슐화를 수행한다.</li> </ul>
흐름 제어	<ul style="list-style-type: none"> <li>• 흐름 제어(Flow Control)는 수신 측의 처리 능력에 따라 송신 측에서 송신하는 데이터의 전송량이나 전송 속도를 조절하는 기능이다.</li> <li>• 정지-대기(Stop-and-Wait)*, 슬라이딩 윈도우(Sliding Window)* 방식을 이용한다.</li> </ul>
오류 제어	오류 제어(Error Control)는 전송중에 발생하는 오류를 검출하고 정정하여 데이터나 제어 정보의 파손에 대비하는 기능이다.
동기화	동기화(Synchronization)는 송·수신 측이 같은 상태를 유지하도록 타이밍(Timing)을 맞추는 기능이다.



### 전문가의 조언

프로토콜이란 두 개체 간에 합의된 약속이란 의미를 가지고 있습니다. 프로토콜의 정의를 확실히 알고, 기본 요소 세 가지를 꼭 외우세요.



### 전문가의 조언

프로토콜의 기능에는 어떤 것들이 있는지 기억해 두세요.

### 프로토콜 데이터 단위(PDU)

- 전송 데이터에 송·수신 측 주소, 오류 검출 코드, 제어 정보가 포함된 것으로, 송·수신 두 기기 사이에 교환되는 데이터의 단위입니다.
- 계층화된 프로토콜에서는 PDU를 각 계층마다 다르게 부르는 데 일반적으로 2계층에서는 프레임(Frame), 3계층에서는 패킷(Packet), 4계층에서는 세그먼트(Segment)라 부릅니다.

### 정지-대기/슬라이딩 윈도우 방식

정지-대기와 슬라이딩 윈도우 방식은 Section 169에 자세히 설명되어 있으니 참조하세요.

### 연결 위주의 데이터 전송 방식

연결 위주의 데이터 전송 방식이란 송·수신 측 상호간에 논리적인 경로가 설정되는 방식으로, 가상 회선 방식이 대표적입니다.

순서 제어	<ul style="list-style-type: none"> <li>• 순서 제어(Sequencing)는 전송되는 데이터 블록(PDU)에 전송 순서를 부여하는 기능으로, 연결 위주의 데이터 전송 방식*에만 사용된다.</li> <li>• 송신 데이터들이 순서적으로 전송되도록 함으로써 흐름 제어 및 오류 제어를 용이하게 하는 기능을 한다.</li> </ul>
주소 지정	<ul style="list-style-type: none"> <li>• 주소 지정(Addressing)은 데이터가 목적지까지 정확하게 전송될 수 있도록 목적지 이름, 주소, 경로를 부여하는 기능이다.</li> <li>• 목적지 이름은 전송할 데이터가 가리키는 곳, 주소는 목적지의 위치, 경로는 목적지에 도착할 수 있는 방법을 의미한다.</li> </ul>
다중화	다중화(Multiplexing)는 한 개의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능이다.
경로 제어	경로 제어(Routing)는 송·수신 측 간의 송신 경로 중에서 최적의 패킷 교환 경로를 설정하는 기능이다.
전송 서비스	<ul style="list-style-type: none"> <li>• 전송하려는 데이터가 사용하도록 하는 별도의 부가 서비스이다.</li> <li>• 우선순위 : 특정 메시지를 최대한 빠른 시간 안에 목적지로 전송하기 위하여 메시지 단위에 우선순위를 부여하여 우선순위가 높은 메시지가 먼저 도착하도록 한다.</li> <li>• 서비스 등급 : 데이터의 요구에 따라 서비스 등급을 부여하여 서비스한다.</li> <li>• 보안성 : 액세스 제한과 같은 보안 체제를 구현한다.</li> </ul>



### 기출문제 따라잡기

Section 155

이전기출

#### 1. 프로토콜이란?

- ① 통신 하드웨어의 표준 규격이다.
- ② 통신 소프트웨어의 개발 환경이다.
- ③ 정보 전송의 통신 규약이다.
- ④ 하드웨어와 사람 사이의 인터페이스이다.

프로토콜은 서로 다른 기기들 간의 데이터 교환을 위해 표준화시켜 놓은 통신 규약입니다.

이전기출

#### 2. 프로토콜의 기본 요소가 아닌 것은?

- ① 구문(Syntax)                      ② 타이밍(Timing)
- ③ 제어(Control)                      ④ 의미(Semantic)

프로토콜의 기본 요소는 구문, 의미, 타이밍(시간)입니다. 꼭 기억하세요.

이전기출

#### 3. 둘 이상의 컴퓨터 사이에 데이터 전송을 할 수 있도록 미리 정의된 송·수신 측에서 정해진 통신 규칙은?

- ① 프로토콜                              ② 링크
- ③ 터미널                                ④ 인터페이스

프로토콜의 정의를 확실히 알아두라고 했죠. 프로토콜(Protocol) = 통신 규약, 통신 규칙입니다.

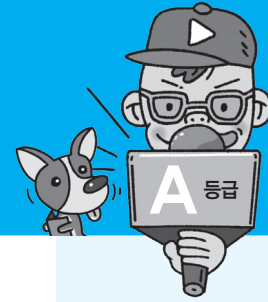
이전기출

#### 4. 프로토콜의 일반적인 기능 중 캡슐화(Encapsulation)할 때 제어 정보에 포함되지 않는 것은?

- ① 연결 제어(Connection Control)
- ② 프로토콜 제어(Protocol Control)
- ③ 에러 검출 코드(Error Detecting Code)
- ④ 주소(Address)

캡슐화에 사용되는 제어 정보 세 가지 기억하나요? 주소, 오류 검출 코드, 프로토콜 제어 정보! 잊지마세요.

