완전 기초 스터디!

세 번째 모임

키-파

May 4, 2018

목차

± (w2-h1)

- 정답자 6명
- 연립방정식을 풀면

$$C = \frac{A+B}{2},\ D = \frac{A-B}{2}$$

  입니다
- 코드 (submitted by kiwan):

```
#include <cstdio>
int main(){
    int A, B, C, D;
    scanf("%d%d",&A, &B);
    C=(A+B)/2;
    D=(A-B)/2;
    printf("%d\n%d", C,D);
    return 0;
}
```

진정한 소수의 배수 확인 (w2-h2)

- 정답자 5명
- 일의 자리는 10으로 나눈 나머지
- 십의 자리 이상은 10으로 나눈 몫
- 코드 (submitted by Coffeetea):

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

int main(){
    int a, b,c,d;
    scanf("%d",&a);
    b=a%10;
    c=(a-b)/10-b*2;
    d=c%7;
    printf("%d\n%d",c,d);
    return 0;
}
```

진정한 소수의 배수 확인 (w2-h2)

- 75점 코드 (submitted by kiwan):

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

int main(){
    int n, a;
    scanf("%d", &n);
    a=n/10-2*n%10;
    printf("%d\n%d", a, a%7);
    return 0;
}
```

진정한 소수의 배수 확인 (w2-h2)

- 75점 코드 (submitted by kiwan):

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

int main(){
    int n, a;
    scanf("%d", &n);
    a=n/10-2*n%10;
    printf("%d\n%d", a, a%7);
    return 0;
}
```

- 연산자 우선순위가 헷갈리면 괄호를 치세요!
- `2*(n%10)` 혹은 `n%10*2`로 고치면 작동합니다

그 달의 일 수 (w2-h3)

- 정답자 5명
- 의도한 풀이 : 30일인 달 수가 4개밖에 안 되므로 이를 이용

```
31 * (1<=x&&x<=12) - (x==4||x==6||x==9||x==11) - 3 * (x==2)
```

- 코드 (submitted by rladusdnd):

```
#include <cstdio>
using namespace std;
int main(){
    int a,b;
    scanf("%d",&a);
    b=30+((a==1)||(a==3)||(a==5)||(a==7)||(a==8)||(a==10)||(a==12))
        -30*(a>=13)-2*(a==2)-30*(a<=0);
    printf("%d",b);
    return 0;
}
```

그 달의 일 수 (w2-h3)

- 100점 코드 (submitted by Coffeetea):

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

int main(){
    int a,b,c,d,e;
    scanf("%d",&a);
    b=(a==1)+(a==3)+(a==5)+(a==7)+(a==8)+(a==10)+(a==12);
    c=(a==4)+(a==6)+(a==9)+(a==11);
    d=(a==2);
    e=(b*31+c*30+d*28)*(13>a>0);
    printf("%d",e);
    return 0;
}
```

그 달의 일 수 (w2-h3)

- 100점 코드 (submitted by Coffeetea):

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

int main(){
    int a,b,c,d,e;
    scanf("%d",&a);
    b=(a==1)+(a==3)+(a==5)+(a==7)+(a==8)+(a==10)+(a==12);
    c=(a==4)+(a==6)+(a==9)+(a==11);
    d=(a==2);
    e=(b*31+c*30+d*28)*(13>a>0);
    printf("%d",e);
    return 0;
}
```

- 13>a와 완전히 같음
- 왜 100점을 받았을까?

그 달의 일 수 (w2-h3)

- 100점 코드 (submitted by Coffeetea):

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

int main(){
    int a,b,c,d,e;
    scanf("%d",&a);
    b=(a==1)+(a==3)+(a==5)+(a==7)+(a==8)+(a==10)+(a==12);
    c=(a==4)+(a==6)+(a==9)+(a==11);
    d=(a==2);
    e=(b*31+c*30+d*28)*(13>a>0);
    printf("%d",e);
    return 0;
}
```

