4과목-프로그래밍 언어 활용

(Part 2. 프로그래밍 언어 활용-3)

프로그래밍 언어 활용 총 파트

프로그래밍 언어 활용 4과목은 총 3Part로 이루어져 있다.

1장 서버 프로그램 구현(0.69%)

2장 프로그래밍 언어 활용(44.83%)

3장 응용 SW 기초 기술 활용(54.48%)

프로그래밍 언어 활용

프로그래밍 언어 활용 Part는 17개의 섹션으로 구성되어 있다.

001 데이터 타입
002 변수
003 연산자
004 데이터 입·출력
005 제어문
006 반복문
007 배열과 문자열
008 포인터
009 Python의 기초
010 Python의 활용
011 절차적 프로그래밍 언어
012 객체지향프로그래밍 언어
013 스크립트 언어
014 선언형 언어
015 라이브러리
016 예외 처리
017 프로토타입

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

1) 배열의 개념

; 배열은 동일한 데이터 유형을 여러 개 사용해야 할 경우 이를 손쉽게 처리하기 위해 여러 개의 변수들을 합쳐서 하나의 이름(배열명)으로 정의해 사용하는 것을 말한다.

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

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

2) 1차원 배열

; 1차원 배열은 변수들을 일직선상의 개념으로 조합한 배열이다.

● 배열은 하나의 이름으로 여러 기억장소를 가리키기 때문에 배열에서 개별적인 요소들의 위치는 첨자(인덱스)를 이용하여 지정한다.

● 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]

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

* a[3] : a는 배열의 이름이고, 3은 첨자(인덱스)로서 배열 a에서의 위치를 나타낸다. [2]에 4를 저장시키려면 'a[2] = 4'와 같이 작성한다.

배열명은 배열의 시작 주소를 가리키지만, 배열명[첨자]는 변수와 동일하게 취급된다.

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

2) 1차원 배열

예제1) 1차원 배열 a의 각 요소에 10, 11, 12, 13, 14를 저장한 후 출력하기

```c
#include <stdio.h>
main() {
    int a[5] = {0};

    int i = 0;
    //반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안
    //5번 문장을 반복하여 수행한다.
    for (i = 0; i < 5; i++)
        a[i] = i + 10; //배열 a의 1번째에 1+10을 저장시킨다. 오는 0~4까지 변하므로 배열
                       //a에 저장된 값은 10, 11, 12, 13, 14이다.
    //배열 값 출력
    for (i = 0; i < 5; i++)
        printf("%d ", a[i]);
}
```

결과 10 11 12 13 14

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

2) 1차원 배열

JAVA에서의 배열 처리

; JAVA에서는 향상된 for문을 사용할 수 있는데, 향상된 for문은 객체를 대상으로만 가능하다. JAVA에서는 배열을 객체로 취급하며, 배열을 이용하여 작업할 때 필요할 만한 내용은 이미 API로 만들어 두었기 때문에 잘 골라서 사용하면 된다. 배열에 대한 기본은 C 언어에서 배웠기 때문에 바로 예제를 보면서 설명하겠다.

예제1) 다음은 앞의 C 프로그램을 JAVA로 구현한 것이다. 프로그램의 출력 결과를 확인해 보시오.

```
public class Example {

    public static void main(String[] args) {

        int[] a = new int[5];
        int i = 0;
        for (i = 0; i < 5; i++)
            a[i]  = i + 10;
        for (i = 0; i<5; i++)
            System.out.printf("%d ", a[i]);
    }
}
```

객체(참조) 변수
객체(참조) 변수, 정확히 말하면 메모리 heap 영역에 객체를 생성 하고 생성된 객체가 있는 곳의 메모리 주소를 객체(참조) 변수에 저장하는 것이다. JAVA에서는 주소를 제어할 수 없으므로 그냥 객체 변수를 생성한다고 이해해도 된다.

배열을 선언하는 부분이 조금 다르다. 배열은 JAVA에서 객체로 취급되며, 객체(참조) 변수는 'new' 명령을 이용해서 생성해야 하며, 그렇게 되면 배열이 다 0으로 초기화 된다.

결과 10 11 12 13 14

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

2) 1차원 배열

JAVA에서의 배열 처리

예제2) 다음은 JAVA에서 향상된 for문을 사용한 예제이다. 결과를 확인하시오.

```
public class Example {

    public static void main(String[] args) {

        int[] arr = {90, 100, 80, 70, 60, 50, 30};
        int hap = 0;
        float avg = 0.0f;

        for (int i : arr)
            hap = hap + i;

        avg = (float)hap /arr.length;
        System.out.printf("%d, %.2f", hap, avg);
    }
}
```

결과 480, 68.57

위의 배열을 선언하면서 초기값을 지정했다. 개수를 정하지 않으면 초기값으로 지정된 수만큼 배열의 요소가 만들어진다. 이건 C언어와 동일하다.
향상된 반복문입니다. arr 배열의 개수 만큼 7번을 반복 수행한다.
•int i : arr 배열의 각 요소가 일시적으로 저장될 변수를 선언한다. arr 배열과 자료형이 같아야 한다.
arr 배열이 정수면 정수, 문자면 문자여야 한다.
•arr : 배열의 이름을 입력한다. 즉, 요소를 가져올 장소를 지정하는 것이다. arr 배열이 7개의 요소를 가지므로 각 요소를 i에 저장하면서 7번 수행한다.
C언어에서는 향상된 for문을 지원하지 않는다.

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

3) 2차원 배열

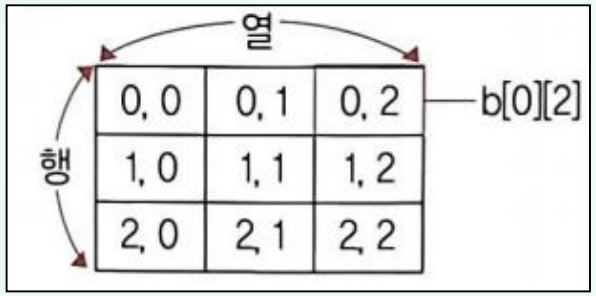
; 2차원 배열은 변수들을 평면(표), 즉 행과 열로 조합한 배열이다.

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

배열 b



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

* 2차원 배열에서는 2차원 배열명도, 1차원 배열명도 모두 주소인 것을 기억하자.

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

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

방법1) char a[3] = {'A', 'B', C'};

방법2) char a[] = {'A', 'B', 'C'};

배열 a	A	B	C
	a[0]	a[1]	a[2]

예) 2차원 배열 초기화

방법1) int a[2][4] = { {10, 20, 30, 40}, {50, 60, 70, 80} };

방법2) 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]

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

4) 배열의 초기화

예제) 3행 4열의 배열에 다음과 같이 숫자 저장하기

배열 a

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

```c
#include <stdio.h>
main() {
    int a[3][4] = {0};
    int i = 0, j = 0, k = 0;
    //2차원 배열이니깐 배열 값에 접근하기 위해서는
    //더블루프가 사용된다.
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 4; j++) {
            k++;
            a[i][j] = k;
        }
    }
}
```

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

4) 배열의 초기화

● 배열의 개수보다 적은 수로 배열을 초기화하면 입력된 값만큼 지정한 숫자가 입력되고, 나머지 요소에는 0이 입력된다.

