4190.103A-001: Programming Practice Lab. 6

Jaewon Choi
Keunsan Park
Haeun Lee
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

Seoul National University

Spring 2023

Lab. 5 실습 풀이

실습 1

사용자로부터 두 수 n, m을 입력받습니다. 그리고 2개의 독립된 길이 n, m짜리 문자열을 각각 scanf 함수를 통해 입력받을 수 있는 1차원 배열을 선언합니다. 1차원 배열에 연속적으로 두 문자열을 입력 받고, 출력합니다. 이후 rand 함수를 이용하여 arr[rand() % (n + m)]를 출력합니다. 배열에 문자열을 연속하여 입력받을 때에는 중간에 \0 문자가 삭제되어 한 번에 printf에서 %s를 이용하여 출력할 수 있도록 하세요. (rand 함수 seed 설정하지 말 것)

(문자열의 길이는 1 이상)

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

int main(void) {
    int n, m, idx = 0;
    scanf("%d %d", &n, &m);
    char input[n + m + 1];

    scanf("%s", input);
    scanf("%s", &(input[n]));

    printf("%s\n", input);
    printf("%c", input[rand() % (n + m)]);

    return 0;
}
```

실습 2

사용자로부터 숫자 n을 입력받고, 함수를 사용하여 1에서 n까지의 숫자 중에서 소수가 몇 개인지와 그들의 합을 출력하는 프로그램을 작성하세요.

```c
int main(void) {
    int n, cnt = 0, sum = 0;

    scanf("%d", &n);

    for(int i = 1; i <= n; i++)
    {
        if(is_prime(i) == 1)
        {
            cnt++;
            sum += i;
        }
    }

    printf("%d\n%d\n", cnt, sum);

    return 0;
}
```

```c
int is_prime(int number) {
    int isPrime = 1;

    if (number == 1)
        isPrime = 0;
    else if (number <= 3)
        isPrime = 1;
    else
    {
        for (int i = 2; i <= number / 2; i++)
        {
            if (number % i == 0)
            {
                isPrime = 0;
                break;
            }
        }
    }

    return isPrime;
}
```

실습 3

사용자로부터 두 정수 n, m을 입력받으시오. 배열을 생성하여 길이 n인 문자열을 입력받은 후, m 번만큼 rand 함수를 호출하여 n으로 나눈 나머지를 구합니다. 이 때 이 값을 k라고 할 때, k번째 칸부터 시작하는 문자열을 출력하시오. 단, rand함수의 seed를 설정하지 말 것.

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

int main(void)
{
    int n, m;
    scanf("%d %d", &n, &m);
    char str[n + 1];

    scanf("%s", str);

    for(int i = 0; i < m; i++)
    {
        printf("%s\n", &(str[rand() % n]));
    }

    return 0;
}
```

실습 4

사용자로부터 정수 n을 입력 받습니다. 길이 10 이하의 n개의 문자열을 입력받고, 각 문자열의 길이를 계산하여 출력합니다. 문자열을 입력받고, 문자열의 길이를 세는 부분을 별도의 함수로 정의하세요. (문자열의 길이는 양수)

```c
#include <stdio.h>

void str_len(void)
{
    char str[11];

    scanf("%s", str);

    for(int i = 0; i <= 10; i++)
    {
        if(str[i] == '\0')
        {
            printf("%d\n", i);
            break;
        }
    }

    return;
}

int main(void) {
    int n;

    scanf("%d", &n);

    for (int i = 0; i < n; i++)
    {
        str_len();
    }

    return 0;
}
```

or

```c
#include <stdio.h>

int str_len(char* str)
{
    int i;

    for(i = 0; i <= 10; i++)
    {
        if(str[i] == '\0')
            break;
    }

    return i;
}

int main(void) {
    int n;
    char str[11];

    scanf("%d", &n);

    for (int i = 0; i < n; i++)
    {
        scanf("%s", str);
        printf("%d\n", str_len(str));
    }

    return 0;
}
```

Lab. 5 과제 풀이

과제 1

사용자에게 정수 n을 입력 받습니다. char 형 배열을 선언하고, random 함수를 이용하여 총 n개의 문자를 입력받습니다. (random 함수에서 얻어진 값을 128으로 나눈 나머지로 배열을 채웁니다. random 함수에 seed를 설정하지 말 것.)

해당 배열에서 printf 함수(%s 이용)를 사용하여 출력 가능한 문자열이 몇 개 있는지 출력하시오.

e.g.
"abc\0def\0"에서 출력 가능한 문자열은 2개. 따라서 2를 출력하시면 됩니다.

