4190.103A-001: Programming Practice Lab. 5

Euidong Lee
Junseok Lee
Minhyo Jeong
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

Seoul National University

Spring 2022

실습조

실습조

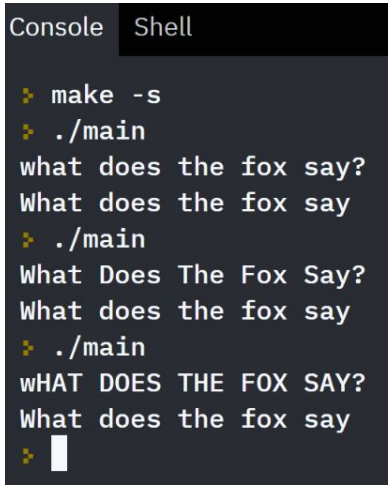
- @MinhyoJung
  - 1조: ‘ㄱ’ ~ ‘ㅅ’ 사이에 해당하는 성을 가진 학생
- @euidong
  - 2조: ‘ㅇ’ ~ ‘ㅎ’ 사이에 해당하는 성을 가진 학생

과제 풀이

Lab 4 과제 1

Lab 4 - 과제1

- getchar()를 이용해 알파벳 문자를 입력 받아 첫 문자는 대문자로, 나머지는 소문자로 변환하여 출력하는 프로그램을 작성하시오.
  - 입력 : 알파벳 문자 (['a','z'],['A','Z']), 개행 문자 (['₩n']), 공백 문자 ([' '])
  - 예외처리 : 위 입력 이외의 character인 경우 무시한다.
  - 출력 : 문제 조건에 따라 case 변경한 문자들
  - 종료조건 : '₩n' 입력되었을 때



Lab 4 - 과제1

- 전체 코드

```c
#include <stdio.h>
#define DISTANCE ('a'-'A')

int main(void)
{
  int c;
  int first = 1;
  while ((c = getchar()) != '\n') {
    if ((c >= 'a' && c <= 'z')) {
      if (first) {
        c -= DISTANCE;
        first = 0;
      }
      putchar(c);
    }
    else if ((c >= 'A' && c <= 'Z')) {
      if (first)
        first = 0;
      else
        c += DISTANCE;
      putchar(c);
    }
    else if (c == ' ')
      putchar(c);
  }
  printf("\n");
  return 0;
}
```

Lab 4 - 과제1

- Part 1
  - 전처리기를 이용하여 소문자와 대문자 값의 차이를 표현
  - 판단할 문자 하나를 담을 변수 c
  - 첫 글자 판단을 위해서 첫 글자 여부를 나타내는 first

```c
#include <stdio.h>
#define DISTANCE ('a'-'A')

int main(void)
{
    int c;
    int first = 1;
```

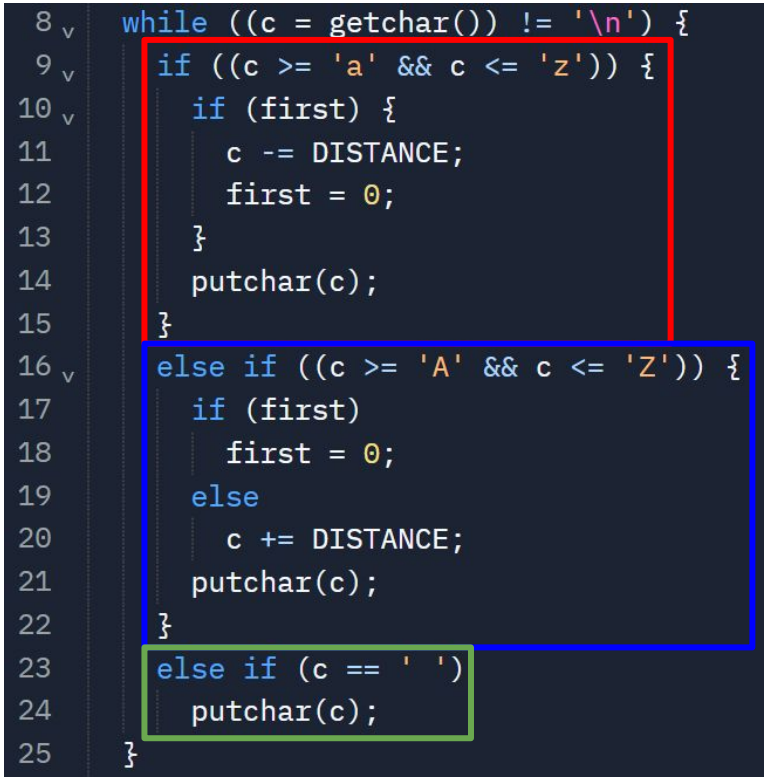
Lab 4 - 과제1

- Part 2
  - 개행문자가 들어올 때까지 반복

```
8   while ((c = getchar()) != '\n') {
9     if ((c >= 'a' && c <= 'z')) {
10      if (first) {
11        c -= DISTANCE;
12        first = 0;
13      }
14      putchar(c);
15    }
16    else if ((c >= 'A' && c <= 'Z')) {
17      if (first)
18        first = 0;
19      else
20        c += DISTANCE;
21      putchar(c);
22    }
23    else if (c == ' ')
24      putchar(c);
25  }
```

