조건문과 반복문

(Condition statement & Loop)

들어가기 전에

- 오늘의 소스코드를 받아봅시다.
  - 중요한 코드는 꼭 다른 곳으로 복사해두세요.
  - phycom2015-2 디렉토리에서,
    - git stash
    - git pull
  - 디렉토리가 사라졌다면,
    - git clone https://github.com/geonmo/phycom2015-2.git

조건문의 핵심 "IF"

- if는 조건의 참과 거짓을 이용하여 해당 scope를 실행하는 문법입니다.
- if문의 사용법

```
if( 조건A ) {
   "Something to do if 조건 A is true"
}
else if( 조건B ) {
   "Something to do if 조건 A is false but B is true"
else {
   "Something to do if A and B are false"
}
```

조건문의 핵심 "IF"

- if문의 예제
  - Week2_condition₩&loop/4_condition_if.cpp

```
#include<iostream>

int main()
{
  unsigned int age = 0;
  std::cout<<"Please, input your age :";
  std::cin>>age;
  if ( age >= 8 && age <=13) {
    std::cout<<"You are a elementary school."<<std::endl;
  }
  else {
    std::cout<<"Wow!"<<std::endl;
  }
  return 0;

}
```

실습(IF문)

- 위 프로그램은 초등학생만을 대상으로 작동하는 프로그램 코드입니다.
  - 유아(<=7세), 중학생, 고등학생, 대학생, 성인으로 보다 세분화하여 프로그램을 수정해봅시다.
    - 조건 : 군대, 유급, 추가학기, 졸업유예 등은 무시!
  - Test case :
    - Input variables : 5 // 12 // 15 // 18// 20 // 24 // 65

Switch-Case 문

- 하나의 변수가 여러 조건에 따라 나눠질 때 사용됨
- 잘 사용되지는 않음

```
int num = 3;
  switch( num ){
      case 1:
          printf("num의 값은 1입니다.");
          break;
      case 2:
          printf("num의 값은 2입니다.");
          break;
      case 3:
          printf("num의 값은 3입니다.");
          break;
      case 4:
          printf("num의 값은 4입니다.");
          break;
      default:
          printf("num의 값은 5입니다.");
          break;
  }
```

반복문 for

- for문은 특정 조건에 따라 일정한 프로그램 루틴을 반복 수행하는 문법입니다.
- for문의 예제

7_for_loop.cpp

```
// for( 초기값 ; 조건값 ; Scope가 끝난 후 할 일)
 for( int i= 0 ; i< 100; i++) {
   std::cout<<"이번에 새로운 i = "<<i<<std::endl;
 }
```

while과 do-while

- while문은 for문에서 초기항과 증감항을 뺀 문법입니다.
- do-while은 처음 한번은 해당 루틴을 실행시켜야 하는 while문을 쓸 때 사용됩니다.

while과 do-while

- while/do-while 예제

8_while_do_while.cpp

```
int a=0;
 while( a<10 ) {
   std::cout<<"Now a is "<<a<<std::endl;
   a++;
 }

 int b = 15;
 do {
   std::cout<<"First trial must be success at do while."<<std::endl;
   std::cout<<"Now b is "<<b<<std::endl;
 } while( b<10 );
```

반복문의 제어 : break, continue

- break
  - 가장 가까운 반복문을 "탈출" 합니다.
- continue
  - 가장 가까운 반복문으로 "복귀"합니다.

10_break_continue.cpp

```
int a=0;
  while ( a<20 ) {
    a++;
    if ( a == 5) {
      continue;
    }
    if ( a== 12) {
      break;
    }
    std::cout<<"This a is "<<a<<std::endl;
  }
```

```
This a is 1
This a is 2
This a is 3
This a is 4
This a is 6
This a is 7
This a is 8
This a is 9
This a is 10
This a is 11
```

실습(반복문)

- 독일의 수학자 "가우스"는 1~100까지를 더하라는 선생님의 지시를 받고 101에 50을 "곱하여" 답을 제출하였다고 합니다.
- 더하기를 연습시키려던 선생님의 깊은 뜻을 이해하지 못한 가우스를 대신하여 1~10000까지의 합을 "더하여" 계산해봅시다.
- 또한, 가우스의 방법으로도 코드를 작성하여 답이 같은지 비교해봅시다.

실습(반복문)

- 1~10만까지를 더하면 답이 맞게 나오나요?
  - 아니라면 왜 그럴까요?

- 가우스의 편법에 화가 난 선생님이 학생들에게 이번엔 1~10000까지 중에서 소수(prime number)를 빼고 더하라는 지시를 하셨다면 그 결과값은 얼마인가요?
  - Tip. 코드가 잘 짜졌는지 범위를 1~10으로 줄여 먼저 확인하세요. (Test case)
- 마지막 문제까지 푸셨다면 나가셔도 좋습니다!