<반복문{for}>

1. For문

: 특정한 문장을 일정한 횟수만큼 반복시킬 때 사용

1. 형식과 실행 순서

For(초기화; 조건식; 증감 식)

{

명령문 1

명령문 2

}

-초기값: 변수의 초기값 지정

-조건식: 변수값이 조건식에 맞으면 반복 계속

-실행순서: 초기화->조건식->명령문->증감식->조건식->명령문->증감식

-종료: 조건식이 거짓이 될 때 종료

1. 다중 for문

: for문을 2개 이상 사용한 것

for(초기화(1); 조건식(2); 증감식(3))

{

for(초기화(4); 조건식(5); 증감식(6))

{

명령문 1

}

}

순서: 1 2 3 4 5 6 (판단 참) 4 5 6 (반복)

1. 실행문이 없는 for문

for(j=2; I % j; j++)

;

Null statement(널 문장)

* 아무 일도 하지 않지만 종료 조건이 I % j이므로 나머지가 0일 경우 종료
* 소수인 경우는 i값과 j값이 동일함

<반복문(while)>

1. while문

: 횟수를 정확하게 알지는 못하지만 반복의 조건을 알고 있을 때 사용

\*횟수를 정확히 알 때는 for문 사용

(1) 기본 형식

While(조건식)

{

명령문 1

명령문 2

…

증감식

}

* while문은 조건식을 검사하여 참이면 명령을 수행함.

1. While문과 for문의 초기화 위치 비교

: 일반적으로 for문은 괄호 안에서 초기화, while문은 외부에서 초기화함

1. 무한반복

: 종료되지 않는 반복문

-for문

for( ; ; ){

문장들

}

-while문

While(1){

문장들

}

* 무한반복에서 빠져나오려면, break문을 사용하거나 shift+F5키로 디버깅을 강제로 종료함.

<반복문(do~while)>

1. do~while문

: 일단 do 안에 있는 명령문을 실행, 나중에 while문의 조건식 비교

-적어도 한번은 명령문을 처리함.

-마지막에 세미콜론(;)을 반드시 추가해야함

(1) 형식

Do

{

명령문 1

}while(조건식);

<기타 제어문(goto,break,continue)>

1. goto문

: 실행 순서를 사용자가 지정한 라벨로 이동시킴

1. 형식

goto 라벨(Label); -> 선언할 때는 라벨 뒤 콜론(:)

-라벨(Label)은 프로그램의 다른 명령문 앞에 선언함.

1. Break문

: 반복문을 빠져나가거나 switch~case문에서 정상적인 흐름을 변경시킴

1. Continue문

: 반복문을 빠져 나오지 않고, 해당 반복문의 처음으로 흐름을 옮김.

-while 문: 조건식으로 이동

-do~while문: 조건식으로 이동

-for문: 증감식으로 이동

<배열>

1. 배열의 정의
2. 배열: 동일한 자료형의 데이터를 메모리에 일련의 구조화된 집합으로 구성한 것.
3. 배열 선언

-int형 변수 1개의 메모리 크기가 4byte

-배열의 각 요소는 배열 명에 원소 번호를 붙여서 명명.

-배열의 첫번째 원소는 0번부터 시작.

1. 배열의 초기화
2. 배열을 먼저 선언하고 배열의 각 원소 값을 할당하는 방법

Int a[3]; // 배열 선언

a[0]=10; // 배열의 초기화

a[1]=15;

a[2]=20;

1. 배열을 선언하면서 초기화하는 방법

Int a[3]={10,15,20}; // 배열을 선언함과 동시에 초기화

Int a[3]={10,15}; ->a[2]=0

-초기값이 할당되지 않는 배열의 원소에는 0이 저장

1. 선언하면서 동시에 초기화하면 원소의 개수 표시 안 해도 됨.
2. 배열 선언과 초기값 할당을 따로 하면 원소의 개수 반드시 기록
3. 1차원 배열

: 대괄호 1개로 선언하는 배열

-반복문과 함께 사용하면 유용. 반복문의 증감 변수를 배열의 첨자와 동일하게 여기면 됨

-반복문의 초기값을 설정할 때, 배열의 원소는 0부터 시작

1. 배열값 교환

: 대각선으로 같으면 됨.