- 13>a와 완전히 같음
- 왜 100점을 받았을까?
  - hint: e=b*31+c*30+d*28로 계산해도 잘 작동함

세 수 정렬하기 (w2-h4)

- 정답자 **3명**
- 의도한 풀이 :

세 수 정렬하기 (w2-h4)

- 정답자 **3명**
- 의도한 풀이:
  1. 가장 큰 수와 가장 작은 수를 먼저 구함

```
biggest  = (a>=b && a>=c) * a + (b>a && b>=c) * b + (c>a && c>b) * c
smallest = (a<=b && a<=c) * a + (b<a && b<=c) * b + (c<a && c<b) * c
```

     - 부등호의 사용을 눈여겨 볼 것

세 수 정렬하기 (w2-h4)

- 정답자 **3명**
- 의도한 풀이:
  1. 가장 큰 수와 가장 작은 수를 먼저 구함

```
biggest  = (a>=b && a>=c) * a + (b>a && b>=c) * b + (c>a && c>b) * c
smallest = (a<=b && a<=c) * a + (b<a && b<=c) * b + (c<a && c<b) * c
```

     - 부등호의 사용을 눈여겨 볼 것
  2. 중간 수는 다음과 같이 구함

```
middle = (a + b + c) - (biggest + smallest)
```

세 수 정렬하기 (w2-h4)

- 정답자 **3명**
- 의도한 풀이:
  1. 가장 큰 수와 가장 작은 수를 먼저 구함

```
biggest  = (a>=b && a>=c) * a + (b>a && b>=c) * b + (c>a && c>b) * c
smallest = (a<=b && a<=c) * a + (b<a && b<=c) * b + (c<a && c<b) * c
```

     - 부등호의 사용을 눈여겨 볼 것
  2. 중간 수는 다음과 같이 구함

```
middle = (a + b + c) - (biggest + smallest)
```

  3. 순서대로 출력

```
printf("%d %d %d\n", smallest, middle, biggest);
```

세 수 정렬하기 (w2-h4)

- 코드 (submitted by gratus907):

```
#include <cstdio>
#pragma warning (disable:4996)
using namespace std;

int main() {
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);

    int Compare_1 = b >= a;
    int p, q, r;
    p = 0;
    q = 0;
    r = 0;
    (Compare_1 == 1) && (p = b, q = a, r = c);
    (Compare_1 == 0) && (p = a, q = b, r = c);

    (Compare_1 == 1 && c >= b) && (p = c, q = b, r = a);
    (Compare_1 == 1 && c < b && c >= a) && (p = b, q = c, r = a);
    (Compare_1 == 1 && c < a) && (p = b, q = a, r = c);

    (Compare_1 == 0 && c >= a) && (p = c, q = a, r = b);
    (Compare_1 == 0 && c < a && c >= b) && (p = a, q = c, r = b);
    (Compare_1 == 0 && c < b) && (p = a, q = b, r = c);

    printf("%d %d %d", r, q, p);
}
// Sorting?
```

더하자 (w2-h5)

- 정답자 5명
- 다음 식을 이용하면 됩니다

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

- 코드 (submitted by ty8900):

```
#include <cstdio>
int main()
{int n;
scanf("%d",&n);
printf("%d",n*(n+1)*(2*n+1)/6);
return 0;
}
```

사칙연산? (w2-h6)

- 정답자 5명
- 빨간색 식으로는 $(a^4 - b^4)$ 이 2^{31} 을 넘어 풀 수 없습니다
  파란색 식으로는 중간 계산 과정이 2^{31} 을 넘지 않아 풀 수 있습니다

$$\frac{a^4 - b^4}{a + b} = (a^2 + b^2)(a - b)$$

  - overflow라고 하며, 이 스터디 후반부에 다룰 주제 중 하나입니다
- 코드 (submitted by ty8900):

```
#include <cstdio>
int main(){
    int a,b,c;
    scanf("%d%d",&a,&b);
    c=(a*a+b*b)*(a-b);
    printf("%d",c);
}
```

사칙연산? (w2-h6)

