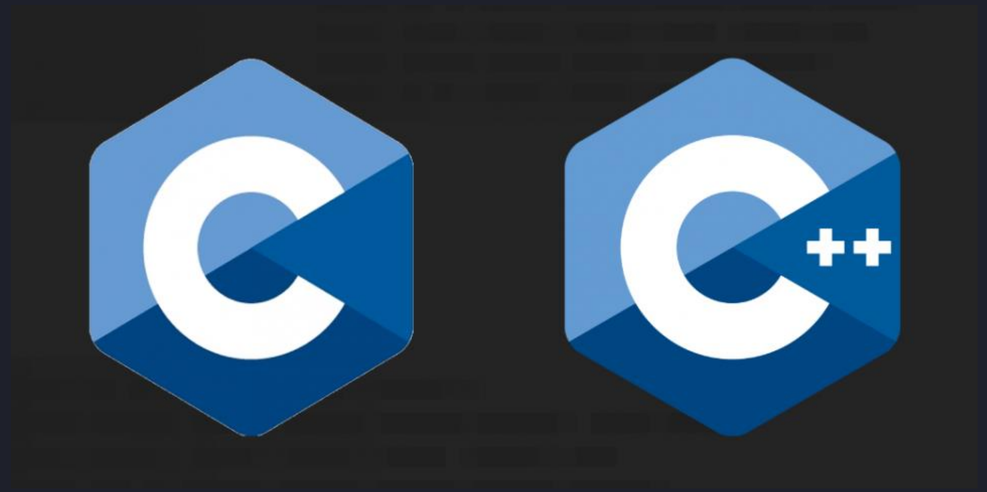


SPLIT 튜터링 3회차
:조건문과 반복문

기계항공공학부 17학번 김기성

Contents

복습

- array를 선언하는 방법
  - type array_name[array_size] = { /*comma_divided_elements*/};
  - array의 쓰임새를 잘(?) 생각해보고 남더라도 여유있게 사이즈를 설정해야 합니다.
    - 런타임(프로그램 실행중)에는 사이즈를 바꿀 수 없습니다.
- while문 사용방법
  - while(조건){}
  - while문 내부에는 while문 종료조건에 가까워지도록 하는 코드가 포함되어야 합니다(무한루프 방지).

```cpp
#include <iostream>

using namespace std;

int main() {
	int array1[5] = { 1, 2, 3, 4, 5 };
	int array2[] = { 1, 2, 3, 4, 5 }; //{1, 2, 3, 4, 5}
	int array3[5] = { 1, 2, 3 }; //{1, 2, 3, 0, 0}
	int array4[5] = { 0 }; //{0, 0, 0, 0, 0}
	return 0;
}
```

```cpp
#include <iostream>

using namespace std;

int main() {
	int count = 1;
	while (count <= 5) {
		cout << count << "번째 루프입니다." << endl;
		count += 1;
	}
	return 0;
}
```

for문

- for(초기화식 ; 조건식 ; 반복식){/*body*/} 의 구조입니다.
  - 반복문 중 정형화된 작업에 유리합니다. 그래서 더 자주 쓰이는 형태입니다.
    - 정해진 횟수 반복하기
    - 배열이나 문자열의 원소 순서대로 접근하기
  - human error의 가능성이 적습니다.
- 반면 while문은 어떤 조건이 충족되기 전까지 반복하도록 할 때 유용합니다.

```
int i = 0;
while (i < 10) {
    cout << i << "번째 루프입니다.\n";
    i += 1;
}
```