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

int main(void) {
    int n, cnt = 0;
    scanf("%d", &n);
    char input[n + 1];

    for(int i = 0; i < n; i++)
    {
        input[i] = rand() % 128;
    }

    for(int i = 0; i < n; i++)
    {
        if (input[i] == '\0')
            cnt++;
    }

    if (n > 0 && input[n-1] != '\0')
        cnt++;

    printf("%d", cnt);

    return 0;
}
```

or

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

int main(void) {
    int n, cnt = 0, input = '\0';

    scanf("%d", &n);

    for(int i = 0; i < n; i ++)
    {
        input = rand() % 128;
        if (input == '\0')
            cnt++;
    }

    if (input != '\0')
        cnt++;

    printf("%d", cnt);

    return 0;
}
```

과제 2

사용자에게 정수 n을 입력받으세요. 함수를 정의하여 n번 만큼 해당 함수의 반환 값을 출력합니다. 함수는 매번 입력 받은 실수 m에 대해 m보다 작거나 같은 가장 큰 2의 제곱 수를 반환합니다.

```c
#include <stdio.h>

int getResult(int m)
{
    int ret = 1;

    while(ret <= m)
    {
        ret *= 2;
    }

    return ret / 2;
}

int main(void) {
    int n, m;

    scanf("%d", &n);

    for(int i = 0; i < n; i ++)
    {
        scanf("%d", &m);
        printf("%d\n", getResult(m));
    }

    return 0;
}
```

printf - 디버깅

printf - 디버깅

- printf 함수를 활용하여 디버깅을 수행 할 수 있습니다.
  - 작성한 코드의 세부 동작 과정을 눈으로 확인 할 수 있습니다.
  - 주요 분기점들이 의도대로 수행되고있는지 확인 할 수 있습니다.
- 지난주 (5주차) 실습문제를 통해 예시를 확인 해 봅시다.
  - 5주차 - 실습 2

> 사용자로부터 숫자 n을 입력받고, 함수를 사용하여 1에서 n까지의 숫자 중에서 소수가 몇 개인지와 그들의 합을 출력하는 프로그램을 작성하세요.

printf - 디버깅

```c
int main(void) {
    int n, cnt = 0, sum = 0;

    scanf("%d", &n);

    for(int i = 1; i <= n; i++)
    {
        if(is_prime(i) == 1)
        {
            cnt++;
            sum += i;
        }
    }

    printf("%d\n%d\n", cnt, sum);

    return 0;
}
```

```c
int is_prime(int number) {
    int isPrime = 1;

    if (number == 1)
        isPrime = 0;
    else if (number <= 3)
        isPrime = 1;
    else
    {
        for (int i = 2; i < number / 2; i++)
        {
            if (number % i == 0)
            {
                isPrime = 0;
                break;
            }
        }
    }

    return isPrime;
}
```

```
/* 코드가 실행되는 중입니다... */
채점을 시작합니다...
================================
Case 1. 테스트를 통과했습니다! (+20)
Case 2. 테스트를 통과했습니다! (+20)
Case 3. 정답이 틀렸습니다. (+0)
Case 4. 정답이 틀렸습니다. (+0)
Case 5. 정답이 틀렸습니다. (+0)
================================
채점을 마쳤습니다.
총 점수: 40 / 100

/* 코드 실행이 완료되었습니다! */
```

printf - 디버깅

1. 오답이 나오는 입력을 찾아봅니다.
2. 결과 값을 계산하는곳, 또는
   결과값을 위한 중요 조건문 등,
   주요 분기점에서 계산에 사용되고있는
   값들을 출력해 봅니다.
3. 오류가 발생하는 case를 확인합니다.

```
/* 코드가 실행되는 중입니다... */
7
5
21

/* 코드 실행이 완료되었습니다! */
```

```c
int main(void) {
    int n, cnt = 0, sum = 0;

    scanf("%d", &n);

    for(int i = 1; i <= n; i++)
    {
        if(is_prime(i) == 1)
        {
            cnt++;
            sum += i;
            printf("%d is prime, cnt: %d\n", i, cnt);
        }
    }

    printf("%d\n%d\n", cnt, sum);

    return 0;
}
```

```
/* 코드가 실행되는 중입니다... */
7
2 is prime, cnt: 1
3 is prime, cnt: 2
4 is prime, cnt: 3
5 is prime, cnt: 4
7 is prime, cnt: 5
5
21

/* 코드 실행이 완료되었습니다! */
```