▶ 정답 : 1. ③ 2. ③ 3. ① 4. ①



## 1 TCP/IP의 개요(Transmission Control Protocol/Internet Protocol)

TCP/IP는 인터넷에 연결된 서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜이다.

- TCP/IP는 1960년대 말 ARPA에서 개발하여 ARPANET(1972)에서 사용하기 시작했다.
- TCP/IP는 UNIX의 기본 프로토콜로 사용되었고, 현재 인터넷 범용 프로토콜로 사용된다.
- TCP/IP는 다음과 같은 기능을 수행하는 TCP 프로토콜과 IP 프로토콜이 결합된 것을 의미한다.

TCP(Transmission Control Protocol)	<ul style="list-style-type: none"> <li>• OSI 7계층의 전송 계층에 해당</li> <li>• 신뢰성 있는 연결형* 서비스를 제공</li> <li>• 패킷의 다중화, 순서 제어, 오류 제어, 흐름 제어 기능을 제공</li> <li>• 스트림(Stream) 전송 기능 제공</li> <li>• TCP 헤더에는 Source/Destination Port Number, Sequence Number, Acknowledgment Number, Checksum 등이 포함된다.</li> </ul>
IP(Internet Protocol)	<ul style="list-style-type: none"> <li>• OSI 7계층의 네트워크 계층에 해당</li> <li>• 데이터그램을 기반으로 하는 비연결형* 서비스를 제공</li> <li>• 패킷의 분해/조립, 주소 지정, 경로 선택 기능을 제공</li> <li>• 헤더의 길이는 최소 20Byte에서 최대 60Byte이다.</li> <li>• IP 헤더에는 Version, Header Length, Total Packet Length, Header Checksum, Source IP Address, Destination IP Address 등이 포함된다.</li> </ul>

## 2 TCP/IP의 구조

TCP/IP는 응용 계층, 전송 계층, 인터넷 계층, 네트워크 액세스 계층으로 이루어져 있다.\*

OSI	TCP/IP	기능
응용 계층 표현 계층 세션 계층	응용 계층	<ul style="list-style-type: none"> <li>• 응용 프로그램 간의 데이터 송·수신 제공</li> <li>• TELNET, FTP, SMTP, SNMP, DNS, HTTP 등</li> </ul>
전송 계층	전송 계층	<ul style="list-style-type: none"> <li>• 호스트들 간의 신뢰성 있는 통신 제공</li> <li>• TCP, UDP</li> </ul>
네트워크 계층	인터넷 계층	<ul style="list-style-type: none"> <li>• 데이터 전송을 위한 주소 지정, 경로 설정을 제공</li> <li>• IP, ICMP, IGMP, ARP, RARP</li> </ul>
데이터 링크 계층 물리 계층	네트워크 액세스 계층	<ul style="list-style-type: none"> <li>• 실제 데이터(프레임)를 송·수신하는 역할</li> <li>• Ethernet, IEEE 802, HDLC, X.25, RS-232C, ARQ 등</li> </ul>

### 전문가의 조언

TCP와 IP의 기능, TCP/IP를 구성하는 계층을 구분할 수 있어야 합니다. TCP와 IP의 기능과 TCP/IP의 계층이 OSI 7계층의 어느 계층에 대응되는지를 잘 알아 두세요. 또한 TCP/IP의 4계층은 꼭 외우세요.

### 연결형(접속) 통신

연결형 통신은 송·수신 측 간을 논리적으로 연결한 후 데이터를 전송하는 방식으로, 가상 회선 방식이 대표적입니다. 데이터 전송의 안정성과 신뢰성이 보장되지만, 연결 설정 지연이 일어나며, 회선 이용률이 낮아질 수 있습니다.

### 비연결형(비접속) 통신

비연결형 통신은 송·수신 측 간에 논리적 연결 없이 데이터를 독립적으로 전송하는 방식으로, 데이터그램 방식이 대표적입니다.

### TCP/IP 계층 구조

네트워크 액세스 계층을 물리 계층과 데이터 링크 계층으로 세분화하여 물리 계층, 데이터 링크 계층, 인터넷 계층, 전송 계층, 응용 계층 이렇게 5계층으로 구분하기도 합니다.



#### 전문가의 조언

각 계층의 프로토콜 종류와 기능을 정리하세요.