- read well, think more and write less.
- 코드 (submitted by gratus907):

```
#include <cstdio>
#pragma warning(disable: 4996)

int main()
{
    int a, b;
    scanf("%d %d", &a,&b);
    int sgn;
    sgn = (a-b);
    sgn = (sgn > -1 * sgn);
    (sgn == 0) && printf("%d", -1);
    (sgn == 1) && printf("%d", ((a*a + b * b) * (a - b)));
}
```

사칙연산? (w2-h6)

- read well, think more and write less.
- 코드 (submitted by gratus907):

```
#include <cstdio>
#pragma warning(disable: 4996)

int main()
{
    int a, b;
    scanf("%d %d", &a,&b);
    int sgn;
    sgn = (a-b);
    sgn = (sgn > -1 * sgn);
    (sgn == 0) && printf("%d", -1);
    (sgn == 1) && printf("%d", ((a*a + b * b) * (a - b)));
}
```

- 문제에 a $>$ b라는 조건이 있습니다

하나, 둘, 셋, 많다 (w2-h7)

- 정답자 **0명**

하나, 둘, 셋, 많다 (w2-h7)

- 정답자 **0명**
- a, b의 크기 제한이 10^6 이고, ab $\leq 10^{12}$ 인데, $10^{12} \gg 2^{31}$ 입니다
  - 400배 이상 차이

하나, 둘, 셋, 많다 (w2-h7)

- 정답자 **0명**
- a, b의 크기 제한이 10^6 이고, ab $\leq 10^{12}$ 인데, $10^{12} \gg 2^{31}$ 입니다
  - 400배 이상 차이
- 의도한 풀이 (often referred to as cutting):

하나, 둘, 셋, 많다 (w2-h7)

- 정답자 **0명**
- a, b의 크기 제한이 10^6 이고, ab $\leq 10^{12}$ 인데, $10^{12} \gg 2^{31}$ 입니다
  - 400배 이상 차이
- 의도한 풀이 (often referred to as cutting):
  1. a, b의 절댓값을 계산

```
absa = ((a > 0) - (a < 0)) * a;
absb = ((b > 0) - (b < 0)) * b;
```

하나, 둘, 셋, 많다 (w2-h7)

- 정답자 **0명**
- a, b의 크기 제한이 10^6 이고, ab $\leq 10^{12}$ 인데, $10^{12} \gg 2^{31}$ 입니다
  - 400배 이상 차이
- 의도한 풀이 (often referred to as cutting):
  1. a, b의 절댓값을 계산

```
absa = ((a > 0) - (a < 0)) * a;
absb = ((b > 0) - (b < 0)) * b;
```

  2. c $=$ ab와 c가 출력될 조건을 계산

```
c = a * b;
cond = (0 <= c && c <= 3)
    && (absa <= 4 || absb <= 4);
```

  3. 조건에 맞게 출력

하나, 둘, 셋, 많다 (w2-h7)

- 정답자 **0명**
- a, b의 크기 제한이 10^6 이고, ab $\leq 10^{12}$ 인데, $10^{12} \gg 2^{31}$ 입니다
  - 400배 이상 차이
- 의도한 풀이 (often referred to as cutting):
  1. a, b의 절댓값을 계산

```
absa = ((a > 0) - (a < 0)) * a;
absb = ((b > 0) - (b < 0)) * b;
```

  2. c $=$ ab와 c가 출력될 조건을 계산

```
c = a * b;
cond = (0 <= c && c <= 3)
    && (absa <= 4 || absb <= 4);
```

  3. 조건에 맞게 출력
- some false positives (assuming `int` multiplication):

```
 65536  65536
137283  62571
125142 137283
 69817 123035
137283 187713
```

나머지 (w2-h8)

- 정답자 5명

나머지 (w2-h8)