예) int a[5] = { 3, }; 또는 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

```
main() {
    int a[2][4] = {
        {10, 20, 30, 40},
        {50, 60, 70, 80}
    };
    printf("%d ", a[0][0]);
    printf("%d ", a[1][1]);
}
```

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

결과 10 60

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

5) 배열 형태의 문자열 변수

- C 언어에서는 큰따옴표(")로 묶인 글자는 글자 수에 관계없이 문자열로 처리된다.
- C 언어에는 문자열을 저장하는 자료형이 없기 때문에 배열, 또는 포인터를 이용하여 처리한다.

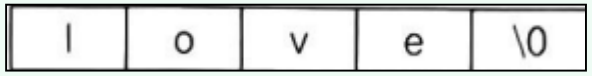
형식

```
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" ; ->



4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

5) 배열 형태의 문자열 변수

예제) 다음의 출력 결과를 확인하시오.

```
main() {
    char a = 'A';
    char b[9] = "KOREA@@@";
    char *c = "KOREA!!!";
    //b(배열명)는 포인터 상수이기에 주소값을 바꿀 수 없다.
    b = c;
    c = b;
    char *d = b;
    char *e = c;
    d[2] = 'a';
    //c[2] = 'a';
    //e[2] = 'a';
    printf("%c₩n", a);
    printf("%s₩n", b);
    printf("%s₩n", c);
    printf("%s₩n", d);
    printf("%s₩n", e);
}
```

```
A
KOaEA@@@
KOaEA@@@
KOaEA@@@
KOaEA@@@
```

```
char b[9] = "KOREA!!!!aaa";
```

```
b = c;
c = b;
```

4. 프로그래밍 언어 활용-SEC_07(배열과 문자열)

5) 배열 형태의 문자열 변수

JAVA의 문자열

- C 언어에서는 문자열을 배열에 넣고 배열의 이름을 이용하든지 포인터 변수를 이용해 처리했지만 JAVA에서는 주소를 컨트롤하는 기능이 없기 때문에 불가능하다. 하지만 JAVA에서는 문자열을 처리할 수 있도록 클래스를 제공한다. 클래스를 제공하므로 당연히 그에 따른 속성과 메소드도 지원하는데 여기서는 문제 풀이에 꼭 필요한 속성과 메소드만 언급하도록 하겠다.

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

```
public class Example {

    public static void main(String[] args) {

        String str = "Information!";
        int n = str.length();
        char[] st = new char [n];
        n--;

        for (int k = n; k >= 0; k--) {
            st[n-k] = str.charAt(k);
        }
        for (char k : st) {
            System.out.printf("%c",  k);
        }
    }
}
```

결과 : !noitamrofnI

프로그래밍 언어 활용-SEC_07(배열과 문자열) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(배열과 문자열)

1. 다음 JAVA프로그램이 실행되었을 때의 결과를 쓰시오.

```
public class Example {

    public static void main(String[] args) {

        int arr[];
        int i = 0;

        arr = new int[10];
        arr[0] = 0;
        arr[1] = 1;

        while(i < 8) {
            arr[i + 2] = arr[i + 1] + arr[i];
            i++;
        }
        System.out.println(arr[9]);
    }
}
```

① 13 ② 21

③ 34 ④ 55

JAVA에서는 배열을 객체로 취급하며, 배열을 이용하여 작업할 때 필요할 만한 내용은 이미 API로 만들어 두었기 때문에 잘 골라서 사용하면 된다. 배열은 JAVA에서 객체로 취급되며, 객체(참조) 변수는 'new'명령을 이용해서 생성해야 하며, 그렇게 메모리 영역 heap에 할당되고 각각의 배열은 데이터 타입의 기본값으로 초기화가 이루어진다.

2. 다음 C언어 프로그램이 실행되었을 때의 결과는?

3. C언어에서 배열 b[5]의 값은?

```
static int b[9] = { 1, 2, 3 };
```

① 0 ② 1

③ 2 ④ 3

배열의 초기값을 배열의 크기보다 적은 수로 초기화 하면 입력된 값만큼 지정한 숫자가 저장되고 나머지 요소에는 데이터 타입의 형태에 따라 기본값으로 초기화가 이루어진다.

4. a[0]의 주소값이 10일 경우 다음 C 언어 프로그램이 실행되었을 때의 결과는? (단, int 형의 크기는 4Byte로 가정한다.)

```
main() {
    int a[] = {  14,22,30,38 };
    printf("%u, ",  &a[2]);
    printf("%u", a);
}
```

① 14, 10 ② 14, M

③ 18, 10 ④ 18, M

a[2]가 가리키는 주소를 %u라는 부호 없는 정수형으로 출력하고 이어서 쉼표(,)와 공백 한 칸을 출력한다. a[0]의 주소값이 문제에서 10이라고 하였고, 정수형(int)의 크기가 4byte이므로 a[2]의 주소는 18과 , 이 출력이

프로그래밍 언어 활용-SEC_07(배열과 문자열) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(배열과 문자열)

5. 다음 JAVA 프로그램이 실행되었을 때, 실행 결과는?

```
public class Example {

    static void rs(char a[]) {
        for (int i = 0; i < a.length; i++)
            if (a[i] =='B')
                a[i] = 'C';
            else if (i == a.length - 1)
                a[i] = a[i - 1];
            else a[i] = a[i + 1];
    }
    static void pca(char a[]) {
        for (int i = 0; i < a.length; i++)
            System.out.print(a[i]);
        System.out.println();
    }