#### TCP를 사용하는 서비스

FTP, SMTP, TELNET, HTTP 등

### 3 응용 계층의 주요 프로토콜

FTP(File Transfer Protocol)	컴퓨터와 컴퓨터 또는 컴퓨터와 인터넷 사이에서 파일을 주고받을 수 있도록 하는 원격 파일 전송 프로토콜
SMTP(Simple Mail Transfer Protocol)	전자 우편을 교환하는 서비스
TELNET	<ul style="list-style-type: none"> <li>멀리 떨어져 있는 컴퓨터에 접속하여 자신의 컴퓨터처럼 사용할 수 있도록 해주는 서비스</li> <li>프로그램을 실행하는 등 시스템 관리 작업을 할 수 있는 가상의 터미널(Virtual Terminal) 기능을 수행</li> </ul>
SNMP(Simple Network Management Protocol)	TCP/IP의 네트워크 관리 프로토콜로, 라우터나 허브 등 네트워크 기기의 네트워크 정보를 네트워크 관리 시스템에 보내는 데 사용되는 표준 통신 규약
DNS(Domain Name System)	도메인 이름을 IP 주소로 매핑(Mapping)하는 시스템
HTTP(HyperText Transfer Protocol)	월드 와이드 웹(WWW)에서 HTML 문서를 송수신 하기 위한 표준 프로토콜

### 4 전송 계층의 주요 프로토콜

TCP(Transmission Control Protocol)*	<ul style="list-style-type: none"> <li>양방향 연결(Full Duplex Connection)형 서비스를 제공한다.</li> <li>가상 회선(Virtual Circuit Connection) 형태의 서비스를 제공한다.</li> <li>스트림 위주의 전달(패킷 단위)을 한다.</li> <li>신뢰성 있는 경로를 확립하고 메시지 전송을 감독한다.</li> <li>순서 제어, 오류 제어, 흐름 제어 기능을 한다.</li> <li>패킷의 분실, 손상, 지연이나 순서가 틀린 것 등이 발생할 때 투명성이 보장되는 통신을 제공한다.</li> </ul>
UDP(User Datagram Protocol)	<ul style="list-style-type: none"> <li>데이터 전송 전에 연결을 설정하지 않는 비연결형 서비스를 제공한다.</li> <li>TCP에 비해 상대적으로 단순한 헤더 구조를 가지므로, 오버헤드가 적다.</li> <li>고속의 안정성 있는 전송 매체를 사용하여 빠른 속도를 필요로 하는 경우, 동시에 여러 사용자에게 데이터를 전달할 경우, 정기적으로 반복해서 전송할 경우에 사용한다.</li> <li>실시간 전송에 유리하며, 신뢰성보다는 속도가 중요시되는 네트워크에서 사용된다.</li> <li>UDP 헤더에는 Source Port Number, Destination Port Number, Length, Checksum 등이 포함된다.</li> </ul>
RTCP(Real-Time Control Protocol)	<ul style="list-style-type: none"> <li>RTP(Real-time Transport Protocol) 패킷의 전송 품질을 제어하기 위한 제어 프로토콜이다.</li> <li>세션(Session)에 참여한 각 참여자들에게 주기적으로 제어 정보를 전송한다.</li> <li>하위 프로토콜은 데이터 패킷과 제어 패킷의 다중화(Multiplexing)를 제공한다.</li> <li>데이터 전송을 모니터링하고 최소한의 제어와 인증 기능만을 제공한다.</li> <li>RTCP 패킷은 항상 32비트의 경계로 끝난다.</li> </ul>



## 5 인터넷 계층의 주요 프로토콜

IP(Internet Protocol)	<ul style="list-style-type: none"> <li>전송할 데이터에 주소를 지정하고, 경로를 설정하는 기능을 한다.</li> <li>비연결형인 데이터그램 방식을 사용하는 것으로 신뢰성이 보장되지 않는다*.</li> </ul>
ICMP(Internet Control Message Protocol, 인터넷 제어 메시지 프로토콜)	IP와 조합하여 통신중에 발생하는 오류의 처리와 전송 경로 변경 등을 위한 제어 메시지를 관리하는 역할을 하며, 헤더는 8Byte로 구성된다.
IGMP(Internet Group Management Protocol, 인터넷 그룹 관리 프로토콜)	멀티캐스트를 지원하는 호스트나 라우터 사이에서 멀티캐스트 그룹 유지를 위해 사용된다.
ARP(Address Resolution Protocol, 주소 분석 프로토콜)	호스트의 IP 주소를 호스트와 연결된 네트워크 접속 장치의 물리적 주소*(MAC Address)로 바꾼다.
RARP(Reverse Address Resolution Protocol)	ARP와 반대로 물리적 주소를 IP 주소로 변환하는 기능을 한다.

### IP의 비신뢰성

비신뢰성이란 패킷이 목적지에 성공적으로 도달하는 것을 보장하지 않는다는 의미입니다. IP는 최선의 서비스를 목적으로 하는 프로토콜로, 신뢰성에 대한 요구는 TCP와 같은 상위 계층에서 제공합니다.

### 물리적 주소(MAC Address)

물리적 주소는 랜카드 제작사에서 랜 카드(네트워크 접속장치)에 부여한 고유 번호입니다.

## 6 네트워크 액세스 계층의 주요 프로토콜

Ethernet(IEEE 802.3)	CSMA/CD 방식의 LAN
IEEE 802	LAN을 위한 표준 프로토콜
HDLC	비트 위주의 데이터 링크 제어 프로토콜
X.25	패킷 교환망을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜
RS-232C	공중 전화 교환망(PSTN)을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜



### 기출문제 따라잡기

Section 156

이전기술

1. TCP/IP의 응용 프로토콜이 아닌 것은?

- ① SNMP
- ② SMTP
- ③ ROS
- ④ FTP

응용 계층 프로토콜의 종류를 기억하라고 했죠? ROS(Read Only Storage)는 읽기만 가능한 저장 매체를 뜻합니다.

이전기술

2. 인터넷 프로토콜 아키텍처를 구성하는 4계층이 아닌 것은?