Lab 4 - 과제1

- Part 2
  - 3가지의 입력으로 구분
    - 소문자
    - 대문자
    - 띄어쓰기



```
8   while ((c = getchar()) != '\n') {
9     if ((c >= 'a' && c <= 'z')) {
10      if (first) {
11        c -= DISTANCE;
12        first = 0;
13      }
14      putchar(c);
15    }
16    else if ((c >= 'A' && c <= 'Z')) {
17      if (first)
18        first = 0;
19      else
20        c += DISTANCE;
21      putchar(c);
22    }
23    else if (c == ' ')
24      putchar(c);
25  }
```

Lab 4 - 과제1

- Part 2 - 1 (소문자)
  - first 변수를 해당 글자가 첫 번째 알파벳인지 판단
  - 처음이면 대문자로 변환하고 first를 0으로 변경
  - putchar로 출력

```
8   while ((c = getchar()) != '\n') {
9     if ((c >= 'a' && c <= 'z')) {
10      if (first) {
11        c -= DISTANCE;
12        first = 0;
13      }
14      putchar(c);
15    }
16    else if ((c >= 'A' && c <= 'Z')) {
17      if (first)
18        first = 0;
19      else
20        c += DISTANCE;
21      putchar(c);
22    }
23    else if (c == ' ')
24      putchar(c);
25  }
```

Lab 4 - 과제1

- Part 2 - 1 (대문자)
  - 해당 문자가 처음으로 받은 알파벳 인지 판단
  - 처음이면 first를 0으로 변경
  - 처음이 아니라면 소문자로 변환
  - putchar로 출력

```
8   while ((c = getchar()) != '\n') {
9     if ((c >= 'a' && c <= 'z')) {
10      if (first) {
11        c -= DISTANCE;
12        first = 0;
13      }
14      putchar(c);
15    }
16    else if ((c >= 'A' && c <= 'Z')) {
17      if (first)
18        first = 0;
19      else
20        c += DISTANCE;
21      putchar(c);
22    }
23    else if (c == ' ')
24      putchar(c);
25  }
```

Lab 4 - 과제1

- Part 2 - 1 (소문자)
  - 띄어쓰기는 그대로 출력

```
8   while ((c = getchar()) != '\n') {
9     if ((c >= 'a' && c <= 'z')) {
10      if (first) {
11        c -= DISTANCE;
12        first = 0;
13      }
14      putchar(c);
15    }
16    else if ((c >= 'A' && c <= 'Z')) {
17      if (first)
18        first = 0;
19      else
20        c += DISTANCE;
21      putchar(c);
22    }
23    else if (c == ' ')
24      putchar(c);
25  }
```

Lab 4 - 과제1

- Part 3
  - 개행문자 출력 후 마무리

```
26    printf("\n");
27    return 0;
28  }
```

Lab 4 - 과제1

- 학생 답안 1
  - ‘first’를 사용한 경우

```c
#include <stdio.h>

int main(void) {
  int c;
  int check=0;
  while((c=getchar())!=EOF){
    if((c<'A'||(c>'Z'&&c<'a')||c>'z')&&c!='\n'&&c!=' ');
    else if(c>='a'&&c<='z'&&check==0){
      putchar(c-'a'+'A');
      check=1;
    }
    else if(c>='A'&&c<='Z'&&check==0){
      putchar(c);
      check=1;
      }
    else if(c>='A'&&c<='Z'&&check==1){
      putchar(c-'A'+'a');
    }
    else
      putchar(c);

    if(c=='\n')
      break;
    }
  return 0;
}
```

Lab 4 - 과제1

- 학생 답안 2
  - ‘first’를 사용하지 않은 경우

```c
#include <stdio.h>

int main(void) {
char c;
int i = 0;

for (; ; i++) {
  c = getchar();
  if (c >= 'a' && c <= 'z') {
    putchar(c - 'a' + 'A');
    break;
    }
  else if (c >= 'A' && c <= 'Z') {
    putchar(c);
    break;
    }
}

while ((c = getchar()) != '\n') {
  if (c >= 'a' && c <= 'z')
    putchar(c);

  else if (c >= 'A' && c <= 'Z')
    putchar(c - 'A' + 'a');

  else if (c == ' ')
    putchar(c);
}
return 0;
}
```

Lab 4 - 과제1

- 학생 답안 3

```c
#include <stdio.h>

int main(void) {
  char c;
  c = getchar();

  if (c >= 'a' && c <= 'z'){
    c = c - 'a' + 'A';
    putchar(c);
  }
  else
    putchar(c);

  while ((c = getchar()) != '\n'){
    if (c >= 'A' && c <= 'Z'){
      c = c - 'A' + 'a';
      putchar(c);}

    else if (c >= 'a' && c <= 'z' || c == ' ' )
      putchar(c);

    else if (c == '\n'){
      putchar(c);
      break;
      }
  }
  printf("\n");
  return 0;
}
```

Lab 4 과제 1 - 주의점

- 특수 문자로 시작하는 입력
  - 입력: !@#student
  - 출력: Student
- 띄어쓰기
  - 출력 결과의 처음이나 중간에 포함된 불필요한 띄어쓰기 - 오답
  - **정답 출력 후, 마지막에 불필요한 띄어쓰기 출력(다른 과제도 해당)**