    public static void main(String[] args) {

        char c[] = {'A','B','D','D', 'A','B','C'};
        rs(c);
        pca(c);
    }
}
```

① BCDABCA ② BCDABCC

③ CDDACCC ④ CDDACCA

4. 프로그래밍 언어 활용-SEC_08(포인터)

1) 포인터와 포인터 변수

; 포인터는 변수의 주소값을 저장하며, C 언어에서는 주소를 제어할 수 있는 기능을 제공한다.

- C 언어에서 변수의 주소를 저장할 때 사용하는 변수를 포인터 변수라 한다.
- 포인터 변수를 선언할 때는 자료의 형을 먼저 쓰고 변수명 앞에 간접 연산자 *(asterisk)를 붙인다. (예 int *a)
- 포인터 변수에 주소를 저장하기 위해 변수의 주소를 알아낼 때는 변수 앞에 번지(주소) 연산자 &를 붙인다(예 a = &b;)
- 실행문에서 포인터 변수에 간접 연산자 *를 붙이면 해당 포인터 변수가 가리키는 곳의 값을 말한다. (간접 참조(Indirect Reference, 예 c = *a;)
- 포인터 변수는 필요에 의해 동적으로 할당되는 메모리 영역인 힙 영역에 접근하는 동적 변수이다.

포인터의 개념

앞에서 어떤 수나 문자를 저장하기 위해 변수를 사용했다. 사실 이 변수는 기억장소의 어느 위치에 대한 이름이며 그 위치는 주소로도 표현할 수 있다. 우리는 친구 홍길동의 집에 모이기 위해 "홍길동이네 집으로 와"라고 말하기도 하지만 "홍길동의 집 주소인 서울시 마포구 서교동 00번지로 와"라고 말하기도 한다. C 언어에서는 변수의 주소를 포인터라고 하고, 포인터를 저장할 수 있는 변수를 포인터 변수라고 한다. 변수의 주소인 포인터는 출력할 수도 있고 포인터가 가리키는 곳에 값을 저장하거나 읽어 오는 등 다양한 조작(간접 참조)이 가능하다. 이런 기능 때문에 C 언어는 주소를 제어할 수 있는 기능이 있다고 말한다.

4. 프로그래밍 언어 활용-SEC_08(포인터)

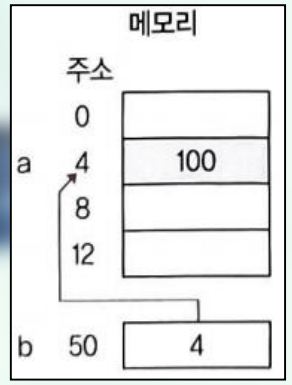
1) 포인터와 포인터 변수

- 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.

4. 프로그래밍 언어 활용-SEC_08(포인터)

1) 포인터와 포인터 변수

; 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 a변수의 주소를 말한다. 즉 &a는 4다.
- 포인터 변수 b는 a변수의 주소를 기억하고 있다.
- 포인터 변수가 가리키는 곳의 값을 말할 때는 *을 붙인다.
- *b는 b에 저장된 주소가 가리키는 곳에 저장된 값을 말하므로 100이다.

메모리 영역

운영체제는 다음과 같이 메모리 영역을 할당하여 프로그램을 관리한다.

- 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

4. 프로그래밍 언어 활용-SEC_08(포인터)

1) 포인터와 포인터 변수

예제1) 다음 C언어로 구현된 프로그램의 출력 결과를 확인하시오.

```
main() {
    int a = 50;
    int* b = NULL;
    b = &a;
    *b = *b + 20;
    printf("%d, %d", a, *b);
}
```

결과 70, 70

예제2) 다음 C언어로 구현된 프로그램의 출력 결과를 확인하시오.

```
main() {
    int a = 3, * b = NULL;
    b = &a;
    printf("%d", ++*b);
}
```

결과 4

4. 프로그래밍 언어 활용-SEC_08(포인터)

2) 포인터와 배열

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

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

예) int a[5] = {0}, *b = NULL;

b = a → 배열의 이름을 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.

- 배열의 요소가 포인터인 포인터형 배열을 선언할 수 있다.

예) char *data[] = {"가나다", "ABC", "포인터"};

4. 프로그래밍 언어 활용-SEC_08(포인터)

2) 포인터와 배열

예제) 다음의 출력 결과를 확인하시오.

```
main() {
    int a[5] = {0};
    int i = 0;
    int *p = NULL;

    for (i = 0; i < 5; i++)
        a[i] = i + 10;
    p = a;          //포인터 변수에 배열명(주소 저장)

    for (i = 0; i < 5; i++)
        printf("%d ", *(p + i));    //포인터 간접 참조로 인한 출력
}
```

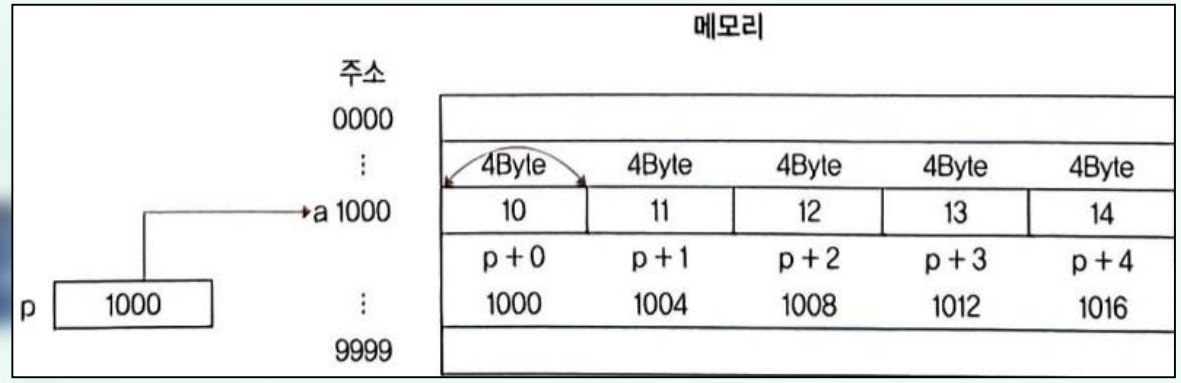
결과 10 11 12 13 14

포인터 표기 방법
a에 저장된 값은 정수형 배열의 시작 주소이다. a의 값을 1증가시킨다는 것은 현재 a가 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. 위의 예제를 통해 배열의 시작 주소에서 1번지씩, 즉 4Byte씩 증가시키는 것을 이해하도록 하자.

4. 프로그래밍 언어 활용-SEC_08(포인터)

2) 포인터와 배열

앞의 예제를 메모리로 출력하면 다음과 같다.



- 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.

포인터의 증가가 헷갈리면 정수형 자료의 크기가 4Byte이기 때문에 포인터를 증가시키면 물리적인 주소는 4Byte가 증가한다고 기억해 두면 된다. 만약 p가 문자 배열의 주소를 가지고 있다면 문자형 자료의 크기는 1Byte이므로 포인터를 증가시킬 때 물리적인 메모리 의 주소도 1Byte 증가한다.

4. 프로그래밍 언어 활용-SEC_08(포인터)

2) 포인터와 배열

문제) 다음 프로그램의 출력 결과를 적으시오.

①

```
main() {
    int a = 5, b, *c = NULL;
    c = &a;
    b = ++(*c);
    printf("%d", b);
}
```

결과 : 6

②

```
main() {
    int a = 10, *b = NULL;
    b = &a;

    for (int i = 0; i < 5; i++)
        *b += i;
    printf("%d", *b);
}
```

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

결과 : 20

4. 프로그래밍 언어 활용-SEC_08(포인터)

2) 포인터와 배열

문제) 다음 프로그램의 출력 결과를 적으시오.

③
```
main() {
    int a = 31, b = 0, *c = NULL, *d = NULL;
    c = &a;
    d = &b;
    *d = --(*c) % 3 ? a + a : a * a;
    printf("%d", *d);
}
```

결과 : 900

④
```
main() {
    int a = 5, b = 7, c = 0, *d = NULL;
    d = &c;
    *d = a & b;
    printf("%d", c);
}
```

결과 : 5

프로그래밍 언어 활용-SEC_08(포인터) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(포인터)

1. 다음 C언어 프로그램이 실행되었을 때의 결과는(단, n의 주소는 1000이라 가정한다.)?

