**4과목 프로그래밍 언어 활용**

1장 서버 프로그램 구현

127 개발 환경 구축

개발 환경 구축 – 응용 소프트웨어 개발을 위해 개발 프로젝트를 이해하고 소프트웨어 및 하드웨어 장비를 구축하는 것을 의미

하드웨어 환경 – 사용자와의 인터페이스 역할을 하는 클라이언트와 클라이언트와 통신하여 서비스를 제공하는 서버로 구성됨

서버 – 웹 서버 / 웹 애플리케이션 서버 / 데이터베이스 서버 / 파일 서버

웹 서버 기능 – HTTP/HTTPS 지원 / 통신 기록 / 정적 파일 관리 / 대역폭 제한 / 가상 호스팅 / 인증

소프트웨어 환경 – 클라이언트와 서버 운영을 위한 시스템 소프트웨어와 개발에 사용되는 개발 소프트웨어로 구성

시스템 소프트웨어 – 운영체제, 웹 서버, WAS 운용을 위한 서버 프로그램, DBMS

개발 소프트웨어 – 요구사항 관리 도구 / 설계/모델링 도구 / 구현 도구 / 빌드 도구 / 테스트 도구 / 형상 관리 도구

개발 언어 선정 기준 – 적정성 / 효율성 / 이식성 / 친밀성 / 범용성

128 서버 개발

- 웹 애플리케이션의 로직을 구현할 서버 프로그램을 제작하여 웹 애플리케이션 서버에 탑재하는 것을 의미

Java / JavaScropt / Python / PHP / Ruby

서버 개발 프레임워크 – 서버 프로그램 개발 시 다양한 네트워크 설정, 요청 및 응답 처리, 아키텍처 모델 구현 등을 손쉽게 처리할 수 있도록 클래스나 인터페이스를 제공하는 소프트웨어

Spring / Node.js / Django / Codeniter / Ruby on Rails

서버 프로그램 – 응용 소프트웨어와 동일하게 모듈 및 공통 모듈을 개발한 후 모듈들을 통합하는 방식으로 구현

모듈의 독립성은 결합도와 응집도에 의해 측정

독립성을 높이려면 모듈의 결합도를 약하게 하고 응집도를 강하게 하여 모듈의 크기를 작게 만들어야 함

129 보안 및 API

소프트웨어 개발 보안 - 소프트웨어 개발 과정에서 발생할 수 있는 보안 취약점을 최소화하여 보안 위협으로부터 안전한 소프트웨어를 개발하기 위한 일련의 보안 활동을 의미

