Java를 알고 C배우기

컴퓨터프로그래밍3

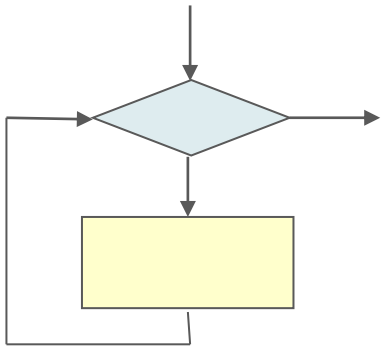
week 3-2 반복문

2022.1학기
충남대 조은선

while 문 vs. do-while문

```
int count = 1;
int sum = 0;
while (count < 11){
    sum += count;
    count ++;
}
```

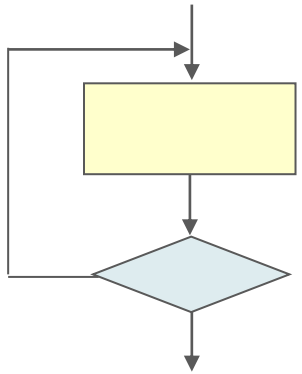
sum = 0+1+2+…10



동일한 결과

```
int count = 1;
int sum = 0;
do {
    sum += count;
    count ++;
} while (count < 11);
```

sum = 0+1+2+…10



while 문 vs. do-while문

while과 do-while의 차이는?
do-while은 무조건 한번은 실행

```
int count = 11;
int sum = 0;
while (count < 10){
    sum += count;
    count ++;
}
```

sum 은 0

다른 결과

```
int count = 11;
int sum = 0;
do {
    sum += count;
    count ++;
} while (count < 10);
```

sum 은 11

Java에서도 그랬다!

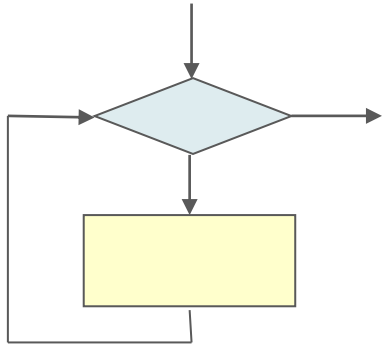
for 문

- 겉 모습

```
for (initialization; test; increment){
    statement;
    statement;
    ...
}
```

```
int count, sum = 0;
for (count = 1; count <= 10; count++)
     sum += count;
```

- 실행 순서
  - initialization ➔ test ➔ statement (또는 루프 바깥) ➔ increment



- for의 각 요소는 생략 가능함

```
for ( ; a < 10; )          // while 루프와 동일
for ( a = 1; a < 10;  )    // 증감식이 루프 내부에 있어야
for ( a = 1;    ; a++)     // 조건식이 루프 내부에 있어야
```

for 문에서 알아둘 것

while	for

count = 1;
while (count <= 10){
  sum += count;
  count ++;
}
	for (count = 1; count <= 10; count++)
	  sum += count;

- 콤마 연산자를 사용하는 경우가 종종 있음

```
for (sum = 0, count = 1; count <= 10; ) ...
```

- for 문에 증감식이 있으면 제어문 내의 증감식은 가급적 피할 것-오류 가능성

```
for (a = 1; a < 10; a++){
   if (a % 2 == 0)
      sum += a;
   a++;
}
```

- Visual C는 `for (int i = 1; i <= 10; i++)`를 허용 - 호환성을 위해 권장안함

중첩 loop

- 구구단 코드

```
int i = 1;
int j = 1;
for (i = 1; i <= 9 ; i++)
    for (j = 1; j <= 9 ; j++)
            printf("%d x %d = %d\n", i, j, i*j);
```

← cartesian 곱의 순서로 실행됨

```
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
...
1 x 9 = 9
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
...
9 x 9 = 81
```

break

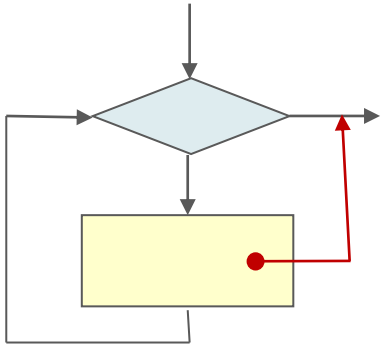
- break;
  - 무조건 탈출; 어떠한 loop나 switch에서;
  - 하나의 레벨만 탈출 가능

```
while ( ... )
{
    for ( ... )
    {
        ...
        if ( 조건식 ) break;
    }
}
```

안쪽 for문 하나만 탈출합니다!

```
while (조건식)
{
    ...
    if (조건식)
    {
        ...
        break;
    }
}
```

반복문 블록 전체를 벗어납니다!

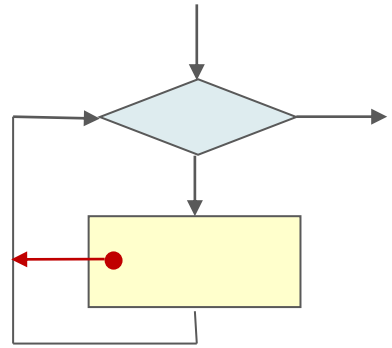


continue

- `continue;`
  - 반복의 나머지 부분을 건너뛰지만, loop를 탈출하지는 않음

```
for ( i = 1; i <= 100; i++)
{
    if ( (i % 3) == 0)
    {
        continue;
    }
    sum += i;
}
```