```
main() {
    int n = 4;
    int* pt = NULL;
    pt = &n;
    printf("%d", &n + *pt - *&pt + n);
}
```

① 0 ② 4

③ 8 ④ 12

풀이 과정

1. int 타입의 변수 n에 4를 저장한다.

2. int* 타입의 포인터 변수 pt에 NULL로 초기화 한다.

3. 포인터 변수 pt에 n의 주소값 1000을 저장하여 간접 참조하게 만든다.

4. &n = 1000, *pt = 4, *&pt = 1000, n = 4로 된다.

위와 같이 연산을 하여 10진수로 출력하면 1000 + 4 – 1000 + 4 가 되어 결과는 8이 된다.

3. 다음 C언어 프로그램이 실행되었을 때, 실행 결과는?

```
#include<stdio.h>
#include<stdlib.h>
int main(void) {
    char str1[20] = "KOREA";
    char str2[20] = "LOVE";
    char* p1 = NULL;
    char* p2 = NULL;

    p1 = str1;
    p2 = str2;

    str1[1] = p2[2];
    str2[3] = p1[4];
    strcat(str1, str2);
    printf("%c", *(p1 + 2));
}
```

① E ② V

③ R ④ O

strcat(문자배열 A, 문자배열 B) 함수 : A 배열에 저장된 문자열의 마지막에 이어서 B 배열의 저장된 문자열을 이어 붙이는 문자열 함수

4. 포인터 변수의 용도로 알맞지 않은 것은?

① 연결된 자료 구조를 구성하기 위해 사용한다.

프로그래밍 언어 활용-SEC_08(포인터) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(포인터)

5. 주소값을 저장하며, C 언어에서는 주소를 제어할 수 있는 기능을 제공하는 것은?

① 배열 ② 포인터

③ 상수 ④ 구조체

포인터 변수

- C 언어에서 변수의 주소값, 배열명(주소값) 등을 저장할 때 사용하는 변수를 포인터 변수라고 한다.
- 포인터 변수를 선언할 때 자료의 데이터 타입(자료형)을 먼저 쓰고 변수명 앞에 *(asterisk)를 붙인다.
- 포인터 변수에 주소를 저장하기 위해서 변수의 주소를 알아낼 때는 변수 앞에 주소(번지)연산자 &를 붙인다.
- 실행문에서 포인터 변수에 간접 연산자 *를 붙이면 해당 포인터 변수가 가리키는 곳의 값을 의미한다.(간접 참조(Indirect Reference))
- 포인터 변수는 필요에 의해 동적으로 할당(malloc(), calloc(), realloc()되는 메모리 영역인 힙(heap) 영역에 접근하는 동적 변수이다.

6. C 언어의 포인터 조작 연산에서 변수 pc에 대입되는 것과 같은 결과를 갖는 것은?

7. C 언어에서 malloc() 함수에 대한 설명으로 틀린 것은?

① 원하는 시점에 원하는 만큼 메모리를 동적으로 할당한다.

② 사용자가 입력한 bit만큼 메모리를 할당한다.

③ free 명령어로 할당된 메모리를 해제한다.

④ 메모리 할당이 불가능할 경우 NULL이 반환된다.

배열의 단점

1. 배열은 한 번 크기를 정하면 바꿀 수 없는 정적인 형태의 자료구조이다.
2. 선언된 자료형(데이터 타입)의 값만 저장할 수가 있다.
3. 정적인 배열의 장점이라고 그나마 뽑을 수 있는 것은 사용하지 않는 메모리 해제를 자동으로 해주어 메모리 누수(leak)를 방지할 수 있다.

메모리 동적 할당

; 사용자가 원하는 메모리 용량만큼 직접 지정하여 할당받는 것을 메모리 동적 할당이라고 한다.(heap에 할당)

- 메모리 동적 할당을 하는 함수

① **malloc**() -> stdlib.h 헤더파일에 존재

형식 : 포인터 변수 = malloc(크기);

포인터 변수가 가리키는 메모리 위치에 지정된 크기 만큼의 공간을 할당하되 그 크기의 단위가 바이트(byte)이다.

프로그래밍 언어 활용-SEC_08(포인터) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(포인터)

9. 다음 C언어 코드의 결과 값은?

```
main(void) {
    int x[] = { 100, 200, 300, 400 };
    int y = 0;

    y = *(x + 2) + 50;
    printf("%d", y);
}
```

① 150　　② 250

③ 350　　④ 450

배열 x가 가리키는 곳의 주소에서 2를 증가한다는 것은 현재 x가 가리키고 있는 정수형 자료의 주소(&x[0])에서 정수형 자료의 크기만큼 즉 4바이트를 2번 증가하므로 x[2]의 주소가 되므로 *(x + 2)는 실제 값이므로 300이다. 그리고 300 + 50을 더하므로 결과는 350이 된다.

4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

1) Python의 기본 문법

- 변수의 자료형에 대한 선언이 없다.
- 문장의 끝을 의미하는 세미콜론(;)을 사용할 필요가 없다.
- 변수에 연속하여 값을 저장하는 것이 가능하다.

  예) x, y, z = 10, 20, 30
- if나 for와 같이 코드 블록을 포함하는 명령문을 작성할 때 코드 블록은 콜론(:)과 여백으로 구분한다.
- 여백은 일반적으로 4칸 또는 한 개의 탭만큼 띄워야 하며, 같은 수준의 코드들은 반드시 동일한 여백을 가져야 한다.

단, Python에서 한 줄에 여러 문장을 쓸 때는 세미콜론을 이용하여 문장을 구분하여 입력한다.

예) a = 1; print(a)

코드 블록 구분

명령문에서 코드의 블록을 지정할 때 C와 Java에서는 중괄호({})를 사용하지만 Python에서는 여백을 통해 코드 블록 을 지정한다('v'는 빈 칸을 의미한다.)

예) Python

```
if a > b :
VVVVprint('a is big')
else :
VVVVprint('b is big')
```

C 언어

```
if(a > b){
VVVVprintf('a is big');
}
else {
 VVVVprint('b is big');
}
```

4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

2) Python의 데이터 입·출력 함수

input() 함수

- input() 함수는 Python의 표준 입력 함수로, 키보드로 입력 받아 변수에 저장하는 함수이다.
- 입력되는 값은 문자열로 취급되어 저장된다.
- 형식 1

```
변수 = input(출력문자)
```

  - '출력문자'는 생략이 가능하며, '변수'는 사용자가 임의로 지정할 수 있다.
  - 값을 입력하고 Enter를 누르면, 입력한 값이 '변수'에 저장된다.

- 형식 2

```
변수1, 변수2, … = input(출력문자).split(분리문자)
```

  - 화면에 '출력문자'가 표시되고 입력 받은 값을 '분리문자'로 구분하여 각각 변수1, 변수2 …에 저장한다.

    예) x, y = input().split('-') "KOREA-FIGHTING"을 입력할 경우, '분리문자'를 '-' 기준으로 "KOREA"은 변수 x에 저장되고, "FIGHTING"은 변수 y에 저장된다.

4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

2) Python의 데이터 입·출력 함수

print() 함수

● 형식 1

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

- '출력값'에는 숫자, 문자, 문자열 변수 등 다양한 값이나 식이 올 수 있다.
- 'sep'는 여러 값을 출력할 때 값과 값 사이를 구분하기 위해 출력하는 문자로, 생략할 경우 기본값은 공백 한 칸(' ')이다.
- 'end'는 맨 마지막에 표시할 문자로, 생략할 경우 기본값은 줄 바꿈이다.

예) print(82, 24, sep = '-', end = '.')

82와 24사이에 분리문자 '-'가 출력되고, 마지막에 종료문자 '.'가 출력된다.

결과 82-24.

4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

2) Python의 데이터 입·출력 함수

print() 함수

● 형식 2