- 정답자 5명
- a가 음수인 경우, `a % b`의 값이 음수라서 이 경우가 말썽입니다
  - 문제에서 정의한 나머지는 수학에서 아주 유용하며, python 등의 프로그래밍 언어에서 이런 나눗셈을 지원합니다
  - 컴퓨터에게 유용한 나눗셈은 몫은 a와 b의 부호가 같으면 양수, 나머지의 부호는 a의 부호 쪽으로 처리하는 것입니다
- `a % b`가 음수인 경우 |b|를 더해 이 문제를 해결할 수 있습니다
- 코드 (submitted by ty8900):

```
#include <cstdio>
int main()
{
    int a,b,bb,rest;
    scanf("%d%d",&a,&b);
    rest=a%b;
    bb=b*(b>=0)-b*(b<0);
    printf("%d",(rest>=0)*rest+(rest<0)*(rest+bb));
    return 0;
    }
```

짧은 회로 평가 (w2-h9)

- 정답자 **2명**
- 문제에서 시키는 대로 하면 됩니다
  - 문제 서술이 애매해 두 가지로 해석할 수 있지만, 예제 넣어 보면 한쪽의 해석은 아예 불가능함을 쉽게 알 수 있습니다
- 코드 (submitted by ty8900):

```
#include <cstdio>
int main()
{int y;
scanf("%d",&y);
((y%400==0)||((y%4==0)&&(y%100!=0)))||printf("common");
((y%400!=0)&&((y%4!=0)||(y%100==0)))||printf("leap");
return 0;
}
```

불완전한 TC (w2-h10)

- 정답자 **2명**

불완전한 TC (w2-h10)

- 정답자 **2명**
- 디스크립션을 잘 읽으면 TC는 **불안전할 뿐 불완전하지 않다**는 사실을 알 수 있는데 안타깝게도 이를 지적한 분은 없었습니다...
  - **완전한 TC**나 **불안전한 TC**가 맞는 말이지만 제목이 좀 틀릴 수도 있죠 뭐

불완전한 TC (w2-h10)

- 정답자 **2명**
- 디스크립션을 잘 읽으면 TC는 **불안전할 뿐 불완전하지 않다**는 사실을 알 수 있는데 안타깝게도 이를 지적한 분은 없었습니다...
  - **완전한 TC**나 **불안전한 TC**가 맞는 말이지만 제목이 좀 틀릴 수도 있죠 뭐
- 많은 풀이 방법이 있지만, 예를 들어

```
scanf("%d", &x); printf("%d\n", x*x*(x>k));
```

의 코드를 제출하면 첫 번째 TC의 입력받은 값이 k보다 크면 **맞았습니다**, 작거나 같으면 **틀렸습니다**가 나오게 됩니다
  - 매 구간마다 정답이 될 수 있는 수의 크기가 반으로 줄어듭니다
  - $10^3 < 2^{10}$ 이므로 10번이면 정답을 알 수 있습니다

불완전한 TC (w2-h10)

- 코드 (submitted by gratus907):

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

int main()
{
    int n;
    scanf("%d", &n);
    n = ???;
    printf("%d\n", n*n);
}
/* 아 첨에 문제이해 못해서 실행횟수 날려먹었다.....ㅠㅠㅠㅠ
계산실수 안했으면 한번 더 줄일수있었는데 ㅠㅠㅠ
SNUPS 파이팅 열심히할게요*/
```

- 이런 코드 좋아요 감사합니다

조건문

- 다음의 코드를 정확하게 따라 입력합니다

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

int main() {
    int v;
    scanf("%d", &v);
    if (v == 1) {
        printf("v is the unit.\n");
    }
    if (v % 2 == 0) {
        printf("v is even.\n");
    } else {
        printf("v is odd.\n");
    }
    printf("abs value is %d.\n", v > 0 ? v : -v);
    return 0;
}
```

- 1을 입력해서 한 번, -4를 입력해서 한 번 돌려 봅니다

조건문 뜯어보기

조건문 뜯어보기 전에

- 앞으로 절대 `(v > 0) * v` 같은 코드를 쓰지 마십시오
  - 여러분의 연산자에 대한 고정관념을 깨기 위한 어쩔 수 없는 선택이었습니다
  - 지금부터 훨씬 명시적으로 **이 조건이면 이것**을 표현할 수 있는 방법을 배웁니다
- 조건에 따라 다른 코드를 실행하게 하는 코드인 `if`를 **조건문**이라 합니다