- ① 표현 계층
- ② 전송 계층
- ③ 인터넷 계층
- ④ 네트워크 액세스 계층

인터넷 프로토콜(TCP/IP) 아키텍처는 응용 계층, 전송 계층, 인터넷 계층, 네트워크 액세스 계층으로 구성됩니다.

▶ 정답 : 1. ③ 2. ①



## 기출문제 따라잡기

Section 156

이전기출

3. TCP/IP에서 사용되는 논리 주소를 물리 주소로 변환시켜주는 프로토콜은?

- ① TCP
- ② ARP
- ③ RARP
- ④ IP

ARP는 '논리 주소 → 물리 주소', RARP는 '물리 주소 → 논리 주소'로 변환합니다. 혼동하지 마세요.

이전기출

4. TCP/IP 프로토콜 중 인터넷 계층에 대응하는 OSI 참조 모델의 계층은?

- ① Physical Layer
- ② Presentation Layer
- ③ Network Layer
- ④ Session Layer

Internet Layer? 아 ~ InterNetwork Layer!

이전기출

5. 호스트의 물리 주소를 통하여 논리 주소인 IP 주소를 얻어오기 위해 사용되는 프로토콜은?

- ① ICMP
- ② IGMP
- ③ ARP
- ④ RARP

이제 ARP와 RARP의 기능을 확실히 구분할 수 있겠죠?

이전기출

6. RTCP(Real-Time Control Protocol)의 특징으로 옳지 않은 것은?

- ① Session의 모든 참여자에게 컨트롤 패킷을 주기적으로 전송한다.
- ② RTCP 패킷은 항상 16비트의 경계로 끝난다.
- ③ 하위 프로토콜은 데이터 패킷과 컨트롤 패킷의 멀티플렉싱을 제공한다.
- ④ 데이터 전송을 모니터링하고 최소한의 제어와 인증 기능을 제공한다.

크기가 정해진 경우에는 크기가 혼동되지 않도록 정확히 기억하고 있어야 합니다. RTCP는 항상 32비트의 경계로 끝납니다.

이전기출

7. 다음과 같은 기능을 가지고 있는 프로토콜은?

- 메시지를 Encapsulation과 Decapsulation 한다.
- 서비스 처리를 위해 Multiplexing과 Demultiplexing을 이용한다.
- 전이중 서비스와 스트림 데이터 서비스를 제공한다.

- ① RTCP
- ② RTP
- ③ UDP
- ④ TCP

"스트림 위주의 데이터 전송" 이것이 TCP하면 가장 먼저 떠올려야 할 내용입니다.

이전기출

8. 인터넷 프로토콜로 사용되는 TCP/IP의 계층화 모델 중 Transport 계층에서 사용되는 프로토콜은?

- ① FTP
- ② IP
- ③ ICMP
- ④ UDP

우리가 학습한 TCP/IP의 계층화 모델 중 전송(Transport) 계층에 속하는 프로토콜은 TCP, UDP, RTCP 이렇게 세 가지입니다.

▶ 정답 : 3. ② 4. ③ 5. ④ 6. ② 7. ④ 8. ④



### 1. 운영체제의 일반적인 역할이 아닌 것은?

- ① 사용자들 간의 하드웨어의 공동 사용
- ② 자원의 효과적인 운영을 위한 스케줄링
- ③ 입 · 출력에 대한 보조 역할
- ④ 실행 가능한 목적(Object) 프로그램 생성

### 2. 운영체제를 계층 구조로 나눌 때 ㉠~㉣에 들어갈 내용이 차례로 옳게 나열된 것은?

하드웨어 - CPU 관리 - (㉠) - (㉡) - (㉢) - 파일 시스템 관리 - 사용자 프로세스

[보기]

- ㉠ 기억장치 관리    ㉡ 주변장치 관리    ㉢ 프로세스 관리

- ① ㉠ - ㉡ - ㉢
- ② ㉠ - ㉢ - ㉡
- ③ ㉡ - ㉢ - ㉠
- ④ ㉡ - ㉠ - ㉢

### 3. 운영체제(Operating System)의 기능으로 옳지 않은 것은?

- ① 컴퓨터의 자원(Resource)들을 효율적으로 관리하는 기능
- ② 입 · 출력에 대한 일을 대행하거나 사용자가 컴퓨터를 손쉽게 사용할 수 있도록 하는 인터페이스 기능
- ③ 사용자가 작성한 원시 프로그램을 기계언어(Machine-Language)로 번역시키는 기능
- ④ 시스템에서 발생하는 오류(Error)로부터 시스템을 보호하는 신뢰성 기능

### 4. UNIX 시스템에서 커널에 대한 설명으로 옳지 않은 것은?

- ① UNIX 시스템의 중심부에 해당한다.
- ② 사용자와 시스템 간의 인터페이스를 제공한다.
- ③ 프로세스 관리, 기억장치 관리 등을 담당한다.
- ④ 하드웨어를 캡슐화한다.

### 5. UNIX 시스템의 특징으로 옳지 않은 것은?

- ① 대화식 운영체제이다.
- ② 소스가 공개된 개방형 시스템이다.
- ③ 멀티 유저, 멀티 태스킹을 지원한다.
- ④ 효과적으로 구현할 수 있는 이중 리스트 구조를 사용한다.