```
print(서식 문자열  % (출력값1, 출력값2, …))
```

- C 와 Java에서 사용했던 서식 문자열이 동일하게 적용된다.
- 출력값이 한 개일 경우 출력값에 대한 괄호를 생략할 수 있다.

예) print("%-8.2f" % 200.20)

▶ % : 서식 문자임을 지정

▶ - : 왼쪽부터 출력

▶ 8 : 출력 자릿수를 8자리로 지정

▶ 2 : 소수점 이하를 2자리로 지정

▶ f : 실수로 출력

4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

2) Python의 데이터 입·출력 함수

입력 값의 형변환(Casting)

- input() 함수는 입력되는 값을 무조건 문자열로 저장하므로, 숫자로 사용하기 위해서는 형을 변환해야 한다.

● 변환할 데이터가 1개일 때

```
변수 = int(input( ))        정수로 변환 시
변수 = float(input( ))      실수로 변환 시
```

예) a = int(input()) -> input()으로 입력 받은 값을 정수로 변환하여 변수 a에 저장한다.

● 변환할 데이터가 2개 이상일 때

```
변수1, 변수2, … = map(int, input( ).split( ))      정수로 변환 시
변수1, 변수2, … = map(float, input( ).split( ))    실수로 변환 시
```

예) a, b = map(int, input().split()) -> input().split()으로 입력 받은 2개의 값을 정수로 변환하여 변수 a, b에 저장한다.

map() 함수

map()함수는 input().split()을 통해 입력 받은 2개 이상의 값을 원하는 자료형으로 변환할 때 사용하는 함수이다.

4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

3) 리스트(List)

- C 와 Java 에서는 여러 요소들을 하나의 이름으로 처리할 때 배열을 사용했는데 Python에서는 리스트를 사용한다.
- 리스트는 필요에 따라 개수를 늘이거나 줄일 수 있기 때문에 리스트를 선언할 때 크기를 적지 않는다.
- 배열과 달리 하나의 리스트에 정수, 실수, 문자열 등 다양한 자료형을 섞어서 저장할 수 있다.
- Python에서 리스트의 위치는 0부터 시작한다.
- 형식

```
리스트명 = [ 값1, 값2, … ]
리스트명 = list([ 값1, 값2, … ])
```

리스트 명은 사용자가 임의로 지정하며, 리스트를 의미하는 대괄호 사이에 저장할 값들을 쉼표(,)로 구분하여 입력한다.

예1) 방법1 : a = [10, 'mike', 23.45]

방법2 : a = list([10, 'mike', 23.45])

	a[0]	a[1]	a[2]
리스트 a	10	mike	23.45

예2) a[0] = 1 -> a[0]에 1을 저장한다.

	a[0]	a[1]	a[2]
리스트 a	1	mike	23.45

4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

4) 딕셔너리(Dictionary)

- 딕셔너리는 연관된 값을 묶어서 저장하는 용도로 사용한다.
- 리스트는 저장된 요소에 접근하기 위한 키로 위치에 해당하는 0, 1, 2 등의 숫자를 사용하지만 딕셔너리는 사용자가 원하는 값을 키로 지정해 사용한다.
- 딕셔너리에 접근할 때는 딕셔너리 뒤에 대괄호([])를 사용하며, 대괄호([]) 안에 키를 지정한다.
- 형식

```
딕셔너리명 = { 키1:값1, 키2:값2, … }
딕셔너리명 = dict({ 키1:값1, 키2:값2, … })
```

딕셔너리명은 사용자가 임의로 지정하며, 딕셔너리를 의미하는 중괄호 사이에 저장할 값들을 쉼표(,)로 구분한다.

예1) 방법1 : a = {'이름' : '홍길동', '나이' : 25, '주소' : '서울'}

방법2 : a = dict({'이름' : '홍길동', '나이' : 25, '주소' : '서울'})

a['이름']	a['나이']	a['주소']
'홍길동'	25	'서울'

예2) a['이름'] = '이순신' 딕셔너리 a의 '이름' 위치에 '이순신'을 저장한다.

a['이름']	a['나이']	a['주소']
'이순신'	25	'서울'

4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

5) Range

; Range는 연속된 숫자를 생성하는 것으로, 리스트, 반복문 등에서 많이 사용된다.

● 형식

```
range(최종값)
range(초기값, 최종값)
range(초기값, 최종값, 증가값)
```

1. range(최종값) : 0에서 '최종값' - 1까지 연속된 숫자를 생성한다.
2. range(초기값, 최종값) : '초기값'에서 '최종값' -1까지 연속된 숫자를 생성한다.
3. range(초기값, 최종값, 증가값)
   - '초기값'에서 '최종값' -1까지 '증가값'만큼 증가하면서 숫자를 생성한다.
   - '증가값'이 음수인 경우 '초기값'에서 '최종값' + 1까지 '증가값'만큼 감소하면서 숫자를 생성한다.

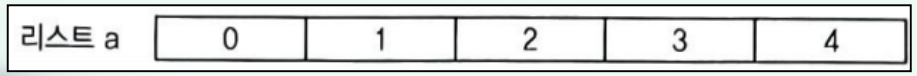
4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

5) Range

예1) a = list(range(5)) -> 0에서 4까지 연속된 숫자를 리스트 a로 저장한다.

```
a = list(range(5))
print(a)
```

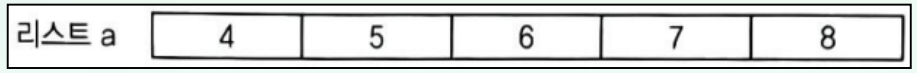
```
[0, 1, 2, 3, 4]
```



예2) a = list(range(4, 9)) -> 4에서 8까지 연속된 숫자를 리스트로 저장한다.

```
a = list(range(4, 9))
print(a)
```

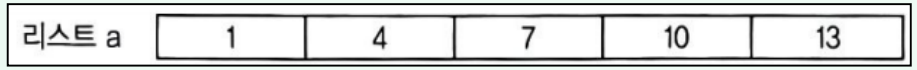
```
[4, 5, 6, 7, 8]
```



예3) a = list(range(1, 15, 3)) -> 1에서 14까지 3씩 증가하는 숫자들을 리스트로 저장한다.

```
a = list(range(1, 15, 3))
print(a)
```

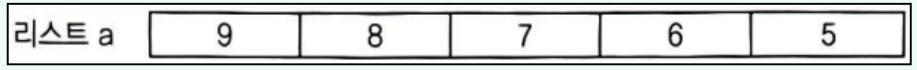
```
[1, 4, 7, 10, 13]
```



예4) a = list(range(9, 4, -1)) -> 9에서 5까지 1씩 감소하는 숫자들을 리스트로 저장한다.