조건문 뜯어보기

- `if (v == 1) {`

조건문 뜯어보기

- `if (v == 1) {`
  - v가 1인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 1이면 실행됩니다

조건문 뜯어보기

- `if (v == 1) {`
  - v가 1인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 1이면 실행됩니다
- `if (v % 2 == 0) {`

조건문 뜯어보기

- `if (v == 1) {`
  - v가 1인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 1이면 실행됩니다
- `if (v % 2 == 0) {`
  - v를 2로 나눈 나머지가 0인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 **짝수**이면 실행됩니다

조건문 뜯어보기

- `if (v == 1) {`
  - v가 1인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 1이면 실행됩니다
- `if (v % 2 == 0) {`
  - v를 2로 나눈 나머지가 0인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 **짝수**이면 실행됩니다
- `} else {`

조건문 뜯어보기

- `if (v == 1) {`
  - v가 1인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 1이면 실행됩니다
- `if (v % 2 == 0) {`
  - v를 2로 나눈 나머지가 0인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 **짝수**이면 실행됩니다
- `} else {`
  - 만약 아니라면...
  - 중괄호 속의 문장은 아니라면 실행합니다

조건문 뜯어보기

- `if (v == 1) {`
  - v가 1인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 1이면 실행됩니다
- `if (v % 2 == 0) {`
  - v를 2로 나눈 나머지가 0인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 **짝수**이면 실행됩니다
- `} else {`
  - 만약 아니라면...
  - 중괄호 속의 문장은 아니라면 실행합니다
  - **무엇에 대해** 아니라면?
    - `else`는 단독으로 존재할 수 없습니다
    - 선행되는 조건이 있어야!

조건문 뜯어보기

- `if (v == 1) {`
  - v가 1인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 1이면 실행됩니다
- `if (v % 2 == 0) {`
  - v를 2로 나눈 나머지가 0인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 **짝수**이면 실행됩니다
- `} else {`
  - 만약 아니라면...
  - 중괄호 속의 문장은 아니라면 실행합니다
  - **무엇에 대해** 아니라면?
    - `else`는 단독으로 존재할 수 없습니다
    - 선행되는 조건이 있어야!
- `v > 0 ? v : -v`

조건문 뜯어보기

- `if (v == 1) {`
  - v가 1인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 1이면 실행됩니다
- `if (v % 2 == 0) {`
  - v를 2로 나눈 나머지가 0인가? 만약 그렇다면...
  - 중괄호 속의 문장은 v가 **짝수**이면 실행됩니다
- `} else {`
  - 만약 아니라면...
  - 중괄호 속의 문장은 아니라면 실행합니다
  - **무엇에 대해** 아니라면?
    - `else`는 단독으로 존재할 수 없습니다
    - 선행되는 조건이 있어야!
- `v > 0 ? v : -v`
  - v가 0보다 큰가? 그렇다면 v가, 아니라면 -v가 결과
  - v의 **절댓값**을 계산합니다

if

- if의 소괄호 안에 있는 조건이 0이 아니라면, 뒤에 나오는 중괄호 안의 문장을 실행합니다
- else가 있다면, if의 소괄호 조건이 아닐 경우 뒤에 나오는 중괄호 안의 문장을 실행합니다
- 지금부터는 단조로운 프로그램 진행을 피할 수 있습니다
- 다음과 같이 쓸 수 있습니다

```
if (cond1) {
    state1;
} else if (cond2) {
    state2;
} else {
    state3;
}
```

if

- if의 소괄호 안에 있는 조건이 0이 아니라면, 뒤에 나오는 중괄호 안의 문장을 실행합니다
- else가 있다면, if의 소괄호 조건이 아닐 경우 뒤에 나오는 중괄호 안의 문장을 실행합니다
- 지금부터는 단조로운 프로그램 진행을 피할 수 있습니다
- 다음과 같이 쓸 수 있습니다

```
if (cond1) {
    state1;
} else if (cond2) {
    state2;
} else {
    state3;
}
```

