

# 프로그래밍언어 추가 강의자료

## #1

교과목명 : 프로그래밍언어

담당교수 : 이 수 형

E-mail : [soohyong@uu.ac.kr](mailto:soohyong@uu.ac.kr)

교재명 : C언어 for Beginner

# 예제 프로그램 : 소수 판단

## ■ 임의의 정수를 입력받아 소수인지 판단하는 프로그램의 작성

- 소수 : 1보다 큰 자연수로 1과 자신으로만 나누어지는 수
- 방법
  - 1부터 자기 자신까지 1씩 증가하면서 나누어 떨어지는 횟수를 구하여 두 번이면 소수라고 출력
  - 1과 자기 자신수를 뺀 나머지 숫자를 비교하여 소수 인지 확인하는 방법. 즉 2부터 1씩 증가하면서  $n-1$  ( $n$  : 입력받은 수)번째까지 나누어 보고 이때 한 번도 나누어 떨어지지 않으면 소수로 출력
- 순서
  - 임의의 자연수 입력 : scanf함수 사용
  - 소수 판단 알고리즘 수행

# 예제 프로그램 : 소수 판단

## ■ 방법 1

- 1부터 자기 자신까지 1씩 증가하면서 나누어 떨어지는 횟수를 구하여 두 번이면 소수라고 출력

```
#include <stdio.h>

int main(void)
{
    int n, i, cnt = 0;

    printf("정수 입력 : ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++) {
        if (n % i == 0)
            cnt++;
    }
    if (cnt == 2)
        printf("%d는 소수 입니다.", n);

    return 0;
}
```

# 예제 프로그램 : 소수 판단

## ■ 방법 2

- 2부터 자기 자신 - 1까지 반복하면서 나누어 떨어지는 횟수가 0이면 소수라고 출력

```
#include <stdio.h>

int main(void)
{
    int n, i, cnt = 0;

    printf("정수 입력 : ");
    scanf("%d", &n);

    for (i = 2; i < n; i++) {
        if (n % i == 0)
            cnt++;
    }
    if (cnt == 0)
        printf("%d는 소수 입니다.", n);

    return 0;
}
```

# 예제 프로그램 : 소수 판단

## ■ 속도 개선 #1

- 2부터 자기  $N - 1$ 까지 반복하면서 나누어 떨어지기 시작하면 for 루프 탈출
- 판단 : for문을 끝까지 수행하면 (i의 값이 마지막 다음 값) → 소수 아님

```
#include <stdio.h>

int main(void)
{
    int n, i;

    printf("정수 입력 : ");
    scanf("%d", &n);

    for (i = 2; i <= n; i++) {
        -----
    }
    if (_____)
        printf("%d는 소수 입니다.", n);

    return 0;
}
```

# 예제 프로그램 : 소수 판단

## ■ 속도 개선 #2

- 2부터 자기  $N - 1$ 까지 반복하면서 나누어 떨어지기 시작하면 for 루프 탈출
- $N-1$  이 아니라  $\sqrt{n}$ 까지 반복 (변수  $sn : \sqrt{n}$  보다 큰 자연수 중에서 최소값) :  $\text{sqrt}(n)$

```
#include <stdio.h>

int main(void)
{
    int n, i, sn;

    printf("정수 입력 : ");
    scanf("%d", &n);

    -----
    for (i = 2; i <= sn; i++) {
        -----
    }
    if (-----)
        printf("%d는 소수 입니다.", n);

    return 0;
}
```

# 예제 프로그램 : 소인수 분해

## ■ 임의의 자연수를 입력받아 소인수 분해하는 프로그램의 작성

■ 소인수 분해 : 소수들의 곱으로 표현

### ■ 알고리즘

- 입력받은 수  $\rightarrow n$
- 가장 작은 소수인 2 (변수  $k$ )로 나누어 떨어지면 2 출력,  
 $n = n / 2$
- 다시 2로 나누어지면 같은 일을 반복
- 나누어지지 않으면 다음 숫자로 증가 ( $2 \rightarrow 3$ )
- 3으로 나누어 떨어지면 3 출력,  $n = n / 3$
- ...
- 동일한 과정을 반복하여  $n$ 이 1이면 중단

n	k	출력
60	2	
$60 / k = 30$	2	2 *
$30 / k = 15$	2	2 * 2 *
15	3	2 * 2 *
$15 / k = 5$	3	2 * 2 * 3 *
5	3	2 * 2 * 3 *
5	4	2 * 2 * 3 *
5	5	2 * 2 * 3 *
$5 / 5 = 1$	5	2 * 2 * 3 * 5

$n$ 의 값이 1보다 큰 동안 반복 : while문 사용

# 예제 프로그램 : 소인수 분해

## ■ 프로그램 작성

```
#include <stdio.h>

int main(void)
{
    int n, k = 2;

    printf("정수 입력 : ");
    scanf("%d", &n);

    return 0;
}
```