```
a = list(range(9, 4, -1))
print(a)
```

```
[9, 8, 7, 6, 5]
```



4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

6) 슬라이스(Slice)

; 슬라이스는 문자열이나 리스트와 같은 순차형 객체에서 일부를 잘라(slicing) 반환하는 기능이다.

- 형식

  객체명[초기위치:최종위치]

  - '초기위치'에서 '최종위치' - 1까지의 요소들을 가져온다.

  객체명[초기위치:최종위치:증가값]

  - '초기위치'에서 '최종위치' - 1까지 '증가값' 만큼 증가하면서 해당 위치의 요소들을 가져온다.

- 슬라이스는 일부 인수를 생략하여 사용할 수 있다.

객체명[:] 또는 객체명[::]	객체의 모든 요소를 반환한다.
객체명[초기위치:]	객체의 '초기위치'에서 마지막 위치까지의 요소들을 반환한다.
객체명[:최종위치]	객체의 0번째 위치에서 '최종위치'-1까지의 요소들을 반환한다.
객체명[::증가값]	객체의 0번째 위치에서 마지막 위치까지 '증가값'만큼 증가하면서 해당 위치의 요소들을 반환한다.

순차형 객체(Sequential Object) : 문자열이나 리스트와 같이 메모리에 순차적으로 데이터가 저장되는 자료 구조의 객체를 의미한다.

4. 프로그래밍 언어 활용-SEC_09(Python의 기초)

6) 슬라이스(Slice)

예) a = ['a', 'b', 'C', 'd, 'e']일 때

```
a = ['a', 'b', 'C', 'd', 'e']
print(a[1:3])
print(a[0:5:2])
print(a[3:])
print(a[:3])
print(a[::3])
```

```
['b', 'C']
['a', 'C', 'e']
['d', 'e']
['a', 'b', 'C']
['a', 'd']
```

예제) 객체에 저장된 값을 코드와 같이 수행했을 때의 결과를 쓰시오.

번호	객체	코드	결과
1	a = 'KOREA'	print(a[3:7])	EA
2	a = list(range(10))	print(a[:7:2])	[0, 2, 4, 6]
3	a = 'hello, world'	print(a[7:])	world
4	a = list(range(5, 22, 2))	print(a[::3])	[5, 11, 17]
5	a = list(range(8))	print(a[2::2])	[2, 4, 6]
6	a = list(range(8, 3, -1))	print(a[:3])	[8, 7, 6]
7	a = 'Programming'	print(a[-7::2])	rmig
8	a = 'Concurrency Control'	print(a[:-10:-2])	lrnCy

프로그래밍 언어 활용-SEC_09(Python의 기초) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(Python의 기초)

1. 다음 파이썬으로 구현된 프로그램의 실행 결과로 옳은 것은?

```
a = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
print(a[:7:2])
```

① [20, 60] ② [60, 20]

③ [0, 20, 40, 60] ④ [10, 30, 50, 70]

a[:7:2]는 리스트 a의 0번째 위치부터 7-1까지의 위치까지 2씩 위치를 증가시키면서 해당 위치의 요소의 값을 출력하라는 의미이다.

2. 다음은 사용자로부터 입력 받은 문자열에서 처음과 끝의 3글자 를 추출한 후 합쳐서 출력하는 파이썬 코드이다. ㉠에 들어갈 내용은?

```
str = input("7문자 이상 문자열을 입력하시오 : ")
m = ㉠
print(m)
```

② str[:3] + str[-3:-1]

③ str[0:3] + str[-3:]

④ str[0:] + str[:-1]

보기에 코드들은 '객체명[초기위치:최종위치'로 기본 형식에서

3. 다음 Python프로그램이 실행되었을 때, 실행 결과는?

```
a = 100
list_data = ['a','b','c']
dict_data = {'a':90, 'b':95}
print(list_data[0])
print(dict_data['a'])
```

① a
90

② 100
90

③ 100
100

④ a
a

4. 다음 파이썬 코드에서 '53t44'를 입력했을 때 출력 결과는?

```
a, b = map(int, input().split("t"))
print(a, b)
```

① 53 t 44 ② 53t44

③ 53 44 ④ 53, 44

input()메소드로 입력 받은 값을 "t"를 구분자로 하여 분리한 후 정수로 캐스팅하여 a, b에 각각 저장한다. 문제에서 "53t44"를 입력하였기에 "t"를 구분자로 하여 53과 44가 분리된 후 정수로 변환되어 각각 저장된다.

map() : 2개 이상의 값을 원하는 자료형으로 변환할 때 사용하는 함수

프로그래밍 언어 활용-SEC_09(Python의 기초) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(Python의 기초)

5. 다음 중, Python의 기본 문법에 대해 틀린 것은?

① 변수의 자료형에 대한 선언이 없다.

② 문장의 끝을 의미하는 세미콜론(;)을 사용할 필요가 없다.

③ if나 for와 같이 코드 블록을 포함하는 명령문을 작성할 때 코드 블록은 콜론(:)과 여백으로 구분한다.

④ 여백은 일반적으로 1칸 만큼 띄워야 한다.

Python의 기본 문법

- 변수의 자료형에 대한 선언이 없다.
- 문장의 끝을 의미하는 세미콜론(;)을 사용할 필요가 없다. 한 문장에 여러 명령문을 작성할 때는 ;으로 구분한다.
- if나 for와 같이 코드 블록을 포함하는 명령문을 작성할 때 코드 블록은 콜론(:)과 여백으로 구분한다.
- 변수에 연속하여 값을 저장하는 것이 가능하다.
- 여백은 일반적으로 4칸 또는 한 개의 탭만큼 띄워야 하며, 같은 수준의 코드들은 반드시 동일한 여백을 가져야 한다.

6. input().split()을 통해 입력 받은 2개 이상의 값을 원하는 자료형으로 변환할 때 사용하는 함수는 무엇인가?

7. 다음 중, 딕셔너리(Dictionary)에 대한 설명으로 옳지 않은 것은?

① 딕셔너리는 연관된 값을 묶어서 저장하는 용도로 사용한다.

② 리스트는 저장된 요소에 접근하기 위한 키로 위치에 해당하는 0, 1, 2 등의 숫자를 사용하지만 딕셔너리는 사용자가 원하는 값을 키로 지정해 사용한다.

③ 딕셔너리에 접근할 때는 딕셔너리 뒤에 대괄호([])를 사용하며, 대괄호([]) 안에 키를 지정한다.

④ 딕셔너리명은 사용자가 임의로 지정하며, 딕셔너리를 의미하는 대괄호 사이에 저장할 값들을 쉼표(,)로 구분한다.

딕셔너리명은 사용자가 임의로 지정하며, 딕셔너리를 의미하는 **중괄호** 사이에 저장할 값들을 쉼표(,)로 구분한다.

8. 다음 중, print() 함수에 대한 설명으로 틀린 것은?

① '출력값'에는 숫자, 문자, 문자열 변수 등 다양한 값이나 식이 올 수 있다.

② 'sep'는 여러 값을 출력할 때 값과 값 사이를 구분하기 위해 출력 하는 문자로, 생략할 경우 기본값은 공백 한 칸(' ')이다.

③ 'end'는 맨 마지막에 표시할 문자로, 생략할 경우 기본값은 줄 바꿈이다.

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

1) if문

● 형식1 : 조건이 참일 때만 실행한다.