### 6. UNIX에 관한 설명으로 옳지 않은 것은?

- ① 셸(Shell)은 사용자와 시스템 간의 대화를 가능케 해주는 UNIX 시스템의 메커니즘이다.
- ② UNIX 시스템은 루트 노드를 시발로 하는 계층적 파일 시스템 구조를 사용한다.
- ③ 커널은 프로세스 관리, 기억장치 관리, 입 · 출력 관리 등의 기능을 수행한다.
- ④ 커널은 파이프라인 기능을 지원한다.

### 7. 150K의 작업요구 시 First Fit과 Best Fit 전략을 각각 적용할 경우, 할당 영역의 연결이 옳은 것은?

할당영역	운영체제
1	50k
	사용중
2	400k
	사용중
3	200k

- ① First Fit : 2, Best Fit : 3
- ② First Fit : 3, Best Fit : 2
- ③ First Fit : 1, Best Fit : 2
- ④ First Fit : 3, Best Fit : 1

### 8. 가상기억장치 구현 기법에 대한 설명으로 옳지 않은 것은?

- ① 가상기억장치 기법은 말 그대로 가상적인 것으로 현재 실무에서는 실현되는 방법이 아니다.
- ② 가상기억장치를 구현하는 일반적 방법에는 Paging과 Segmentation 기법이 있다.
- ③ 주기억장치의 이용률과 다중 프로그래밍의 효율을 높일 수 있다.
- ④ 주기억장치의 용량보다 큰 프로그램을 실행하기 위해 사용한다.

### 9. 페이징 기법과 세그멘테이션 기법에 대한 설명으로 옳지 않은 것은?

- ① 페이징 기법에서는 주소 변환을 위한 페이지 맵 테이블이 필요하다.
- ② 페이지 크기로 일정하게 나누어진 주기억장치의 단위를 페이지 프레임이라고 한다.
- ③ 페이징 기법에서는 하나의 작업을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행한다.

▶ 정답 : 1. ④ 2. ② 3. ③ 4. ② 5. ④ 6. ④ 7. ① 8. ① 9. ③



## 예 · 상 · 문 · 제 · 은 · 행

- ④ 세그먼테이션 기법을 이용하는 궁극적인 이유는 기억 공간을 절약하기 위해서이다.

**10.** 4개의 페이지를 수용할 수 있는 주기억장치가 있으며, 초기에는 모두 비어 있다고 가정한다. 다음의 순서로 페이지 참조가 발생할 때, FIFO 페이지 교체 알고리즘을 사용할 경우 페이지 결함의 발생 횟수는?

페이지 참조 순서 : 1, 2, 3, 1, 2, 4, 5, 1, 2

- ① 6회                      ② 7회  
③ 8회                      ④ 9회

**11.** 페이지 교체 기법 중 LRU와 비슷한 알고리즘이며, 최근에 사용하지 않은 페이지를 교체하는 기법으로 시간 오버헤드를 줄이기 위해 각 페이지마다 참조 비트와 변형 비트를 두는 교체 기법은?

- ① FIFO                      ② LFU  
③ NUR                      ④ OPT

**12.** 캐시의 라인 교체 정책 가운데, 최근에 가장 적게 사용된 라인부터 교체하는 정책은?

- ① LRU                      ② FIFO  
③ LFU                      ④ LIFO

**13.** 4개의 페이지를 수용할 수 있는 주기억장치가 있으며, 초기에는 모두 비어 있다고 가정한다. 다음의 순서로 페이지 참조가 발생할 때, LRU 페이지 교체 알고리즘을 사용할 경우 몇 번의 페이지 결함이 발생하는가?

페이지 참조 순서 : 1, 2, 3, 1, 2, 4, 1, 2, 5

- ① 5회                      ② 6회  
③ 7회                      ④ 8회

**14.** 페이지징에서 페이지 크기에 관한 고려 사항으로 틀린 것은?

- ① 페이지 크기가 작을수록 페이지 테이블 크기가 커진다.  
② 페이지 크기가 작을수록 좀더 알찬 워킹 셋을 유지할 수 있다.  
③ 페이지 크기가 클수록 실제 프로그램 수행과 무관한 내용이 포함될 수 있다.  
④ 페이지 크기가 클수록 전체적인 입 · 출력이 비효율적이다.

**15.** 구역성(Locality)에 대한 설명으로 옳지 않은 것은?

- ① 실행 중인 프로세스가 일정 시간 동안에 참조하는 페이지의 집합을 의미한다.  
② 시간 구역성과 공간 구역성이 있다.  
③ 캐시 메모리 시스템의 이론적 근거이다.  
④ Denning 교수에 의해 구역성의 개념이 증명되었다.

**16.** 워킹 셋(Working Set)에 대한 설명으로 옳지 않은 것은?

- ① 주기억장치에 적재되지 않으면 스래싱이 발생할 수 있다.  
② 실행 중인 프로세스가 일정 시간 동안 참조하는 페이지들의 집합이다.  
③ 주기억장치에 적재되어야 효율적인 실행이 가능하다.  
④ 프로세스 실행 중에는 크기가 변하지 않는다.

**17.** 프로세스들간의 메모리 경쟁으로 인하여 지나치게 페이지 폴트가 발생하여 전체 시스템의 성능이 저하되는 현상은?

- ① Fragmentation  
② Thrashing  
③ Locality  
④ Prepaging

**18.** 스레드의 특징으로 옳지 않은 것은?

- ① 실행 환경을 공유시켜 기억장소의 낭비가 줄어든다.  
② 프로세스 외부에 존재하는 스레드도 있다.  
③ 하나의 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진시킬 수 있다.  
④ 프로세스들 간의 통신을 향상시킬 수 있다.