Lab 4 과제 2

Lab 4 - 과제2

- **아주 간단한 암호 해독 프로그램**
- **조건 1**
  - 사용자는 다음과 같이 입력한다고 가정 : [자연수] (1 <= n <= 26)
  - 입력 예시) 10
  - 이외의 입력은 예외처리한다.
- **조건 2**
  - 이를 해독하기 위해, 입력된 숫자를 순서에 해당하는 대문자 알파벳으로 변환 후,
  - 오른쪽으로 다섯 칸 SHIFT 한다. (※ A의 옆은 B, B의 옆은 C, …, Z의 오른쪽은 A입니다.)
- **조건 3**
  - 입력된 위치에서 다섯칸 칸 오른쪽으로 SHIFT 한 위치의 해독된 알파벳을 출력
  - 입출력 예시1) 입력 - 10, 출력 - O
  - A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  - 입출력 예시2) 입력 - 25, 출력 - D
  - A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Lab 4 - 과제2

- 전체 코드

```c
#include <stdio.h>
#define SHIFT 5

int main(void)
{
  int index, decoded;
  scanf("%d", &index);

  if (index <= 1 || index >= 26) {
    printf("wrong input\n");
    return 0;
  }

  index = (index - 1 + SHIFT) % 26;
  decoded = 'A' + index;

  printf("%c\n", decoded);

  return 0;
}
```

Lab 4 - 과제2

- 코드 설명
  - 전처리기를 이용하여 해독에 필요한 값 표현
  - 사용자에게 입력을 받을 변수 index
  - 해독된 결과가 저장될 변수 decoded
  - scanf를 이용해서 index를 받아온다

```c
#include <stdio.h>
#define SHIFT 5

int main(void)
{
  int index, decoded;
  scanf("%d", &index);

  if (index <= 1 || index >= 26) {
    printf("wrong input\n");
    return 0;
  }

  index = (index - 1 + SHIFT) % 26;
  decoded = 'A' + index;

  printf("%c\n", decoded);

  return 0;
}
```

Lab 4 - 과제2

- 코드 설명
  - 입력이 원하는 범위에 들어와 있는지 판단하는 부분
  - 잘못된 입력이면 오류를 출력하고 프로그램을 종료

```c
#include <stdio.h>
#define SHIFT 5

int main(void)
{
  int index, decoded;
  scanf("%d", &index);

  if (index <= 1 || index >= 26) {
    printf("wrong input\n");
    return 0;
  }

  index = (index - 1 + SHIFT) % 26;
  decoded = 'A' + index;

  printf("%c\n", decoded);

  return 0;
}
```

Lab 4 - 과제2

- 코드 설명
  - 해독 후, 출력하는 부분
  - ‘A’에서 index - 1 만큼 더하면 해당 index의 대문자가 만들어 짐
    - 1 - ‘A’, 2 - ‘B’, 3 - ‘C’, …
  - 이때, 해독을 위해 5 만큼 index를 증가
  - index가 25를 넘는 경우를 위해 26으로 모듈로 연산 수행
  - ‘A’에 계산된 index를 더해서 결과 출력

```c
#include <stdio.h>
#define SHIFT 5

int main(void)
{
  int index, decoded;
  scanf("%d", &index);

  if (index <= 1 || index >= 26) {
    printf("wrong input\n");
    return 0;
  }

  index = (index - 1 + SHIFT) % 26;
  decoded = 'A' + index;

  printf("%c\n", decoded);

  return 0;
}
```

Lab 4 - 과제2

- 학생 답안 1
  - % 연산을 사용하지 않은 경우

```c
#include <stdio.h>

int main(void) {
  int num;
  scanf("%d", &num);
  if(num>=1 && num<=26){
    num += 5;
    if(num>26) num -= 26;
    char ch = 'A' + num - 1;
    printf("%c", ch);
  }
  printf("\n");
  return 0;
}
```

실습 1

실습 1 - 설명

- 별을 이용하여 다양한 그림을 그려봅시다!
- printf와 '*' 문자를 활용해 봅시다
- 반복문의 활용법에 대해 익숙해질 수 있습니다

실습 1 - 복습

- 먼저, 사용자에게 자연수 하나를 입력 받습니다
- 입력 받은 정수 만큼 별을 가로로 출력해보세요

```
> make -s
> ./main
1
*
> ./main
5
*****
> ./main
20
********************
>
```

실습 1 - 복습

- 코드 설명
  - 입력 받은 값을 저장할 변수를 선언합니다
  - scanf를 이용해서 정수를 받아옵니다
    - 변수 앞에 '&' 가 필요합니다!

```c
#include <stdio.h>

int main(void) {
    int num;

    scanf("%d", &num);

    for(int i = 0; i < num; i++){
        printf("*");
    }
    printf("\n");

    return 0;
}

```

실습 1 - 복습

- 코드 설명
  - 별을 출력하는 부분 입니다
  - 별 1개를 출력하는 문장을, 입력 받은 만큼 반복합니다

```c
#include <stdio.h>

int main(void) {
    int num;

    scanf("%d", &num);

    for(int i = 0; i < num; i++){
        printf("*");
    }
    printf("\n");

    return 0;
}

