Declaration variable in loop/continue, break

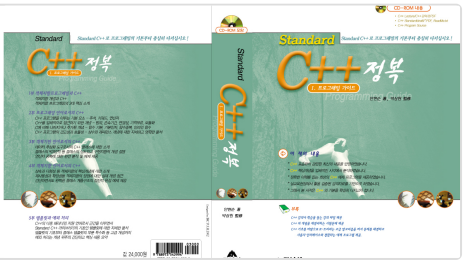
태그	
공개여부	☑
날짜	
작성일자	

Declaration variable in loop (scope, lifetime 개념 복습해야함)

C++ 프로그램의 구조:: 존속기간(lifetime)

Standard C++ 정복 | 프로그램 내의 심볼은 자신이 생성되는 시점과 소멸되는 시점, 즉 존속기간(lifetime)을 가진다. 심볼이 생성된다는 것은 프로그램이 실행되어 메모리 공간을 할당 받는 것을 의미하고 소멸

https://brunch.co.kr/@stdcpp/16



프로그램 내의 심볼은 자신이 생성되는 시점과 소멸되는 시점, 즉 **존속기간(lifetime)**을 가진다. 심볼이 생성된다는 것은 프로그램이 실행되어 메모리 공간을 할당 받는 것을 의미하고 소멸된다는 것은 공간을 반환하는 것이다.

지역 존속기간 (Local Lifetime)

지역 존속기간은 블록이 시작되면 심볼이 생성되고 블록이 끝날 때 소멸되는 것으로 스택에 잡히는 데이터가 이에 해당한다. 지역
존속기간을 갖는 심볼은 지역 범위를 가진다. 그러나 역은 성립하지 않는다. 내부 정적 (static) 변수와 같이 지역 범위를 갖지만 정적
존속기간을 가질 수 있기 때문이다. 범위와 존속기간은 밀접한 관계가 있지만 분명히 다른 개념이다. 또 스택에 잡히지 않으면서도
지역 존속기간을 갖는 변수가 있다. CPU에는 레지스터(register)라는 고속의 메모리가 있는데, 여기에 잡힐 수 있는 정수형
변수도 지역 존속기간을 가질 수 있다. 만일 할당할 레지스터 여유분이 없다면 스택에 잡히

므로 일반적으로 지역 존속기간을 갖는
심볼은 스택에서 삶을 갖는다고 말할 수 있다.

Difference between continue with break

https://ponyozzang.tistory.com/673

C++ 반복문인 **for**문 사용방법을 알아보겠습니다.

반복문이란 동일한 처리 또는 비슷한 처리를 여러 번 해야 하는 경우에

그 처리를 반복해서 실행하도록 하는 문법입니다.

for

for문 작성 방법은 아래와 같습니다.

> for(초기화 ; 조건식 ; 증감식){
>
> **반복할 처리 내용**
>
> **}**

- **초기화**는 처음 **1번만 실행**됩니다.
- **초기화**에는 반복문에서 사용할 **변수를 선언**하는데 사용합니다.
- **조건식**이 **true**일 경우 처리를 **반복**해서 합니다.
- **증감식**에는 변수 값의 **변화 처리**를 작성합니다.

```
#include <iostream>using namespace std;

int main() {
  for (int i = 0; i < 3; i++) {
    cout <<  i << "\n"; // 0 1 2 출력
  }
  return 0;
}
```

결과

> 0

> **1**
>
> **2**

초기화에서는 변수 i에 0을 대입했습니다.

증감식에서는 반복 처리가 한번 실행될 때마다 i가 1씩 증가하도록 지정했습니다.

조건식인 i <3은 변수 i가 3보다 작을 경우 처리를 하도록 하고 있습니다.

변수 i가 3보다 작으면 **true**, 3보다 크면 **false**로 반복 처리를 종료합니다.

break

for문 조건식이 **true**이지만 종료하고 싶은 경우에 **break**를 사용합니다.

```
#include <iostream>using namespace std;

int main() {
  for (int i = 0; i < 3; i++) {
    if (i == 1) {
      break;
    }
    cout << "i =" << i << "\n"; // i=0 출력
  }
  return 0;
}
```

결과

> i =0

for문 조건식은 변수 i가 3보다 작을 동안 처리를 반복하도록 되어있습니다.

하지만 **for**문 안에서 if 조건을 사용해 변수 i가 1과 같으면 **for**문을 종료하도록 하는 **break**를 작성했습니다.

continue

for문에 작성한 조건식과 일치하지만 특정 조건에서는 처리를 하지 않고 넘어가고 싶은 경우에는 **continue**를 사용합니다.

```
#include <iostream>using namespace std;

int main() {
  for (int i = 0; i < 3; i++) {
    if (i == 1) {
```

```
      continue;
    }
    cout << "i =" << i << "\n"; // i=0 i=2 출력
  }
  return 0;
}
```

결과

> i =0
>
> **i =2**

continue와 **break**의 차이점은 **break**의 경우에는 **for**문을 종료하지만, **continue**는 **for**문을 종료하지 않고

지정한 조건만 처리하지 않습니다.

샘플에서는 변수 i가 1일 경우에 **continue** 하도록 되어있기 때문에 결과에는 출력되지 않았습니다.