**19.** 프로세스의 상태 전이에 속하지 않는 것은?

- ① Dispatch  
② Spooling  
③ Wake Up  
④ Workout

**20.** 프로세스(Process)에 대한 설명으로 옳지 않은 것은?

- ① 트랩 오류, 프로그램 요구, 입 · 출력 인터럽트에 대해 조치를 취한다.  
② 비동기적 행위를 일으키는 주체로 정의할 수 있다.  
③ 실행 중인 프로그램을 말한다.  
④ 프로세스는 각종 자원을 요구한다.

**21. 스레드(Thread)에 대한 설명으로 옳지 않은 것은?**

- ① 프로세스 내부에 포함되는 스레드는 공통적으로 접근 가능한 기억장치를 통해 효율적으로 통신한다.
- ② 다중 스레드 개념을 도입하면 자원의 중복 할당을 방지하고 훨씬 작은 자원만으로도 작업을 처리할 수 있다.
- ③ 하나의 프로세스를 구성하고 있는 여러 스레드들은 공통적인 제어 흐름을 가지며, 각종 레지스터 및 스택 공간들은 모든 스레드들이 공유한다.
- ④ 하나의 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진시킬 수 있다.

**22. 프로세스의 상태 정보를 갖고 있는 PCB(Process Control Block)의 내용이 아닌 것은?**

- ① 프로세스 식별 정보
- ② 프로세스 제어 정보
- ③ 프로세스(CPU) 상태 정보
- ④ 프로세스 생성 정보

**23. 스케줄링의 목적으로 가장 거리가 먼 것은?**

- ① 모든 작업들에 대해 공평성을 유지하기 위하여
- ② 단위 시간당 처리량을 최대화하기 위하여
- ③ 응답 시간을 빠르게 하기 위하여
- ④ 운영체제의 오버헤드를 최소화하기 위하여

**24. 선점 스케줄링과 비선점 스케줄링에 대한 비교 설명 중 옳은 것은?**

- ① 선점 스케줄링은 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없다.
- ② 선점 스케줄링은 상대적으로 과부하가 적다.
- ③ 비선점 스케줄링은 시분할 시스템에 유용하다.
- ④ 비선점 스케줄링은 응답시간의 예측이 용이하다.

**25. 프로세스가 컴퓨터에서 동작하는 방식에 영향을 미치는 동적 인 값들의 모임으로, 시스템이 사용하고자 하는 쉘의 환경을 작업 환경에 맞게 혹은 작업의 필요에 맞추어서 설정하는 데 사용되는 값들을 가지고 있는 것은?**

- ① 매개 변수
- ② 환경 변수
- ③ 측정 변수
- ④ 프로세스 변수

**26. 다음 중 UNIX에서 사용하는 환경 변수가 아닌 것은?**

- ① ProgramFiles                      ② DISPLAY
- ③ TERM                                ④ PS1

**27. UNIX 명령 중 Windows 명령 'type'과 유사한 기능을 갖는 것은?**

- ① cp                                      ② cat
- ③ ls                                      ④ rm

**28. UNIX에서 파일 내용을 화면에 표시하는 명령과 파일의 소유자를 변경하는 명령을 순서적으로 옳게 나열한 것은?**

- ① dup, mkfs                          ② cat, chown
- ③ type, chmod                        ④ type, cat

**29. MS-DOS 명령어 중 디스크 표면을 트랙과 섹터로 나누어 초기화하는 것은?**

- ① FORMAT                            ② CHKDSK
- ③ SYS                                  ④ ATTRIB

**30. 파일을 삭제하는 UNIX 명령은?**

- ① rm                                      ② delete
- ③ mkdir                                ④ mv

**31. IPv6의 주소 체계로 거리가 먼 것은?**

- ① Unicast                                ② Anycast
- ③ Broadcast                            ④ Multicast

**32. OSI 7계층 모델에서 전송에 필요한 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성을 정의한 계층은?**

- ① 물리 계층
- ② 데이터 링크 계층
- ③ 네트워크 계층
- ④ 전송 계층

**33. OSI 7계층의 데이터 링크 계층 기능에 대한 설명으로 적절하지 않은 것은?**

- ① 최적의 경로를 설정하기 위한 경로 설정 기능
- ② 오류의 검출과 회복을 위한 오류 제어 기능
- ③ 프레임의 순서적 전송을 위한 순서 제어 기능
- ④ 프레임의 시작과 끝을 구분하기 위한 프레임의 동기화 기능



- ① OSI 7계층 중 하위 3계층과 상위 3계층의 인터페이스를 담당한다.
- ② 응용 프로세스에게 일정한 전송 품질(Qos)을 제공하기 위한 기능이다.
- ③ 경로 설정(Routing), 데이터 교환 및 중계, 트래픽 제어, 패킷 정보 전송을 수행한다.
- ④ 네트워크 타입(Type)에 따라 다양한 서비스의 품질(Qos)을 제공한다.