```

실습 1 - 복습

- 코드 설명
  - 별을 전부 찍으면 개행문자를 출력하며 마무리 합니다

```c
#include <stdio.h>

int main(void) {
    int num;

    scanf("%d", &num);

    for(int i = 0; i < num; i++){
        printf("*");
    }
    printf("\n");

    return 0;
}

```

실습 1

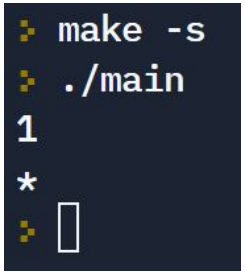
- 마찬가지로 입력 받은 값 만큼 별을 출력해봅시다
- 이번에는 가로가 아닌 세로로 출력이 되도록 만들어 보세요
- 개행문자(엔터)를 언제 넣을지 고민해보세요!

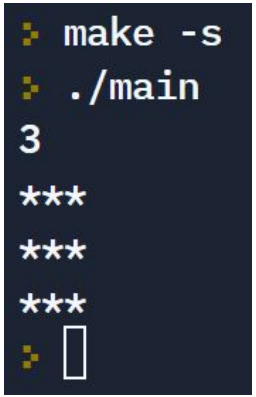
```
> make -s
> ./main
1
*
> ./main
3
*
*
*
> ./main
5
*
*
*
*
*
>
```

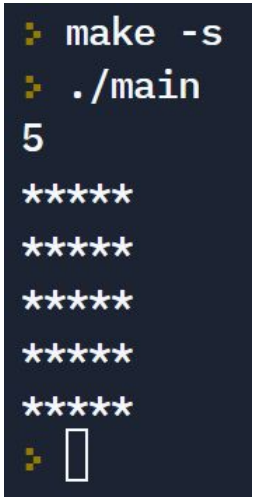
실습 2

실습 2 - 설명

- 이번에는 정사각형을 그려봅시다
- 사용자에게 자연수 하나를 입력받습니다
- 가로, 세로 별의 개수가 입력 받은 값과 맞는 정사각형을 그려봅시다!







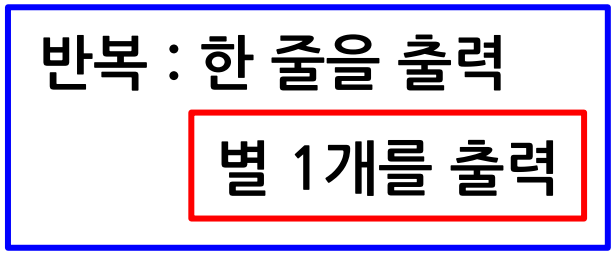
실습 2 - 설명

- 단계 별로 나누어 문제를 해결해봅시다

별 1개를 출력

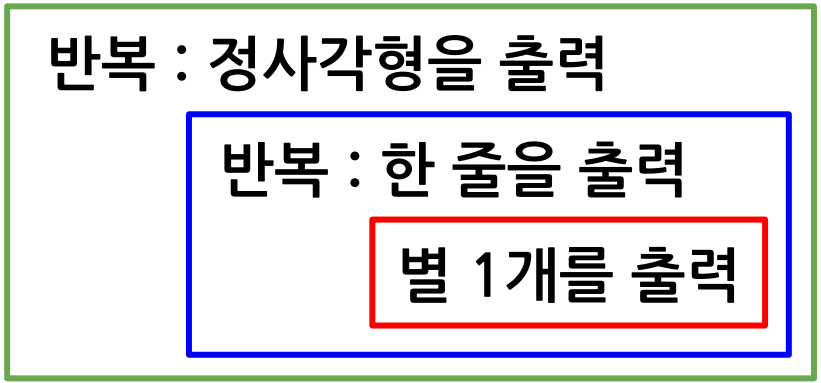
실습 2 - 설명

- 단계 별로 나누어 문제를 해결해봅시다



실습 2 - 설명

- 단계 별로 나누어 문제를 해결해봅시다



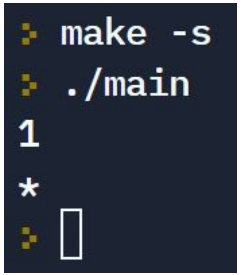
실습 2

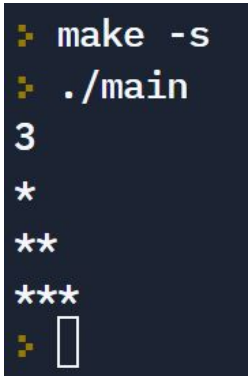
- 이제 정사각형을 그려봅시다
- 이중 반복문(반복문 안의 반복문)이 필요할 수 있습니다
- 마찬가지로 개행문자를 어디에 넣어야 할지가 중요합니다!