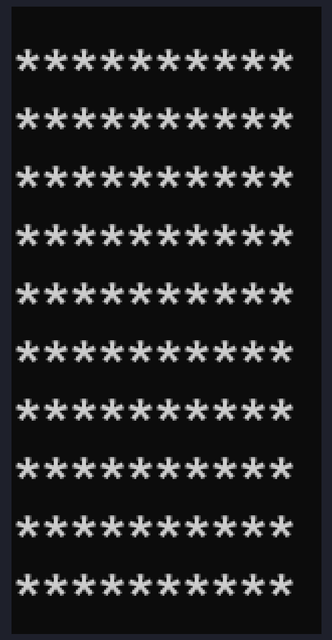
```
for (int i = 0; i < 10; i += 1) {
    cout << i << "번째 루프입니다.\n";
}
```

```
몇 단을 출력할까요?:5
5단을 출력합니다.
5x1 = 5
5x2 = 10
5x3 = 15
5x4 = 20
5x5 = 25
5x6 = 30
5x7 = 35
5x8 = 40
5x9 = 45
```

for문

- 2중 반복문도 가능합니다.
- 두 반복문에서 사용하는 반복자(i, j)를 다르게 선언해주어야 합니다.

```
for (int i = 0; i < 10; i += 1) {
    for (int j = 0; j < 10; j += 1) {
        cout << '*';
    }
    cout << endl;
}
```



조건문: if, else if, else

- 조건문은 프로그램의 흐름을 조절하기 위한 중요한 구조입니다.
  - 조건에 따라 다른 코드를 실행할 수 있습니다.
- if(조건){} else if(조건){} … else if(조건){} else{}

```
#include <iostream>
using namespace std;
int main() {
    int k;
    cout << "k를 입력하세요:";
    cin >> k;
    if (k % 2 == 1) {
        cout << "홀수입니다." << endl;
    }
    else if (k % 2 == 0) {
        cout << "짝수입니다." << endl;
    }
}
```

```
k를 입력하세요:4
짝수입니다.
```

```
k를 입력하세요:5
홀수입니다.
```

조건문: if, else if, else

- 연습문제) 유저에게 0~100점 사이의 정수를 입력받아 다음의 기준에 따라 학점을 출력하는 프로그램을 작성하세요,
  - 90점 이상: A, 80점 이상: B, 70점 이상: C, 60점 이상: D, 그 이외: F

```
점수를 입력하세요:90
A
```

```
점수를 입력하세요:80
B
```

```
점수를 입력하세요:70
C
```

조건문: if, else if, else

- 솔루션)
- else나 else if를 사용하는 것과 if만 사용하는 것은 어떤 차이가 있을까요?
  - if, else if, ...는 앞에서 참이면 그 이후의 else, else if 블록은 실행되지 않습니다.

```cpp
#include <iostream>
using namespace std;
int main() {
	int score;
	cout << "점수를 입력하세요:";
	cin >> score;
	if (score >= 90) {
		cout << "A" << endl;
	}
	else if (score >= 80) {
		cout << "B" << endl;
	}
	else if (score >= 70) {
		cout << "C" << endl;
	}
	else if (score >= 60) {
		cout << "D" << endl;
	}
	else {
		cout << "F" << endl;
	}
}
```

```cpp
#include <iostream>
using namespace std;
int main() {
	int score;
	cout << "점수를 입력하세요:";
	cin >> score;
	if (score >= 90) {
		cout << "A" << endl;
	}
	if (score >= 80) {
		cout << "B" << endl;
	}
	if (score >= 70) {
		cout << "C" << endl;
	}
	if (score >= 60) {
		cout << "D" << endl;
	}
	if (score < 60) {
		cout << "F" << endl;
	}
}
```