printf - 디버깅

4. 세부 동작 부분에 printf를 추가해봅니다.
5. 결과를 분석합니다.
   a. 계산 오류를 확인하기 위해 추가한 printf 문이 출력 안된 것을 확인
   b. “number ==4” 일 때, else 문 안에서 for문이 실행 될 것이라 예상한 것이 잘못됨을 확인
   c. for문 조건 식을 확인 하고, “number == 4” 일 때 오작동 함을 확인

```c
int is_prime(int number) {
    int isPrime = 1;

    if (number == 1)
        isPrime = 0;
    else if (number <= 3)
        isPrime = 1;
    else
    {
        for (int i = 2; i < number / 2; i++)
        {
            if(number == 4)
            {
                printf("i: %d, number: %d, %% = %d\n",
                        i, number, i % number);
            }
            if (number % i == 0)
            {
                isPrime = 0;
                break;
            }
        }
    }

    return isPrime;
}
```

```
/* 코드가 실행되는 중입니다... */
7
2 is prime, cnt: 1
3 is prime, cnt: 2
4 is prime, cnt: 3
5 is prime, cnt: 4
7 is prime, cnt: 5
5
21

/* 코드 실행이 완료되었습니다! */
```

printf - 디버깅

- [5주차 - 과제 1]의 경우, 입력이 rand() 로 주어져서 입력과 출력을 분석하는데 어려움이 있습니다.

  1. printf 로 rand() 결과를 출력 해 볼 수 있습니다.

  2. 입력을 조작하여 넣고 결과를 확인 할 수 있습니다.

```
for (int i = 0; i < n; i++)
{
    input[i] = rand() % 128;
    if(i % 20 == 0) printf("\n");
    printf("% 4d", input[i]);
}
```

```
/* 코드가 실행되는 중입니다... */
200

 103  70 105 115  81 127  74 108  41  77  58  43 114 123  99  70 124  66  84 120
  27 104 103  13 118  90  46  99  51  31  73  26 102  50  13  55  49  88  35  90
  37  93   5  23  88 105  94  84  43  50  77  70  27  52  84  17  14   2 116  65
  33  61  92   7 112 105  62  33  65  97 124 103  62   1 126  23 106  92 107  22
  15  56  92  42 108  48  59 123  50  47  60  84 108  24  91  92   2  26 126  67
 123 122  42  58 123  41  81 102   5  60 124  20 117  88  62  97   9 121  92  59
  40  25  15  21  49 107 113  51   5 111 119   0 105  33  58 101  74  11  75  80
  72  71 100  61  31  35  30  40  28 123 100  69  20 115  90  69  94  75 121  99
  59 112 100  36  17  30   9  92  42  84  44 114  27  16  47  59  51  77  99  80
  72  71  21  92  59 111  34  25  58  27 125 117  11  97  26  28 127  35 120  41
2

/* 코드 실행이 완료되었습니다! */
```

```
// for (int i = 0; i < n; i++)
// {
//     input[i] = rand() % 128;
// }
input[0] = 'a';
input[1] = 'b';
input[2] = 'c';
input[3] = '\0';
input[4] = 'd';
input[5] = 'e';
input[6] = 'f';
input[7] = '\0';
```

```
/* 코드가 실행되는 중입니다... */
8
2

/* 코드 실행이 완료되었습니다! */
```

printf - 서식지정자

printf - 서식지정자

- printf 함수의 서식지정자 형식: “%[플래그][폭][.정밀도][길이]지정자”
- 예시: “%06lld” -> “%[0][6][][ll]d” , “%08.4f” -> “[0][8][.4][]f”
- 플래그: 출력에 대한 플래그 설정
  - 0: 출력의 왼쪽을 0으로 채움
  - +: 부호 출력을 강제
- 폭: 최소 출력길이를 설정
- .정밀도: 소수점 아래 출력길이를 설정
- 길이: data type의 길이를 지정 (h: short, l: long)

```c
#include <stdio.h>

int main(void)
{
    long long int a = 38;
    float b = 6.136;

    printf("%06lld\n", a);
    printf("%08.4f\n", b);

    return 0;
}
```

```
/* 코드가 실행되는 중입니다... */
000038
006.1360

/* 코드 실행이 완료되었습니다! */
```

math library

math library

- “math.h” 헤더파일을 추가하여 수학적 연산을 위한 함수들을 사용 가능합니다.
  - 삼각함수
    - sin(), cos(), tan() …
  - 지수/로그 함수
    - exp(), log(), log10() …
  - C 표준라이브러리 문서에서 더 자세한 내용들을 볼 수 있습니다.