```
if 조건:
    실행할 문장
```

- 예약어 if와 참 또는 거짓이 결과로 나올 수 있는 조건을 입력한 후 끝에 콜론(:)을 붙여준다.
- 조건이 참일 경우 실행할 문장을 적는다.

예제1) a가 10보다 크면 a에서 10을 빼기

```
a = 15
if a > 10:
    a -= 10
print(a)
```

결과 : 5

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

1) if문

● 형식2 : 조건이 참일 때와 거짓일 때 실행할 문장이 다르다.

```
if 조건:
    실행할 문장1
else:
    실행할 문장2
```

예제2) a가 b보다 크면 a - b, 아니면 b - a를 수행하기

```
a, b = 10, 20
if a > b:
    cha = a - b
    print(cha)
else:
    cha = b - a
    print(cha)
```

결과 : 10

```
main(void) {
    int a = 10, b = 20, cha = 0;

    if(a > b) {
        cha = a - b;
        printf("%d", cha);
    }
    else {
        cha = b - a;
        printf("%d", cha);
    }
}
```

C 언어 코드, 결과 : 10

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

1) if문

● 형식3 : 조건이 여러 개이고, 조건마다 실행할 문장이 다르다.

```
if 조건1:
    실행할 문장1
elif 조건2:
    실행할 문장2
elif 조건3:
    실행할 문장3
    :
else:
    실행할 문장4
```

예제3) 점수에 따라 등급 표시하기

```python
jum = 85
if jum >= 90:
    print('학점은 A입니다.')
elif jum >= 80:
    print('학점은 B입니다.')
elif jum >= 70:
    print('학점은 C입니다.')
else:
    print('학점은 F입니다.')
```

결과 : 학점은 B입니다.

```c
main(void) {
    int jum = 85;

    if(jum >= 90)
        printf("학점은 A입니다.");
    else if (jum >= 80)
        printf("학점은 B입니다.");
    else if (jum >= 70)
        printf("학점은 C입니다.");
    else
        printf("학점은 F입니다.");
}
```

C 언어 코드

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

1) if문

● 형식4 : if문 안에 if문이 포함된다.(중첩(Nested) if문)

```
if 조건1:
    if 조건2:
        실행할 문장1
    else:
        실행할 문장2
else:
    실행할 문장3
```

예제4) 홀수, 짝수 판별하기

```python
a, b = 21, 10
if a % 2 == 0:
    if b % 2 == 0:
        print('모두 짝수')
    else:
        print('a : 짝수, b : 홀수')
else:
    if b % 2 == 0:
        print('a : 홀수, b : 짝수')
    else:
        print('모두 홀수')
```

결과 : a : 홀수, b : 짝수

```c
main(void) {
    int a = 21, b = 10;
    if (a % 2 == 0)
        if (b % 2 == 0)
            printf("모두 짝수");
        else
            printf("a : 짝수, b : 홀수");
    else
        if (b % 2 == 0)
            printf("a : 홀수, b : 짝수");
        else
            printf("모두 홀수");
}
```

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

2) for문

● 형식1 : range를 이용하는 방식이다.

```
for 변수 in range(최종값):
    실행할 문장
```

- 0에서 '최종값' - 1까지 연속된 숫자를 순서대로 변수에 저장하며 '실행할 문장'을 반복 수행한다.
- 반복 수행할 문장을 적는다.

예제1)

```
sum = 0
for i in range(10):
    sum += i
print(sum)
```

① 누적할 변수 sum은 반드시 초기화를 해야 한다.

② 0에서 9까지 순서대로 저장하며 실행할 문장을 반복 수행한다.

③ i의 값을 sum에 누적한다. sum에는 0부터 9까지의 합 45가 저장된다.

예제2)

```
sum = 0
for i in range(11, 20):
    sum += i
print(sum)
```

누적할 변수 sum은 반드시 초기화를 해야 한다.

② i에 11에서 19까지 순서대로 저장하며 실행할 문장을 반복 수행한다.

③ i의 값을 sum에 누적. sum에는 11부터 19까지의 합 135가 저장된다.

예제3)

```
sum = 0
for i in range(-10, 20, 2):
    sum += i
print(sum)
```

적할 변수 sum은 반드시 초기화를 해야 한다.

② i에 -10에서 19까지 2씩 증가하는 숫자를 순서대로 저장하며 실행할 문장을 반복 수행한다.

③ i의 값을 sum에 누적. sum에는 -10, -8... 18의 합 60이 저장된다

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

2) for문

● 형식2 : 리스트(List)를 이용하는 방식이다.

```
for 변수 in 리스트
    실행할 문장
```

- 리스트의 0번째 요소에서 마지막 요소까지 순서대로 변수에 저장하며 실행할 문장을 반복 수행한다.

예제) 다음은 리스트 a에 저장된 요소들의 합과 평균을 구하는 프로그램을 Python으로 구현한 것이다.

```
a = [35, 55, 65, 84, 45]
hap = 0
for i in a:
    hap += i
avg = hap / len(a)
print(hap, avg)
```

- len(리스트) : 리스트의 요소 수를 구한다. len(a)는 5다.

결과 : 284, 56.8

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

3) while문

● 형식 : 리스트(List)를 이용하는 방식이다.

```
while 조건:

    실행할 문장
```

- while은 예약어로, 그대로 입력한다.
- 참이나 거짓을 결과로 갖는 수식을 조건에 입력한다. 참(1 또는 True)을 입력하면 무한 반복이 된다.
- 조건이 참인 동안 반복 수행할 문장을 적는다.
- Python에서도 반복문의 실행을 제어하는 break, continue C, Java와 동일하게 사용할 수 있다.

예제) 다음은 1 ~ 5까지의 합을 구하는 프로그램을 Python으로 구현한 것이다.

```
i, hap = 0, 0
while i < 5:
    i += 1
    hap += i
print(hap)
```

결과 : 15

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

4) 클래스

- 정의 형식

```
# class는 예약어이며 클래스명은 사용자가 임의로 지정한다.
class Member:
    # 멤버 변수
    age = 0
    height = 178
    weight = 87.2

    # 생성자(객체화 시 바로 수행하는 것)
    # self 란 자기 자신 즉, 객체(인스턴스)로 메모리에 있는 자신의 주소를 의미한다.
    def __init__(self):
        print("생성자 호출")

    # 멤버 메서드
    def print(self):
        print(self.age, self.height, self.weight)
    # return 문을 이용하여 멤버 메서드로 멤버 변수를 리턴하게 할 수도 있다.
    def getHeight(self):
        return self.height
```

- 객체의 선언 형식

변수명 = 클래스명() → 변수명은 사용자가 임의로 지정하고, 사전에 정의한 클래스명과 괄호()를 적는다.

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

4) 클래스

예제1) 다음은 두 수를 교환하는 프로그램을 Python으로 구현한 것이다.