break, continue: 반복문 제어하기

- break: 현재 반복을 중단하고 반복문을 탈출합니다.
- continue: 현재 반복을 중단하고, 다음 반복으로 이동합니다.
- 보통 if문을 이용해 조건에 따라 중단, 탈출하는 로직을 사용합니다.

```
제가 생각하고 있는 정수를 맞혀보세요:5
틀렸습니다.
제가 생각하고 있는 정수를 맞혀보세요:4
틀렸습니다.
제가 생각하고 있는 정수를 맞혀보세요:3
틀렸습니다.
제가 생각하고 있는 정수를 맞혀보세요:7
틀렸습니다.
제가 생각하고 있는 정수를 맞혀보세요:10
맞혔습니다!
```

```cpp
#include <iostream>

using namespace std;

int main() {
	int my_integer = 10;
	while (true) {
		cout << "제가 생각하고 있는 정수를 맞혀보세요:";
		int input;
		cin >> input;
		if (input == my_integer) {
			cout << "맞혔습니다!" << endl;
			break;
		}
		else {
			cout << "틀렸습니다." << endl;
		}
	}
	return 0;
}
```

```cpp
#include <iostream>

using namespace std;

int main() {
	int my_integer = 10;
	while (true) {
		cout << "제가 생각하고 있는 정수를 맞혀보세요:";
		int input;
		cin >> input;
		if (input != my_integer) {
			cout << "틀렸습니다." << endl;
			continue;
		}
		cout << "맞혔습니다!" << endl;
		break;
	}
	return 0;
}
```

연습문제

- 연습문제 1)
  - 정수 N을 입력받습니다. N은 100000이하입니다.
  - 유저로부터 N개의 정수 $a_0, \ldots, a_{n-1}$을 입력받아 array a에 저장합니다.
  - a에 저장된 정수 중 최대값과 최소값을 구합니다.

```
N을 입력하세요:5
5개의 정수를 입력하세요:5 10 100 -2 5
최소값: -2
최대값: 100
```

- 연습문제 2)
  - 정수 N을 입력받습니다.
  - 1부터 N까지의 정수를 출력하되 3의 배수는 숫자 대신 Fizz를, 5의 배수는 Buzz를, 3의 배수이면서 5의 배수이면 FizzBuzz를 출력하는 코드를 작성하세요.
  - hint1: 어떤 조건을 제일 먼저 검사해야 할까요?
  - hint2: C++에서 and연산자는 '&&'입니다.

```
N을 입력하세요:15
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
```

연습문제

- 연습문제 3)
  - 유저로부터 정수 N을 입력받습니다.
  - 정수 N이 소수(prime number)면 “소수입니다.”를, 합성수일 경우 “합성수입니다.”를 출력하세요.
  - 소수는 1과 자기 자기 자신만 약수로 갖는 수입니다.
    - 소수인지 판별하려면, 2부터 N-1까지 나누어 떨어지는 수가 있는지 검사하면 됩니다.
      - 여담) 정말 2부터 N-1까지 모두 검사해야 할까요? 반복문을 도는 횟수를 최소화해봅시다.

```
N을 입력하세요:10
합성수입니다.
```

```
N을 입력하세요:1000000007
소수입니다.
```

여담-증가 연산자, 감소 연산자

- 파이썬에는 없는 일항연산자(unary operator)입니다.
- integer변수의 앞이나 뒤에 ++, --가 붙으면 값을 증가시키거나 감소시킵니다.
- 연산자가 변수 앞에 붙는지, 뒤에 붙는지에 따라 느낌이 조금 다릅니다.
  - 앞에 붙으면 연산 이전의 값을 반환하고, 뒤에 붙으면 연산 이후의 값을 반환합니다.
  - 모든 연산자, 혹은 함수는 부작용(side-effect)과 리턴값이 있습니다.
  - 즉 부작용은 같으나 리턴값이 다릅니다.

```cpp
#include <iostream>

using namespace std;

int main() {
	int a = 5;
	int b = 5;

	cout << a++ << endl;
	cout << a << endl;

	cout << ++b << endl;
	cout << b << endl;

	return 0;
}
```

	a++	++a
부작용	a += 1	a += 1
리턴값	a-1	a

여담-변수의 수명

- 선언된 변수는 선언된 body 안에서만 유효합니다.
  - 해당 body가 끝나면 그 변수는 사라집니다.
- 전역 변수(global variable)이란, 함수나 블록 안에서 선언되지 않아 어떤 곳에서든 접근 가능한 변수입니다.
  - 좋은 코드는 아니긴 하지만, problem solving 등 짧은 코드에서는 자주 사용 됩니다.

```
#include <iostream>

using namespace std;

int a = 1;
int main() {
    int b = 0;
    {
        int c = 1;
    }
    return 0;
}
```