Int x=1,y=1;

Int temp;

temp=x

x=y

y=temp

1. 버블 배열

-정렬: 데이터를 오름차순이나 내림차순으로 나타내는 것

-버블 정렬: 이웃 값과 비교하고 결과에 따라 값을 교환

1. 문자 배열(문자열)

: char 형 문자를 여러 개 모아놓은 문자 집합

-문자 ‘ ‘

-문자열 “ “

-‘\0’(NULL): 문자열 끝에 항상 문자코드 ‘\0’을 덧붙여야함

Char str[5]={‘A’,’B’,’C’,’D’,’\0’};

* 배열의 크기를 실제 문자의 개수 + 1로 선언해야 하는 이유

1. 반복문에서 문자열의 종료 조건

For(i=0;str[i];i++)

* “\0” 거짓

While(str[i])

* ‘\0’ 거짓 그 외 문자 참

1. 문자와 문자열 차이

잭, 디자인이(가) 표시된 사진

자동 생성된 설명

1. 배열 주소
2. 메모리가 할당된 변수

-변수명으로 접근

-변수 앞 &를 붙임

1. 배열의 주소 표시

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

1. 2차원 배열

: 행과 열로 표현된 행렬

Int a[2][4]

행,열

1. 초기화 방법

텍스트, 폰트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

-2차원 배열을 선언할 때, 행의 개수는 생략(O), 열의 개수는 생략(X)

<함수>

1. 함수의 일반적인 개념

: 입력=>함수=>출력

어떤 입력에 대해서 특별한 기능을 수행한 후 출력을 하는 것

1. 함수 사용의 장점

-함수: 특정한 기능을 수행하고 그 수행 결과로 값을 반환하는 부 프로그램

-실행 흐름

: 프로그램 실행 중에 함수명과 동일한 부분을 만나면,

해당 함수로 이동->함수 실행->main()함수로 다시 돌아옴.

1. 함수의 종류

-시스템 라이브러리 함수: 사용자가 바로 이용할 수 있도록, 자주 사용하는 함수를 미리 만들어 놓은 것

-사용자 정의 함수: 사용자가 자신에 필요한 함수를 직접 만드는 것

1. 함수의 정의 형태

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

1. 함수의 정의
2. MuitTwo() 함수: 정수(a,b) 2개를 입력받아 곱함
3. SumTwo() 함수: 정수(a,b) 2개를 입력받아 더함
4. Big() 함수: 정수(a,b) 2개를 입력 받아 두 정수 중에서 큰 수 반환
5. Integral() 함수: 시작과 종료 값을 입력하면 그 사이의 값을 모두 더하는 함수

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

1. Void upper\_case: 소문자를 대문자로 바꿔주는 함수

소문자-32=대문자

1. Max: 입력 받은 수 중 가장 큰 값 반환
2. 함수 선언

: 컴파일러에 함수명과 반환값 그리고 함수의 매개변수를 알려줌

-함수 원형이라고도 함.

-함수를 정의하기 전에 선언함.

(1) 선언 형태

반환되는 자료형 함수명(매개변수들);

* 구체적인 인자명 대신, 자료형과 개수만 나타내도 됨.

1. 함수의 호출

: 함수가 정의된 형태에 맞추어 입력과 출력을 고려해서 함수를 사용하는 것

* 호출보다 정의가 먼저 있으면 원형 선언 안해도 됨

1. 함수(인자 값, 재귀 함수)
2. SumArray()함수

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

-배열: 배열명은 가인자이기 때문에 식별자 작성 규칙에 따라 임의로 작성

-원소 개수: 일반적으로 배열명에는 빈 대괄호 [ ] 사용,

실제 배열의 원소 개수는 두 번째 인자로 표시. 변수 size가 배열의 원소 개수

-반환 값: 함수에서 계산된 결과값을 반환

1. 함수의 호출

Sum=SumArray(a,5); -> 배열 부분에 배열명만 기록!!

1. 재귀함수
2. 재귀호출: 함수에서 그 함수를 다시 호출하는 것
3. 재귀함수: 재귀호출을 구현한 함수

n\*factorial(n-1)

return 1

1. 표준 함수
2. Abs(): 정수 절댓값 반환하는 함수

-<stdlib.h>,<math.h>

1. Fabs(): 실수 절댓값 반환하는 함수

-<math.h>

1. Labs: long 타입의 정수 절댓값 반환하는 함수

-<stdlib.h>,<math.h>

1. Pow(x, y): 제곱 -> x^y

-<math.h>

1. Sqrt: 제곱근

-<math.h>

1. Rand: 무작위 수

Srand((unsigned)time(NULL))

Int x=rand()

-<stdlib.h>,<time.h>

1. Int rand(void);

-의사 난수 발생

-0~32767

- srand((unsigned)time(NULL))// <time.h>

-<stdlib.h>

2) void srand(unsigned int seed);

-난수의 초기값 지정

-<stdlib.h>

-난수의 크기 조정

RAND\_MAX

Rand()%((max+1)-min)+min