- ① 데이터 링크 계층
- ② 네트워크 계층
- ③ 세션 계층
- ④ 표현 계층

- ① 데이터 링크 계층
- ② 물리 계층
- ③ 응용 계층
- ④ 세션 계층

① 라우터                  ② 브리지  
③ 허브                    ④ 리피터

- ① 인터페이스(Interface) : 통신장치와 사람 사이의 상호 작용을 위한 수단을 규정
- ② 구문(Syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
- ③ 의미(Semantics) : 두 개체 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
- ④ 시간(Timing) : 두 개체 간의 통신 속도, 메시지의 순서 제어 등을 규정

- ① 단편화(Fragmentation)
- ② 캡슐화(Encapsulation)
- ③ 동기화(Synchronization)
- ④ 다중화(Multiplexing)

- ① 비연결형 전송 서비스 제공
- ② 비신뢰성 전송 서비스 제공
- ③ 데이터그램 전송 서비스 제공
- ④ 스트림 전송 계층 서비스 제공

① PMA                      ② ICMP  
③ ARP                     ④ IP

**1. Section 141**

목적 프로그램을 생성하는 것은 컴파일러(Compiler)의 역할이다.

**3. Section 141**

③번은 언어 번역 프로그램의 기능이다.

**4. Section 143**

사용자와 시스템 간의 인터페이스를 제공하는 것은 셸(Shell)이다.

**5. Section 143**

UNIX는 트리(Tree) 구조를 사용한다.

**6. Section 143**

파이프라인 기능을 지원하는 것은 셸(Shell)이다.

**7. Section 144**

150K 작업을 최초 적합(First Fit)으로 할당할 경우 400K 공백에, 최적 적합(Best Fit)으로 할당할 경우 200K 공백에, 최악 적합(Worst Fit)으로 할당할 경우 400K 공백에 할당된다.

**8. Section 146**

가상기억장치는 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법으로, 현재 실무에서 유용하게 사용되고 있는 기법이다.

**9. Section 146**

페이징 기법에서는 하나의 작업을 동일한 크기로 나눈 후 주기억장치에 적재시켜 실행한다. ③번의 내용은 세그먼테이션 기법에 대한 설명이다.

**10. Section 146**

4개의 페이지를 수용할 수 있는 주기억장치이므로 아래 그림과 같이 4개의 페이지 프레임으로 표현할 수 있다.

참조 페이지	1	2	3	1	2	4	5	1	2
페이지 프레임	1	1	1	1	1	1	5	5	5
		2	2	2	2	2	2	1	1
			3	3	3	3	3	3	2
						4	4	4	4
부재 발생	●	●	●			●	●	●	●

※ ● : 페이지 부재 발생

참조 페이지가 페이지 테이블에 없을 경우 페이지 결함(부재)이 발생된다. 초기에는 모든 페이지가 비어 있으므로 처음 1, 2, 3 페이지 적재 시 페이지 결함이 발생된다. 다음 참조 페이지 1, 2는 이미 적재되어 있

으므로 그냥 참조한다. FIFO 기법은 각각의 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법이므로, 참조 페이지 5를 참조할 때에는 1을 제거한 후 5를 가져오게 된다. 이러한 과정으로 모든 페이지에 대한 요구를 처리하고 나면 총 페이지 결함 발생 횟수는 7회이다.

**11. Section 146**

- FIFO : 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
- LFU : 사용 빈도가 가장 적은 페이지를 교체하는 기법
- OPT : 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법

**13. Section 146**

4개의 페이지를 수용할 수 있는 주기억장치이므로 아래 그림과 같이 4개의 페이지 프레임으로 표현할 수 있다.

참조 페이지	1	2	3	1	2	4	1	2	5
페이지 프레임	1	1	1	1	1	1	1	1	1
		2	2	2	2	2	2	2	2
			3	3	3	3	3	3	5
						4	4	4	4
부재 발생	●	●	●			●			●

※ ● : 페이지 부재 발생

참조 페이지가 페이지 테이블에 없을 경우 페이지 결함(부재)이 발생된다. 초기에는 모든 페이지가 비어 있으므로 처음 1, 2, 3 페이지 적재 시 페이지 결함이 발생된다. 다음 참조 페이지 1, 2는 이미 적재되어 있으므로 그냥 참조하고, 참조 페이지 4를 적재할 때 페이지 결함이 발생된다. 다음 참조 페이지 1, 2 역시 이미 적재되어 있으므로 그냥 참조한다. LFU 기법은 최근에 가장 오랫동안 사용되지 않은 페이지를 교체하는 기법이므로, 마지막 페이지 5를 참조할 때는 3을 제거한 후 5를 가져오게 된다. 그러므로 총 페이지 결함 발생 횟수는 5회이다.

**14. Section 147**

페이지의 크기가 클 경우 적은 수의 페이지가 존재하게 되어 작은 페이지 테이블 공간을 필요로 하고, 참조되는 정보와 무관한 많은 양의 정보가 주기억장치에 남게 되며, 전체적인 입·출력 효율이 증가된다.

**15. Section 147**

실행 중인 프로세스가 일정 시간 동안에 참조하는 페이지의 집합은 워킹 셋(Working Set)이다.

**16. Section 147**

워킹 셋(Working Set)은 자주 참조하는 페이지들의 집합을 의미하는 것으로, 스래싱이 발생되지 않게 하기 위해서 워킹 셋을 계속하여 변경해 주기 때문에 프로세스 실행중에





워킹 셋의 크기가 변경될 수 있다.

### 18. Section 148

스레드(Thread)는 프로세스 내에서의 작업 단위로 외부에 존재하는 스레드는 없다.

### 20. Section 148

트랩 오류, 프로그램 요구, 입·출력 인터럽트에 대해 조치를 취하는 것은 운영체제의 역할이다.