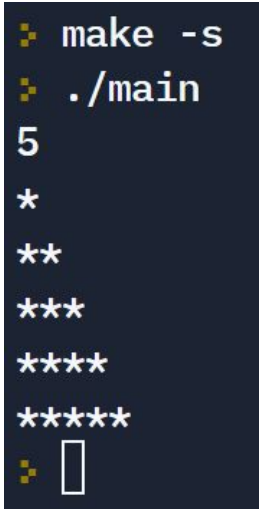
실습 3

실습 3 - 설명

- 이번에는 삼각형을 그려봅시다
- 사용자에게 자연수 하나를 입력받습니다
- 밑변과 높이가 입력 받은 값인 직각삼각형을 그려봅시다







실습 3 - 설명

- 실습 2와 유사한 방법으로 해결할 수 있습니다
- 차이점은 한 줄의 길이가 매번 달라진다는 것입니다

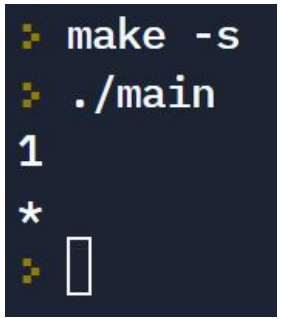
실습 3 - 설명

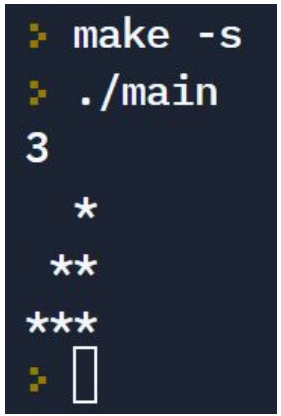
- 코드 설명

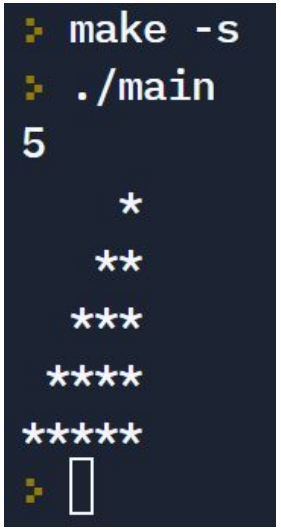
```c
#include <stdio.h>

int main(void) {
    int num;

    scanf("%d", &num);

    for(int i = 0; i < num; i++){
        for(int j = 0; j < i+1; j++){
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```

실습 3 - 제출

- 이번에는 오른쪽으로 정렬된 직각삼각형을 그려봅시다
- 나머지 조건은 앞선 직각삼각형 예제와 동일합니다
- 줄을 출력할 때, 빈 공간이 필요합니다!
  - ‘ ’(띄어쓰기)를 몇 번 출력할지 고민해보세요!



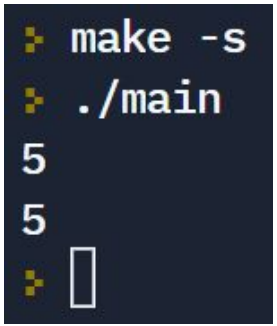


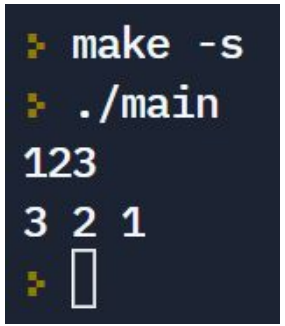


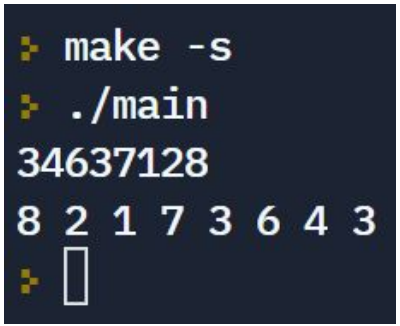
실습 4

실습 4 - 설명

- 양의 정수의 각 자리수를 분해하여 봅시다
- 사용자에게 양의 정수를 하나 입력 받습니다
- 해당 양의 정수의 각 자리수를 분해하여 띄어쓰기와 함께 출력해봅시다
  - 출력은 1의 자리부터 증가하는 순서대로 출력합니다







실습 4 - 설명

- 코드 설명

```c
#include <stdio.h>

int main(void) {
    int num;

    scanf("%d", &num);

    while(num / 10){
        printf("%d ", num % 10);
        num /= 10;
    }
    printf("%d\n", num);

    return 0;
}
```

실습 4 - 제출

- 369 게임을 만들어 봅시다!
- 사용자에게 양의 정수 하나를 입력 받습니다
- 프로그램은 입력 받는 정수 각 자리에 들어있는 3,6,9 숫자의 수를 셉니다
- 3,6,9 가 들어있는 만큼 박수를 출력합니다(짝!)