데이터의 기밀성 / 무결성 / 가용성을 유지하는 것이 목표

소프트웨어 개발 보안 점검 항목

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

API – 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스

130 배치 프로그램

- 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하는 것

정기 배치 / 이벤트성 배치 / On-Demand 배치

배치 프로그램이 갖추어야 하는 필수 요소 – 대용량 데이터 / 자동화 / 견고성 / 안정성/신뢰성 / 성능

배치 스케줄러 – 일괄 처리 작업이 설정된 주기에 맞춰 자동으로 수행되도록 지원해주는 도구

스프링 배치 – Job / Job Launcher / Step / Job Repository

Quartz – Scheduler / Job / JobDetail /Trigger

131 패키지 소프트웨어

- 기업에서 일반적으로 사용하는 여러 기능들을 통합하여 제공하는 소프트웨어를 의미

요구사항을 분석하여 업무 특성에 맞게 전용으로 개발되는 소프트웨어와 비교하여 안정성, 라이선스, 생산성 등에서 차이가 있음

문제정리

소프트웨어 환경 – 시스템 소프트웨어 / 개발 소프트웨어

가상 호스팅 – 하나의 서버로 여러 개의 도메인 이름을 연결하는 기능

대역폭 제한 – 네트워크 트래픽의 포화를 방지하기 위해 응답 속도를 제한하는 기능

개발 언어 선정 기준 – 적정성 / 효율성 / 이식성 / 친밀성

패키지 소프트웨어 – 결함이 발생했을 때 판매처의 프로세스에 따라 보안되므로 이용자의 사저엥 따라 즉각적이고 능동적인 대처가 어려움

전용 개발 소프트웨어 – 사업 환경에 맞추어 직접 개발한 소프트웨어

라이브러리 – 개발 편의를 위해 자주 사용되는 코드

통합 개발 환경 도구 – 개발에 필요한 편집기, 컴파일러, 디버거 등의 다양한 툴을 하나의 인터페이스로 제공하는 소프트웨어

소프트웨어 개발 보안 점검 항목 – 세션 통제 / 입력 데이터 검증 및 표현 / 보안 기능 / 시간 및 상테 / 에러처리 / 코드 오류 / 캡슐화 / API 오용

입력 데이터 검증 및 표현 – 입력 데이터에 대한 유효성 검증체계를 갖추고 검증 실패 시 이를 처리할 수 있도록 코딩하는 것

캡슐화 – 데이터와 데이터를 처리하는 함수를 하나의 객체로 묶어 코딩하는 것

스프링 배치 주요 구성 요소 – Job / JobLauncher / Step / JobRepository

Quartz – 오류가 발생하면 오류의 발생 위치, 시간 등을 추적할 수 있는 안정성, 신뢰성을 필수적으로 갖추고 있음

API – 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스

응용 프로그램 – 특정 업무를 처리하기 위해 만들어진 프로그램

ORM – 객체지향 프로그래밍의 객체와 관계형 데이터베이스의 데이터를 연결하는 기술

프로그래밍 언어의 선정 기준

친밀감 / 언어의 능력 / 처리의 효율성 / 프로그램 구조 / 프로그램 길이 / 이식성 / 과거의 개발 실적 / 알고리즘과 계산상의 난이도 / 자료 구조의 난이도 / 성능 고려 사항들 / 대상 업무의 성격 / 소프트웨어의 수행 환경 / 개발 담당자의 경험과 지식 / 사용자의 요구사항 / 컴파일러의 이용 가능성

키워드 : 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

2장 프로그래밍 언어 활용

132 데이터 타입

- 변수에 저장될 데이터의 형식을 나타내는 것

변수에 값을 저장하기 전에 문자형, 정수형, 실수형 등 어떤 형식의 값을 저장할지 데이터 타입을 지정하여 변수를 선언해야 함

데이터 타입 유형

정수 타입 – 정수, 즉 소수점이 없는 숫자를 저장할 때 사용

부동 소수점 타입 – 소수점 이하가 있는 실수를 저장할 때 사용

문자 타입 – 한 문자를 저장할 때 사용 / 작은따옴표 안에 표시

문자열 타입 – 문자열을 저장할 때 사용 / 큰 따옴표 안에 표시

불린 타입 – 조건의 참, 거짓 여부를 판단하여 저장할 때 사용 / 기본 값은 거짓

배열 타입 – 같은 타입의 데이터 집합을 만들어 저장할 때 사용 / 데이터는 중괄호 안에 콤마로 구분하여 값들을 나열

C/C++의 데이터 타입 크기 및 기억 범위

문자 char 1Byte / 부호없는 문자형 unsigned char 1Byte / 정수 short 2Byte / 정수 int 4Byte / 정수 long 4Byte / 정수 long long 8Byte / 부호 없는 정수형 unsigned short 2Byte / 부호 없는 정수형 unsigned int 4Byte / 부호 없는 정수형 unsigned ling 4Byte / 실수 float 4Byte / 실수 double 8Byte / 실수 long double 8Byte

JAVA의 데이터 타입 크기 및 기억 범위

문자 char 2Byte / 정수 byte 1Byte / 정수 short 2Byte / 정수 int 4Byte / 정수 long 8Byte / 실수float 4Byte / 실수 double 8Byte / 논리 Boolean 1Byte

Python의 데이터 타입 크기 및 기억 범위

문자 str 무제한 / 정수 int 무제한 / 실수 float 8Byte / 실수 complex 16Byte

133 변수

- 컴퓨터가 명령을 처리하는 도중 발생하는 값을 저장하기 위한 공간으로 변할 수 있는 값을 의미

변수명 작성 규칙

영문자, 숫자, \_ 사용 가능

첫 글자 영문자 or \_ / 숫자X

글자수 제한 X

공백이나 특수문자 사용 X

대 소문자 구분

예약어 변수명으로 사용X

변수 선언 시 문장 끝 ;