소인수 분해 알고리즘 작성



# 예제 프로그램 : 소인수 분해

## ■ 소인수 분해 알고리즘

- 소수 후보인  $k$ 를 2로 초기화
- $n$ 이 1보다 클 동안 반복
  - 가장 작은 소수인 2 (변수  $k$ )로 나누어 떨어지면 2 출력,  $n = n / 2$
  - 다시 2로 나누어지면 같은 일을 반복
  - 나누어지지 않으면 다음 숫자로 증가 ( $2 \rightarrow 3$ )
  - 3으로 나누어 떨어지면 3 출력,  $n = n / 3$
  - ...
  - 동일한 과정을 반복하여  $n$ 이 1이면 중단

```
while (n > 1) {  
    ① 만일  $n$ 이  $k$ 로 나누어지면 {  
        ②  $n$ 은  $k$ 로 나눈 값을 저장  
        ③  $k$ 값 출력  
    }  
    ① 나누어지지 않으면 {  
        ④  $k$ 값 증가  
    }  
}
```

# 예제 프로그램 : 소인수 분해

## ■ 소인수 분해 알고리즘

### ■ ① 만일 $n$ 이 $k$ 로 나누어지면

- if ( $n \% k == 0$ )

### ■ ② $n$ 은 $k$ 로 나눈 값을 저장

- $n = n / k$

### ■ ③ $k$ 값 출력

- printf 함수 사용
- 만일  $n$ 이 1보다 크면 더 출력할 수가 있으므로  
“\* ” 출력

### ■ ④ $k$ 값 증가

- $k++$
- 원래 다음 소수로 가야하지만, 소수가 아닌 경우에도  $k$ 의 나누어지는 수는 이미 나누어져 있으므로  $k$ 로 나누어지는 경우는 없음 (예: 4의 경우, 이미 2로 두번 나누어졌음)

```
while (n > 1) {  
    ① 만일  $n$ 이  $k$ 로 나누어지면 {  
        ②  $n$ 은  $k$ 로 나눈 값을 저장  
        ③  $k$ 값 출력  
    }  
    ① 나누어지지 않으면 {  
        ④  $k$ 값 증가  
    }  
}
```

# 예제 프로그램 : 최대공약수 구하기

## ■ 임의의 두 자연수를 입력받아 최대공약수(GCD : Greatest Common Divider)를 구하는 프로그램 작성

### ■ 알고리즘 #1

- 앞의 소인수분해 알고리즘 사용 (두 자연수 :  $a, b \rightarrow \text{gcd}$ )
- $\text{gcd} = 1$ ,  $k$  가 2부터 반복
- 두 숫자 모두 나누어지면 최대공약수  $\text{gcd}$ 에 곱하고, 두 숫자를  $k$ 로 나누어서 저장
- 나누어지지 않으면,  $k$  증가
- $k$ 가 두 숫자보다 작은 경우에는 반복

a	b	k	gcd
36	24	2	1
18	12	2	1 * 2
9	6	2	2 * 2
9	6	3	2 * 2
3	2	3	4 * 2
	중단		

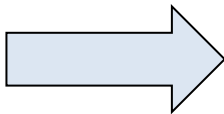
두 숫자 중에서 작은 값까지 반복

# 예제 프로그램 : 최대공약수 구하기

## ■ 임의의 두 자연수를 입력받아 최대공약수(GCD : Greatest Common Divider)를 구하는 프로그램 작성

### ■ 알고리즘 #2 : 유클리드 호제법 (Euclidean Algorithm)

- 두 자연수 :  $a > b \rightarrow \text{gcd}$ 
  - 입력받은 자연수가  $b$ 가 더 크면  $a, b$ 의 값을 교환
- $r = a \% b$ 
  - $r$ 이 0이면 :  $r$ 이 최대공약수
  - $r$ 이 0이 아니면 :  $a, b$ 의 최대공약수는  $b$ 와  $r$ 의 최대공약수가 됨  
→  $a = b, b = r$  대입 후 반복
- GCD( $a, b$ )의 법칙
  - $\text{GCD}(a, b) = \text{GCD}(a - b, b)$
  - $\text{GCD}(a, b) = \text{GCD}(b, a)$
  - $\text{GCD}(a, 0) = a$



1.  $\text{GCD}(36, 24) = \text{GCD}(36 - 24, 24)$
2.  $\text{GCD}(12, 24) = \text{GCD}(24, 12)$
3.  $\text{GCD}(24, 12) = \text{GCD}(24 - 12, 12)$
4.  $\text{GCD}(12, 12) = \text{GCD}(12 - 12, 12)$
5.  $\text{GCD}(12, 0) = 12$

a	b	r
36	24	$36 \% 24 == 12$
24	12	$24 \% 12 == 0$

$r$ 이 0보다 클 때까지 반복