```
class Cls:        # Cls 클래스 정의부의 시작점이다.
    # Cls 클래스의 멤버 변수(속성) x와 y를 선언하고, 각각 10과 20으로 초기화한다.
    x, y = 10, 20
    def chg(self):
        temp = self.x
        self.x = self.y
        self.y = temp    # Cls 클래스 정의부 끝점`

a = Cls()
print(a.x, a.y)
a.chg()
print(a.x, a.y)
```

결과 10 20

20 10

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

4) 클래스

예제2) 다음은 0부터 10까지 더하는 프로그램을 Python으로 구현한 것이다.

```
class Cls:        # Cls 클래스 정의부의 시작점이다.
    # rep 메소드의 시작점이다.
    def rep(self, r):
        hap = 0
        for i in range(r + 1):
            hap += i
        return hap

a = Cls()
b = a.rep(10)
print(b)
```

결과 55

4. 프로그래밍 언어 활용-SEC_10(Python의 활용)

5) 클래스 없는 메소드의 사용

; C언어의 사용자 정의 함수와 같이 클래스 없이 메소드만 단독으로 사용할 수 있다.

예제) 다음 프로그램의 실행 결과를 확인하시오.

```
def calc(x, y): # 메소드 calc 의 시작점이다. 매개변수로 x, y를 받는다.
    x *= 3
    # Python 에서는 나눗셈을 할 때 자동으로 자료형이 float 로 변환되기
    # 때문에 y = 3의 결과로 4가 아닌 4.0이 출력되는 것에 주의하자.
    y /= 3
    print(x, y)
    return x

a, b = 3, 12
a = calc(a, b)
print(a, b)
```

결과 9 4.0

9 12

프로그래밍 언어 활용-SEC_10(Python의 활용) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(Python의 활용)

1. 다음 Python프로그램의 실행 결과가 [실행결과]와 같을 때, 빈 칸에 적합한 것은?

```
x = 20
if x == 10:
    print('10')
    x == 20:
    print('20')
else:
    print('other')
```

① other ② elif

③ else if ④ else

Python에서 조건문에서 여러 개의 조건 추가하여 사용하는 예약어는 if ~ elif ~ else문을 사용한다. 조건마다 실행할 문장이 다를 때 사용한다.

2. 다음 파이썬(Python) 프로그램이 실행되었을 때의 결과는?

```
class FourCal:
    def setdata(self, fir, sec):
        self.fir = fir
        self.sec = sec
    def add(self):
        result = self.fir + self.sec
```

3. 다음은 파이썬으로 만들어진 반복문 코드이다. 이 코드의 결과는?

```
while True:
    print('A')
    print('B')
    print('C')
    continue
    print('D')
```

① A, B, C 출력이 반복된다.

② A, B, C

③ A, B, C, D 출력이 반복된다.

④ A, B, C, D 까지만 출력된다.

무한 루프를 돌릴 때는 반드시 if 조건문을 이용하여 탈출 구문을 작성해야 한다. while True: 는 조건이 항상 참이므로 블록 내의 코드들을 무한 반복시키며, continue라는 예약어는 continue 이후에 코드는 실행하지 않고 반복문의 처음으로 돌아가라는 예약어이다. 따라서, 화면에서는 D를 출력하지 않고 계속 D를 제외한 A 개행 B 개행 C 개행을 출력한다.

4. 다음 Python 프로그램이 실행되었을 때, 실행 결과는?

```
a = ["대", "한", "민", "국"]
```

프로그래밍 언어 활용-SEC_10(Python의 활용) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(Python의 활용)

5. 다음 중, 자바에서 this와 동일하며 자기 자신의 주소를 가지고 있는 Python의 클래스의 예약어는 무엇인가?

① self ② elif

③ sep ④ range

Python에서 self는 자바에서 this와 동일하며, 자기 자신(인스턴스)의 메모리 주소의 시작점을 가지고 있는 인자이다.

self는 객체(인스턴스)가 실행부에서 생성되어야 비로소 self는 활성화가 이루어진다. 아울러, 클래스 내부에서만 사용 가능하며 클래스 외부 즉 실행부에서는 사용할 수가 없다.

self는 매개변수와 멤버변수를 구분할 때도 사용된다.

6. 다음 중, 파이썬(Python)의 클래스의 요소가 아닌 것은?

① 멤버 변수 ② 멤버 메소드

③ 리턴문 ④ 생성자

클래스의 요소

① 생성자 : 클래스를 정의할 때 생성자는 필수적인 요소이다. 하지만, 생략을 하게 되면 인터프리터가 기본적으로 기본 생성자를 추가해주기 때문에 인스턴스를 생성하는데 문제가 없는 것이다.

7. 다음 중, Python의 생성자(Constructor)의 역할로 틀린 것은?

① 클래스를 대상으로 객체를 생성하는 역할을 하는 것.

② 클래스의 이름과 동일함

③ 클래스에 포함되어 있으며, 객체를 생성할 때 자동으로 딱 한번만 호출이 됨.

④ 클래스는 생성자를 통해서 객체로 생성이 됨.

Python에서 생성자는 def __init(self, 초기화 할 매개변수(옵션)__의 형태로 만들어지며 클래스명으로 생성자를 만드는 것은 JAVA언어이다.

일반적으로 함수는 사용자가 함수 이름()와 같은 형태로 호출해야 코드가 수행된다. 이와 달리 클래스 내에서 특별한 이름(__init()__)을 갖기만 하면 객체가 생성될 때 자동으로 호출되는 함수가 바로 생성자라고 한다.

생성자는 객체가 생성될 때 자동으로 호출되기 때문에 객체의 멤버 변수를 초기화하는 용도로 유용하게 사용할 수 있다.

클래스 내부에 포함되어 있으며, 객체를 생성할 때 자동으로 딱 한번만 호출이 된다. 모든 클래스는 반드시 한 개 이상의 생성자를 가지고 있어야 한다. 기본 생성자라 함은 매개변수가 없는 생성자를 의미한다. 매개변수가 있는 생성자가 존재하면 기본 생성자를 추가하더라도

프로그래밍 언어 활용-SEC_10(Python의 활용) 기출 및 출제 예상 문제

기출 및 출제 예상 문제(Python의 활용)

9. 다음 파이썬(Python) 프로그램이 실행되었을 때의 결과는?

```
hap = 0
for x in range(0, 20, 1):
    hap = hap + x
print("누계합 :", hap)
```

① 누계합 : 20　　② 누계합 : 185

③ 누계합 : 0　　④ 누계합 : 190

range(초기값, 최종값, 증감값) 즉 매개변수가 3개짜리 함수를 이용하고 있다. 누적을 시키는 hap은 반드시 초기화가 이루어져야 한다.

10. 다음 중, 사용자로부터 2개의 정수 값을 입력 받고 첫 번째 입력 받은 값부터 두 번째 입력 받은 값까지의 범위에서 3의 배수 를 제외하고 출력하는 프로그램이다. 출력 결과는?

```
n1 = int(input("첫 번째 수 : "))
n2 = int(input("두 번째 수 : "))

for i in range(n1, n2+1):
    if (i % 3) == 0:
```

```
첫 번째 수 : 1
두 번째 수 : 10
```

감사합니다.