```
> make -s
> ./main
369
짝!짝!짝!
>
```

```
> make -s
> ./main
1242169
짝!짝!
>
```

```
> make -s
> ./main
999999
짝!짝!짝!짝!짝!짝!
>
```

실습 5

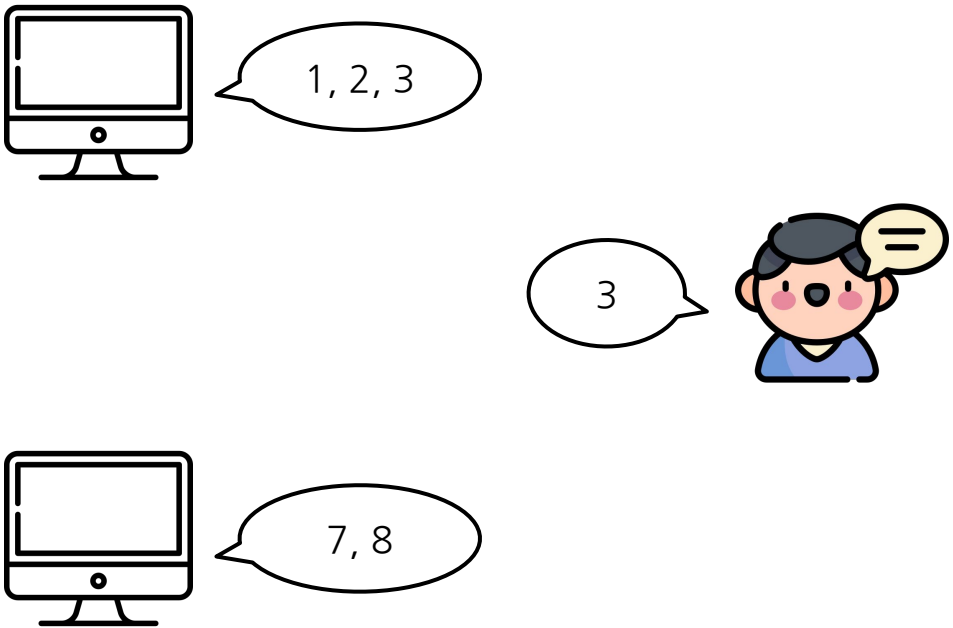
실습 5 - 설명

- 베스킨라빈스31 게임을 만들어볼까요?
- 컴퓨터와 사람이 번갈아 가면서 숫자를 말합니다
- 숫자는 1부터 31까지 순서대로 말할 수 있습니다
- 한 차례에 1개에서 3개 사이의 수를 말할 수 있습니다
- 31을 말하는 차례가 패배하게 됩니다

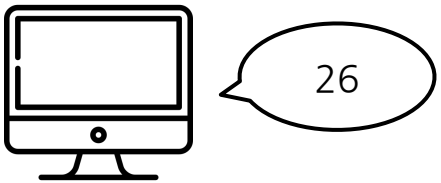
실습 5 - 설명

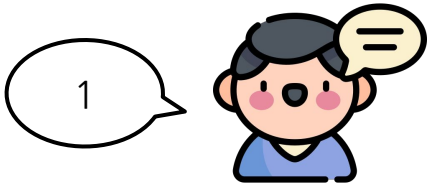
- 컴퓨터가 먼저 숫자를 말합니다
- 사람의 차례에서는 사람이 부를 숫자의 개수를 입력 받습니다
  - 이전 차례의 컴퓨터가 1, 2, 3을 출력한 다음, 사람이 4, 5를 말하고 싶으면 2를 입력합니다
- 컴퓨터가 부를 숫자의 개수는 매 차례마다 랜덤으로 정합니다
- 컴퓨터가 30을 부를 수 있다면, 반드시 30까지 불러서 승리합니다

실습 5 - 설명

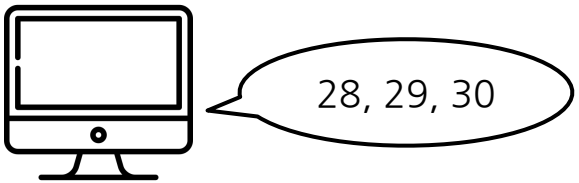


실습 5 - 설명





승리!



실습 5 - 설명

- 간단하게 하기 위해서 다음의 조건을 추가합니다
  - 사람은 반드시 1, 2, 3만을 입력합니다
- 게임이 끝나면 누가 이겼는지 출력합니다
  - Computer wins! / Player wins!

실습 5 - 설명

- 결과 예시

```
> make -s
> ./main
Computer: 1
3
Player: 2 3 4
Computer: 5
1
Player: 6
Computer: 7 8
2
Player: 9 10
Computer: 11 12 13
1
Player: 14
Computer: 15 16
2
Player: 17 18
Computer: 19 20 21
1
Player: 22
Computer: 23 24 25
1
Player: 26
Computer: 27
3
Player: 28 29 30
Computer: 31
Player wins!!
>
```

```
> make -s
> ./main
Computer: 1
3
Player: 2 3 4
Computer: 5
3
Player: 6 7 8
Computer: 9 10 11
3
Player: 12 13 14
Computer: 15 16
3
Player: 17 18 19
Computer: 20 21 22
3
Player: 23 24 25
Computer: 26
3
Player: 27 28 29
Computer: 30
1
Player: 31
Computer wins!!
>
```

실습 5 - 복습

- 랜덤 숫자 생성

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {
    int num;

    srand(time(NULL));

    num = rand() % 3;
    printf("%d\n", num);

    return 0;
}
```

실습 5

- 한 단계씩 같이 만들어 봅시다
- 본인 만의 방법으로 구현해보셔도 좋습니다

실습 5

- 코드 예시

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define COM 0
#define PLY 1

int main(void)
{
  int turn = COM;
  int num, curr = 0;

  srand(time(NULL));

  while(curr < 31){
    if(turn == COM){
      printf("Computer:");

      if(curr < 27)
        num = rand() % 3 + 1;
      else if(curr < 30)
        num = 30 - curr;
      else
        num = 1;

      turn = PLY;
    }
    else{
      scanf("%d", &num);

      printf("Player:");

      turn = COM;
    }
    for(int i = 0; i < num && curr < 31; i++){
      printf(" %d", ++curr);
    }
    printf("\n");
  }

  if(turn == COM)
    printf("Computer wins!!\n");
  else
    printf("Player wins!!\n");

  return 0;
}
```

실습 5 - Step 1

- 먼저, 게임을 지속할 조건을 만들어 봅시다
- 현재까지 불린 숫자가 31보다 작을 동안 게임을 지속합니다