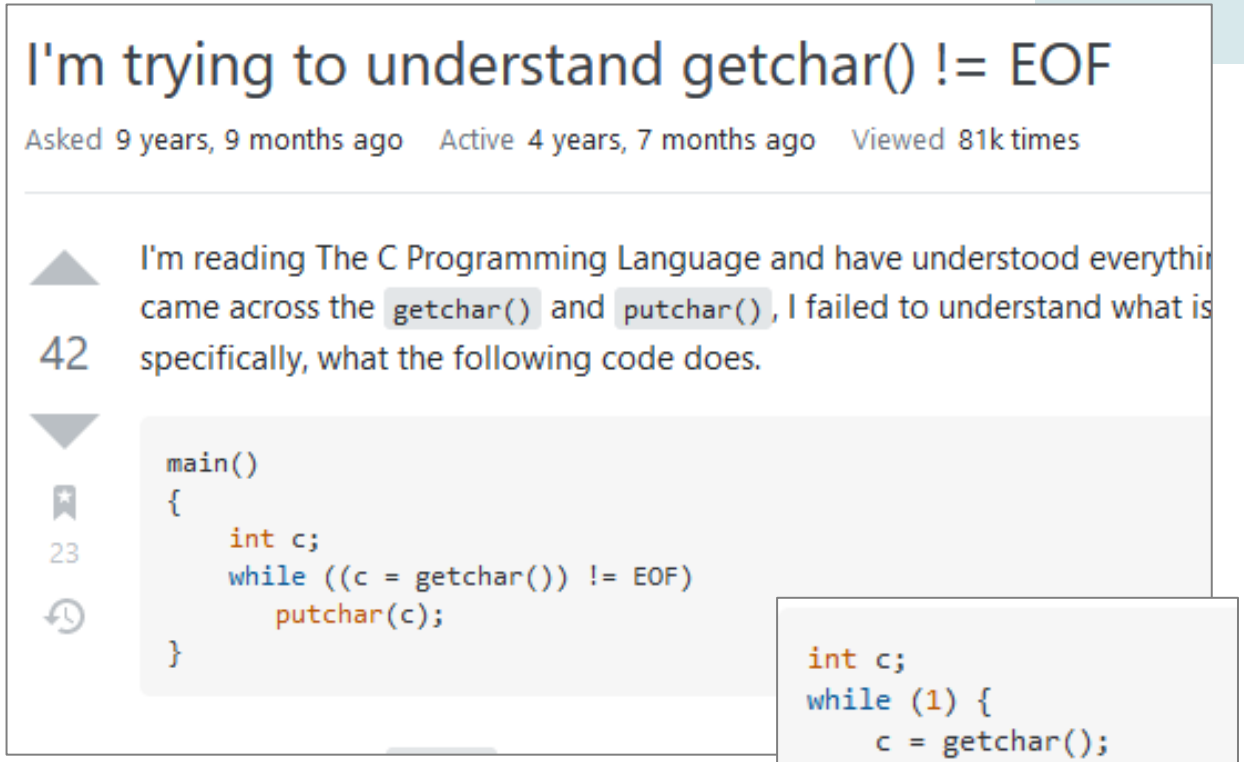
i가 3의 배수면 sum += i 문장을 건너뛰고 블록 끝으로 간 후에 다시 반복합니다.



C 반복문 예제

- EOF
  - End of File를 만나거나, 입출력 실행 중 오류가 일어날 때의 리턴 값 (-1)
  - 키보드 입력에서 파일 끝을 표시하려면 Ctrl-Z (cf. Ctrl-D in Unix)

- `int getchar( );`
  `int putchar(int ch);`
  - 문자 하나를 입력하거나 출력하는 함수
  - EOF를 입력 받으려면 정수형 변수를 선언
  - 키보드 입력에서 파일 끝을 표시하려면 Ctrl-Z (cf. Ctrl-D in Unix)



I'm trying to understand getchar() != EOF

Asked 9 years, 9 months ago Active 4 years, 7 months ago Viewed 81k times

42

I'm reading The C Programming Language and have understood everythi
came across the `getchar()` and `putchar()`, I failed to understand what is
specifically, what the following code does.

```
main()
{
    int c;
    while ((c = getchar()) != EOF)
        putchar(c);
}
```

```
int c;
while (1) {
    c = getchar();
    if (c != EOF)
        putchar(c);
    else
        break;
}
```

https://stackoverflow.com/questions/10720821/im-trying-to-understand-getchar-eof

goto와 레이블

- 어셈블리의 무조건 분기(Unconditional Branch)에 해당
- 일종의 문장 레이블을 필요로 함
- 목적지는 다음 중 하나를 만족해야함
  - goto 를 포함하고 있는 제어문 또는, goto 문장이 포함되어 있는 statement 그룹 내에 있는 문장 / goto 를 포함하는 statement 들을 포함하는 statement 그룹 내에 있는 문장 / goto 가 있는 서브프로그램을 둘러싸고 있는 서브프로그램의 범위에 있는 문장으로서, 문장그룹의 내부가 아닌 문장
  - 하지만, 절대로 같은 레벨이나 goto보다 더 안쪽으로 nested 된 statement 그룹에는 goto의 목적지가 있을 수 없다.
- 가급적 사용하지 말 것-가독성 저하, 복잡한 코드 양산
  - loop 탈출문은 (continue, break,...) 모두 본질적으로 goto이지만, reducible loop를 만들기 때문에 가독성 안전성 보장. 일반적인 goto는 그렇지 못함

```c
#include <stdio.h>
int main() {
    int sum = 0, count = 1;
LOOP:
    sum += count;
    count++;
    if (count <= 10)
        goto LOOP;
    printf("Sum is %d.\n", sum);
    return 0;
}
```

Quiz

- 다음 코드가 하는 일이 무엇인지 유추해보시오.
  - 주의) while이나 for문은 본문 생략도 가능하다.

```
while(getchar() != '\n');
```