### 21. Section 148

스레드는 독립된 제어 흐름을 갖고, 고유의 레지스터와 스택을 사용한다.

### 22. Section 148

PCB에 저장되어 있는 정보

- 프로세스의 현재 상태 : 준비, 대기, 실행 등의 프로세스 상태
- 포인터 : 부모 프로세스에 대한 포인터, 자식 프로세스에 대한 포인터, 프로세스가 위치한 메모리에 대한 포인터, 할당된 자원에 대한 포인터
- 프로세스 고유 식별자 : 프로세스를 구분할 수 있는 고유의 번호
- 스케줄링 및 프로세스의 우선순위 : 스케줄링 정보 및 프로세스가 실행될 우선순위
- CPU 레지스터 정보
- 주기억장치 관리 정보
- 입·출력 상태 정보
- 계정 정보

### 23. Section 149

스케줄링은 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업을 의미하는 것으로, 운영체제의 오버헤드를 최소화하기 위해 사용한다.

### 24. Section 149

- ① 선점 스케줄링은 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 있다.
- ② 선점 스케줄링은 많은 과부하(Overhead)를 초래한다.
- ③ 비선점 스케줄링은 일괄 처리 시스템에 유용하며, 시분할 시스템에 유용한 것은 선점 스케줄링이다.

### 26. Section 150

ProgramFiles는 Windows에서 기본 프로그램이 설치되어 있는 폴더를 지정하는 환경 변수이다.

### 27. Section 151

TYPE은 파일의 내용을 표시하는 Windows 명령으로 UNIX에서 cat과 같은 명령어이다.

- cp : 파일 복사
- ls : 현재 디렉토리 내의 파일 목록 확인
- rm : 파일 삭제

### 28. Section 151

- mkfs : 파일 시스템 생성
- type : 파일의 내용을 표시하는 명령어로, Windows 명령어임
- chmod : 파일의 보호 모드를 설정하여 파일의 사용 허가 지정

### 29. Section 151

- CHKDSK : 디스크 상태 점검
- SYS : 시스템 파일 복사
- ATTRIB : 파일 속성 변경

### 30. Section 151

- mkdir : 디렉터리 생성
- mv : 파일 이동 또는 이름 변경

### 31. Section 152

IPv6 주소의 주소 체계

- 유니캐스트(Unicast) : 단일 송신자와 단일 수신자 간의 통신(1 대 1 통신에 사용)
- 멀티캐스트(Multicast) : 단일 송신자와 다중 수신자 간의 통신(1 대 다 통신에 사용)
- 애니캐스트(Anycast) : 단일 송신자와 가장 가까이 있는 단일 수신자 간의 통신(1 대 1 통신에 사용)

### 32. Section 153

물리 계층(Physical Layer)

- 전송에 필요한 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성을 정의한다.
- 물리적 전송 매체와 전송 신호 방식을 정의한다.
- RS-232C, X.21 등의 표준이 있다.





### 33. Section 153

경로 설정은 네트워크 계층의 기능이다.

### 34. Section 153

경로 설정(Routing), 데이터 교환 및 중계, 트래픽 제어, 패킷 정보 전송을 수행하는 것은 네트워크 계층의 기능이다.

### 35. Section 153

- 데이터 링크 계층(Data Link Layer) : 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 함
- 네트워크 계층(Network Layer, 망 계층) : 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 기능을 함
- 표현 계층(Presentation Layer) : 응용 계층으로부터 받은 데이터를 세션 계층에 보내기 전에 통신에 적당한 형태로 변환하고, 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능을 함

### 36. Section 153

- 데이터 링크 계층(Data Link Layer) : 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 함
- 물리 계층(Physical Layer) : 전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의함
- 세션 계층(Session Layer) : 송·수신 측 간의 관련성을 유지하고 대화 제어를 담당함

### 37. Section 154

- 브리지(Bridge) : LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹(세그먼트)을 연결하는 기능을 수행하며, 데이터 링크 계층에서 동작함
- 허브(Hub) : 네트워크를 구성할 때 한꺼번에 여러 대의 컴퓨터를 연결하는 장치로, 각 회선을 통합적으로 관리함
- 리피터(Repeater) : 전송되는 신호가 전송 선로의 특성 및 외부 충격 등의 요인으로 인해 원래의 형태와 다르게 왜곡되거나 약해질 경우 원래의 신호 형태로 재생하여 다시 전송하는 역할을 수행하며, 물리 계층에서 동작함

### 38. Section 155

프로토콜의 구성 요소는 구문(Syntax), 의미(Semantics), 시간(Timing)이다.

### 39. Section 155

- 캡슐화(Encapsulation) : 단편화된 데이터에 주소, 오류 검출 코드, 프로토콜 제어 정보 등을 추가하는 작업
- 동기화(Synchronization) : 송·수신 측이 같은 상태를 유지하도록 타이밍(Timing)을 맞추는 기능
- 다중화(Multiplexing) : 한 개의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능

### 40. Section 156

UDP는 신뢰성이 떨어지므로, 신뢰성보다는 속도가 중요시 되는 네트워크에서 사용된다.

### 41. Section 156

스트림(Stream) 전송 기능을 제공하는 것은 TCP이다.

### 42. Section 156

IP 계층, 즉 인터넷 계층에 속하는 프로토콜에는 IP, ICMP, IGMP, ARP, RARP 등이 있다.