```c
#include <stdio.h>

int main(void){
    int curr = 0;

    while(curr < 31){
        //do game
    }

    return 0;
}
```

실습 5 - Step 2

- 게임이 진행되기 위해서는 턴(차례)이 필요합니다
- 턴을 주고받는 로직을 추가해줍니다

```c
#include <stdio.h>

#define COM 0
#define PLY 1

int main(void){
    int curr = 0;
    int turn = COM;

    while(curr < 31){//do game
        if(turn == COM){
            turn = PLY;
        }
        else{
            turn = COM;
        }
    }
```

실습 5 - Step 3

- 각 차례에 맞춰서 각자 부를 숫자의 개수를 결정해야합니다
- 컴퓨터 차례는 3가지 방법으로 개수를 정합니다
  - 현재까지 불린 수가 1~26일 때는 랜덤으로 결정
  - 27~29일 때는 반드시 30까지 부름
  - 상대가 30을 불렀다면, 1개(31)만 부름

```c
#include <stdio.h>

#define COM 0
#define PLY 1

int main(void){
    int curr = 0;
    int turn = COM;

    while(curr < 31){//do game
        if(turn == COM){
            turn = PLY;
        }
        else{
            turn = COM;
        }
    }
```

실습 5 - Step 3

- 랜덤을 사용하기 위한 라이브러리를 추가합니다
- 부를 숫자의 개수를 저장할 변수를 만듭니다
- 앞서 나눈 경우에 따라서 부를 숫자의 개수를 결정합니다

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define COM 0
#define PLY 1

int main(void){
  int curr = 0;
  int num, turn = COM;

  srand(time(NULL));

  while(curr < 31){//do game
    if(turn == COM){
      if(curr < 27)
        num = rand() % 3 + 1;
      else if(curr < 30)
        num = 30 - curr;
      else
        num = 1;
      turn = PLY;
    }
    else{
```

실습 5 - Step 4

- 사람의 차례에는 입력을 받아 개수를 결정합니다

```
19            num = 30 - curr;
20          else
21            num = 1;
22          turn = PLY;
23        }
24        else{
25          scanf("%d", &num);
26
27          turn = COM;
28        }
29      }
```

실습 5 - Step 5

- 수를 출력하는 부분을 추가합니다
- 결정된 수 만큼, 그리고 31을 넘지 않을 때까지 반복합니다
- 현재까지 불렸던 숫자(curr)를 하나씩 증가시키면서 출력합니다

```
14  while(curr < 31){//do game
15    if(turn == COM){
16      if(curr < 27)
17        num = rand() % 3 + 1;
18      else if(curr < 30)
19        num = 30 - curr;
20      else
21        num = 1;
22      turn = PLY;
23    }
24    else{
25      scanf("%d", &num);
26
27      turn = COM;
28    }
29    for(int i = 0; i < num && curr < 31; i++){
30      printf(" %d", ++curr);
31    }
32    printf("\n");
33  }
```

실습 5 - Step 6

- 게임의 승패 판단하는 부분을 추가합니다
- while문 밖에 작성해야합니다
  - while문이 끝났다는 것이 게임의 승패가 결정되었다는 뜻이 됩니다
- 턴을 넘겨주고 게임이 끝나기 때문에, 31은 현재 턴의 상대방이 불렀습니다
- 따라서, 현재 턴이 게임의 승리자입니다

```
35    if(turn == COM)
36      printf("Computer wins!!\n");
37    else
38      printf("Player wins!!\n");
39
40    return 0;
41  }
```

실습 5

- 중간 중간 printf문을 추가해 게임을 완성해보세요

실습 5 - Optional

- 둘이서 하는 베스킨라빈스 31 게임은 먼저 부른 사람이 반드시 이기는 방법이 존재합니다
- 컴퓨터가 반드시 이길 수 있는 알고리즘을 생각해보세요!

과제 1

과제1

- 반쪽짜리 크리스마스 트리를 만들어봅시다
- 반쪽짜리 크리스마스 트리는 다음과 같은 조건을 만족합니다
  - 트리의 모양은 '*' 문자를 이용하여 표현합니다
  - 왼쪽으로 정렬된 직각삼각형이 순차대로 쌓여있는 모습입니다
    - 1 층은 '*'이 1개
    - 2 층은 삼각형의 가로, 높이가 2인 직각삼각형
    - 3 층은 삼각형의 가로, 높이가 3인 직각삼각형
    - …

과제1

- 출력 예

```
> make -s
> ./main
1
*
>
```

```
> make -s
> ./main
3
*
*
**
*
**
***
>
```

```
> make -s
> ./main
5
*
*
**
*
**
***
*
**
***
****
*
**
***
****
*****
>
```

과제1

- 다음 조건을 만족할 수 있도록 프로그래밍 해보세요
- 사용자에게 트리 층의 개수를 입력 받습니다
- 사용자는 올바른 값만 입력한다고 가정합니다
  - 1 이상의 자연수
  - 입력 범위: 1 ~ 10
- 입력 받은 만큼 층을 갖는 반쪽짜리 크리스마스 트리를 그려보세요!