⇒

```
if (cond1) {
    state1;
} else {
    if (cond2) {
        state2;
    } else {
        state3;
    }
}
```

삼항 연산자

- `cond ? val1 : val2`와 같이 항이 세 개 있는 **연산자**입니다
  - 값으로 평가됩니다
- `cond`가 0이 아니면 계산 값은 `val1`, 0이면 계산 값은 `val2`이 됩니다
- 너무 간단해서 `if`로 쓰기 싫을 때 쓰면 됩니다
- 연산자 우선순위는 일부러 알려드리지 않습니다
  - 외우면 좋지만 외우지 않는 것을 추천합니다
  - 헷갈리면 괄호를 치세요
- 연산자는 아주 중요하지 않으면 더 이상 슬라이드에 의미를 적어 두지 않겠습니다
  - 여러분이 직접 `c++ operator` 등을 검색해서 찾아 보세요
  - 물론 저한테 물어보셔도 좋습니다

연습

- 문제는 **w2-h9**와 **w2-p2**입니다
- `if`와 삼항 연산자를 이용해서 다시 풀어 보세요
  - 훨씬 편해졌나요?

반복문

- 다음의 코드를 정확하게 따라 입력합니다

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

int main() {
    int i;
    for (i=1; i<=10; ++i) {
        printf("the value of i is %d.\n", i);
    }
    return 0;
}
```

- 돌려 봅니다

반복문 뜯어보기

- `for (i=1; i<=10; ++i) {`

반복문 뜯어보기

- `for (i=1; i<=10; ++i) {`
  - `i`를 `1`부터 `10`까지 반복하면서 중괄호 속 문장을 실행한다는 뜻입니다
    - 왜 필요할까요?
  - 초록색에는 임의의 (선언된) 변수를, 빨간색, 파란색에는 임의의 계산 값을 넣을 수 있습니다
    - 예: `for (i=n; i<=2*n; ++i)`

for 1

- `for (i=1; i<=10; ++i) { ... }`
- 일단 헷갈리지 않기 위해서 이 형태로 훈련합시다
  - 다른 형태로 쓰고 싶어도 잠시 잊어버려 주세요 감사합니다
- 변수 이름은 여태까지 보던 그 변수 이름이 맞습니다
  - i로 부족하면 j, k 등을 사용합니다
  - 의미가 있으면 의미가 부여된 이름을 사용할 때도 있습니다: `len` 등...
- 시작점은 처음에 어디서 시작할지를 정합니다
- 끝점은 어디서 끝날지를 정합니다
  - 시작점에서 끝점까지 1씩 더해 가며 (끝점 - 시작점 + 1)회 중괄호 안을 실행합니다

for 1

- `for (i=1; i<=10; ++i) { ... }`
- 일단 헷갈리지 않기 위해서 이 형태로 훈련합시다
  - 다른 형태로 쓰고 싶어도 잠시 잊어버려 주세요 감사합니다
- 변수 이름은 여태까지 보던 그 변수 이름이 맞습니다
  - i로 부족하면 j, k 등을 사용합니다
  - 의미가 있으면 의미가 부여된 이름을 사용할 때도 있습니다: `len` 등...
- 시작점은 처음에 어디서 시작할지를 정합니다
- 끝점은 어디서 끝날지를 정합니다
  - 시작점에서 끝점까지 1씩 더해 가며 (끝점 - 시작점 + 1)회 중괄호 안을 실행합니다
- 사실 `for`는 엄청나게 많이 훈련해야 하는 우리의 영원한 친구입니다
  - 매우 (쉽지만) 다양한 사고방식이 녹아들어 있습니다
  - 프로그래밍을 공부하는 많은 사람들의 1차 장벽입니다
  - 컴퓨터를 컴퓨터이게 해 주는 구조입니다

연습

- 문제는 **w3-p1**입니다
- 백준 문제 2741, 2742, 2739도 연습삼아 풀어볼 만합니다