기억 클래스 – 자동 변수 / 레지스터 변수 / 정적 변수(내부) / 정적 변수(외부) / 외부 변수

자동 변수 – 함수나 코드의 범위를 한정하는 블록 내에서 선언되는 변수

외부 변수 – 현재 파일이나 다른 파일에서 선언된 변수나 함수를 참조하기 위한 변수

정적 변수 – 함수나 블록 내에서 선언하는 내부 정적 변수와 함수 외부에서 선언하는 외부 정적 변수

레지스터 변수 – 메모리가 아닌 CPU 내부의 레지스터에 기억영역을 할당받는 변수

변수 선언

자료형 : 변수에 저장될 자료의 형식을 지정

변수형 : 사용자가 원하는 이름을 임의로 지정 / 변수명 작성 규칙에 맞게 지정

값 : 변수를 선언하면서 초기화할 값을 지정

134 연산자

산술 연산자 – 산술 계산에 사용되는 연산자

+ 덧셈 / - 뺄셈 / \* 곱셈 / / 나눗셈 / % 나머지 / ++ 증가 연산자 / -- 감소연산자

관계 연산자 – 두 수의 관계를 비교하여 참 또는 거짓을 결과로 얻는 연산자

== 같다 / != 같지 않다 / > 크다 / >= 크거나 같다 / < 작다 / <= 작거나 같다

비트 연산자 – 비트별(0,1)로 연산하여 결과를 얻는 연산자

& and / ^ xor / ! or / ~ not / << 왼쪽 시프트 / >> 오른쪽 시프트

논리 연산자 – 두 개의 논리 값을 연산하여 참 또는 거짓을 결과로 얻는 연산자

! not / && and / || or

대입 연산자 – 연산 후 결과를 대입하는 연산식을 간략하게 입력할 수 있도록 대입 연산자를 제공

+= / -= / \*= / /= / %= / <<= / >>=

조건 연산자 – 조건에 따라 서로 다른 수식을 수행

조건 ? 수식1:수식2;

135 데이터 입•출력

scanf() – 키보드로 입력받아 변수에 저장하는 함수

scanf(서식 문자열, 변수의 주소)

서식 문자열 - %d / %u/ %o / %x / %c / %s / %f / %e / %d / %e / %ld / %lo / %lx / %p

printf() – 인수로 주어진 값을 화면에 출력하는 함수

printf(서식 문자열, 변수)

주요 제어 문자 - \n / \b / \t / \r / \0 / \’ / \” / \a / \\ / \f

기타 표준 입•출력 함수

getchar() / gets() – 입력

putchar() / puts() – 출력

136 제어문

- 프로그램의 순서를 변경할 때 사용하는 명령문

단순 if – 조건이 한 개일 때 사용하는 제어문

형식1: 조건이 참일 때만 실행

if(조건)

실행할 문장;

if(조건) {

실행할 문장1;

실행할 문장 2 ;

… }

형식2 : 조건이 참일 대와 거짓일 때 실행할 문장이 다르다

if(조건)

실행할 문장1;

else

실행할 문장2;

다중if – 조건이 여러 개 일 때 사용하는 제어문

if(조건1)

실행할 문장1;

else if(조건2)

실행할 문장2;

…

else

실행할 문장3;

형식2: if문 안에 if문이 포함된다

if(조건1) {

if(조건2)

실행할 문장1;

else

실행할 문장2; }

else

실행할 문장3;

switch – 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리할 수 있는 제어문

switch(수식)

{

case 레이블1:

실행할 문장1;

break;

case 레이블2:

실행할 문장2;

break;

default:

실행할 문장3;

}

goto - 프로그램 실행 중 현재 위치에서 원하는 다른 문장으로 건너뛰어 수행을 계속하기 위해 사용하는 제어문

goto 레이블;

레이블:

실행할 문장

137 반복문

- 한 종류로 일정한 횟수를 반복하는 명령문

for – 초기값, 최종값, 증가값을 지정하는 수식을 이용해 정해진 횟수를 반복하는 제어문

for(식1; 식2; 식3)

실행할 문장;

while – 조건이 참인 동안 실행할 문장을 반복 수행하는 제어문

while(조건)

실행할 문장;

do~while – 실행할 문장을 우선 실행한 후 조건을 판별하여 조건이 참이면 실행할 문장을 계속 반복 수행하고 거짓이면 do~while문을 끝낸 후 다음 코드를 실행

do

실행할 문장;

while(조건);

break : switch문이나 반복문 안에서 break가 나오면 블록을 벗어남

continue – continue 이후의 문장을 실행하지 않고 제어를 반복문의 처음으로 옮김

138 배열과 문자열

배열 – 동일한 데이터 유형을 여러 개 사용해야 할 경우 이를 손쉽게 처리하기 위해 여러 개의 변수들을 조합해서 하나의 이름으로 정의해 사용하는 것

1차원 배열 – 변수들을 일직선상의 개념으로 조합한 배열

자료형 변수명[개수];

2차원 배열 – 변수들의 평면 / 행과 열로 조합한 배열

자료형 변수명[행개수][열개수]

배열은 선언 시 초기값을 지정할 수 있음

char 배열이름[크기]=”문자열”

“ “ 사용하면 문자열로 지정

139 포인터

- 변수의 주소를 말하며 C언어에서는 주소를 제어할 수 있는 기능을 제공함

-> 연결된 자료 구조를 구성하기 위해 사용

-> 동적으로 할당된 자료 구조를 지정하기 위해 사용

-> 배열을 인수로 전달하기 위해 사용

-> 문자열을 표현하기 위해 사용

-> 커다란 배열에서 요소를 효율적으로 저장하기 위해 사용

-> 메모리에 직접 접근하기 위해 사용

배열은 포인터 변수에 저장한 후 포인터를 이용해 배열의 요소에 접근할 수 있음

140 Python의 기초

- 변수의 자료형에 대한 선언이 없음

- ; 을 사용할 필요가 없음

- 변수에 연속하여 값을 저장하는 것이 가능

input() 함수

- 키보드로 입력받아 변수에 저장하는 함수

형식1

변수 = input(출력문자)

형식2

변수1, 변수2, … = input(출력문자).split(분리문자)

print() 함수

형식1

print(출력값1, 출력값2, … ,sep = 분리문자, end = 종료문자)

형식2

print(서식 문자열 \* % (출력값1, 출력값2, … ))

리스트

형식

리스트명 = [값1, 값2, …]

리스트명 = list([값1,값2, … ])

딕셔너리

형식

딕셔너리명 = {키1:값1, 키2:값2, … }

딕셔너리명 = dict({키1:값1, 키2:값2,…})

range – 연속된 숫자를 생성하는 것으로 리스트, 반복문에서 많이 사용

형식

range(최종값)

range(초기값, 최종값)

range(초기값, 최종값, 증가값)

슬라이스 – 문자열이나 리스트와 같은 순차형 객체에서 일부를 잘라 반환하는 기능

객체명[초기위치:최종위치]

객체명[초기위치:최종위치:증가값]

객체명[:] or 객체명[::]

객체명[초기위치:]

객체명[:초기위치]

객체명[::증가값]

141 Python 활용

if문

if 조건:

실행할 문장

- 조건이 참일 때만 실행

if 조건:

실행할 문장1

else:

실행할 문장2

- 조건이 참일 때와 거짓일 때 실행할 문장

if 조건1:

실행할 문장1

elif 조건2:

실행할 문장2

else:

실행할 문장3

- 조건이 여러 개이고 조건마다 실행할 문장이 다름

if 조건1:

if 조건2:

실행할 문장1

else:

실행할 문장2

else:

실행할 문장3

- if문 안에 if문 포함

for문

for 변수 in range(최종값):

실행할 문장

- range를 이용하는 방식

for 변수 in 리스트

실행할 문장

- 리스트를 이용하는 방식

while문

while 조건:

실행할 문장