과제1 - 평가 기준

- 기준 1
  - 프로그램이 동작하는가? - 1점
- 기준 2
  - 프로그램이 정상적으로 입력을 받는가? - 1점
- 기준 3
  - 기준에 부합하여 결과를 출력하는가? - 3점

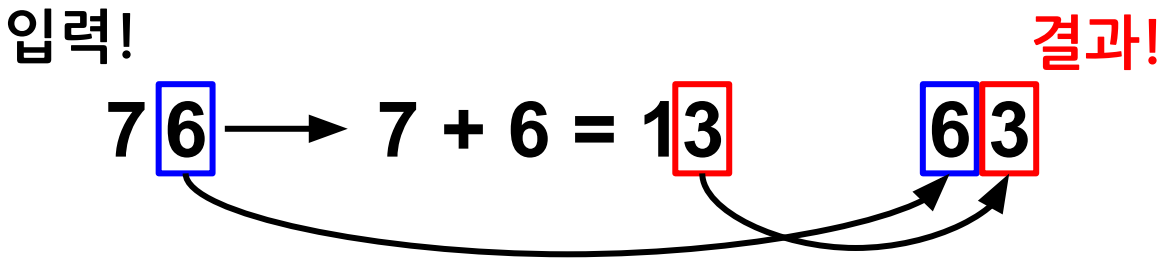
과제 2

과제2

- 다음 문제를 해결할 수 있는 프로그램을 만들어보세요!
- 사용자로부터 0~99 사이의 정수를 하나 입력 받습니다
  - 사용자는 항상 옳은 값만 입력한다고 가정합니다
- 주어진 규칙을 통해서 입력 받은 정수를 계산합니다
- 계산된 숫자가 원래의 입력 받은 숫자로 돌아오는데 걸린 사이클의 수를 구해보세요

과제2

- 주어진 수의 십의 자리와 일의 자리를 더합니다
  - 주어진 수가 한 자리 수 라면, 10을 곱해서 2자리로 만든 후 계산합니다
  - 0을 입력 받았다면, 10을 곱해도 0이므로 0과 0을 더하는 과정이 됩니다
- 주어진 수의 1의 자리와, 더해서 나온 수의 1의 자리를 조합합니다



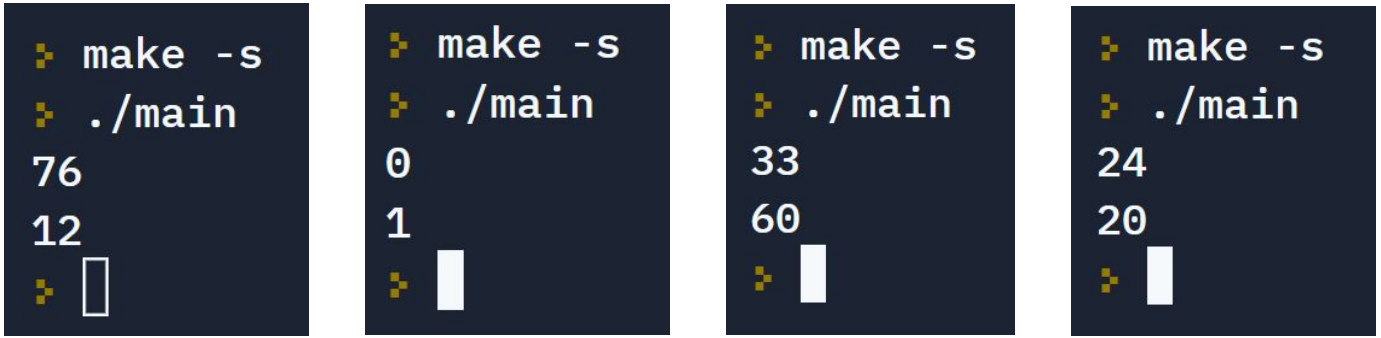
과제2

- 계산 과정
  - 이해를 위한 출력 결과입니다
  - 과제는 총 사이클 수만 출력하시면 됩니다

```
> make -s
> ./main
76
Cycle  1: 63
Cycle  2: 39
Cycle  3: 92
Cycle  4: 21
Cycle  5: 13
Cycle  6: 34
Cycle  7: 47
Cycle  8: 71
Cycle  9: 18
Cycle 10: 89
Cycle 11: 97
Cycle 12: 76
12
>
```

과제2

- 새롭게 만들어진 숫자를 이용하여 같은 계산을 반복합니다
- 처음에 입력 받은 숫자가 다시 나올 때 까지, 이 계산을 몇 번 반복해야하는지 알아내는 프로그램을 작성해보세요



과제2 - 평가 기준

- 기준 1
  - 프로그램이 동작하는가? - 1점
- 기준 2
  - 프로그램이 정상적으로 입력을 받는가? - 1점
- 기준 3
  - 기준에 부합하여 결과를 출력하는가? - 3점