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
    double a = 4;
    double b = 3;

    printf("%f\n", pow(a, b));
    printf("%f\n", sqrt(a));

    return 0;
}
```

```
/* 코드가 실행되는 중입니다... */
64.000000
2.000000

/* 코드 실행이 완료되었습니다! */
```

```c
#include <stdio.h>
#include <math.h>

#define PI 3.14159265

int main ()
{
  double param = 30;

  // 1 radian = 180 / PI degrees
  printf ("%.2f\n", sin(param * PI / 180));

  return 0;
}
```

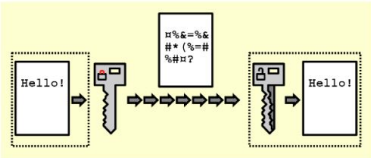
```
/* 코드가 실행되는 중입니다... */
0.50

/* 코드 실행이 완료되었습니다! */
```

함수

함수

- 함수를 사용하여 코드를 기능 별로 분리할 수 있습니다.
- 반복 되는 기능들을 위한 코드를 재사용하기 좋고,
  코드의 가독성을 높일 수 있습니다.
- 기존에 풀었던 문제를 함수를 이용하여 해결 할 수 있습니다.
  - 4주차 - 과제1 을 함수를 사용하여 풀어봅시다.

Q. 암호화(Encryption) 프로그램을 만들어보자.



`getchar()` 를 이용해 `\n` 전까지 문자를 입력받고, 입력된 문자들을 아래 조건에 따라 암호화 해보자.

조건

- 첫번째 입력받은 문자를 key로 사용한다. 즉, `"ABC"` 를 입력했을 경우 `'A'` 가 key이며, `"ABC"` 가 plaintext가 된다.
- key값을 2진수로 표현했을 때 1인 자릿수 갯수를 N이라고 할 때, plaintext에 해당하는 모든 문자들을 N만큼 left shift한다.

  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

- 예를 들어, `'A'` 가 3만큼 left shift되면 `'X'` 가 되며, `'X'` 가 2만큼 left shift되면 `'V'` 가 된다.
- 최소 하나 이상의 알파벳 대문자만 입력된다고 가정한다.

함수

```c
#include<stdio.h>

int main(){
    char c;
    int key, i = 0, shift = 0;

    while((c = getchar()) != '\n')
    {
        if (i++ == 0)
        {
            key = c;

            while(key > 0)
            {
                shift += key % 2;
                key /= 2;
            }
        }

        c = c - shift;
        if (c < 'A')
            c += 'Z' - 'A' + 1;

        putchar(c);
    }

    return 0;
}
```

```c
int main(){
    char c;
    int shift, i = 0;

    while((c = getchar()) != '\n')
    {
        if (i++ == 0)
        {
            shift = getShiftNum(c);
        }

        printEncryptedChar(c, shift);
    }

    return 0;
}
```

```c
int getShiftNum(int key)
{
    int shift = 0;

    while(key > 0)
    {
        shift += key % 2;
        key /= 2;
    }

    return shift;
}

void printEncryptedChar(char c, int shift)
{
    c = c - shift;
    if (c < 'A')
        c += 'Z' - 'A' + 1;

    putchar(c);
}
```

재귀 함수

재귀 함수

- 특정 함수가 내부에서 자신을 호출하는 것을 재귀 호출이라 하고,
  재귀 호출이 존재하는 함수를 재귀 함수라고 합니다.
- 이전에 풀어보았던 factorial 문제를 재귀함수를 통해 풀어봅시다.

n!(Factorial) 계산하는 프로그램을 만들어보세요

- 양의 정수 n(n <= 12)을 입력받습니다
- n! = (n) x (n-1) x (n-2) .... x (1)

```c
int main()
{
    int n;
    int result = 1;

    scanf("%d",&n);

    for(int i = n; i > 0; i--)
    {
        result = result * i;
    }

    printf("%d\n", result);

    return 0;
}
```

```c
int factorial(int n)
{
    if (n == 1)
        return n;
    else
        return n * factorial(n-1);
}

int main()
{
    int n;

    scanf("%d",&n);

    printf("%d\n", factorial(n));

    return 0;
}
```

재귀 함수

- 함수내부에서 재귀호출을 2회 이상 수행할 수 있습니다.
- 대표적인 재귀함수 예제인 피보나치 수열을 통해 확인 해 봅시다.

피보나치 수열의 n번 째 값을 계산해봅시다.

- 피보나치 수열은 아래와 같습니다.
  - f(0) = 0
  - f(1) = 1
  - f(n) = f(n-1) + f(n-2) (for n >= 2)
  - 0 1 1 2 3 5 8 13 ...

```c
int fibonacci(int n)
{
    if (n <= 1)
        return n;
    else
        return fibonacci(n-1) + fibonacci(n-2);
}

int main(void)
{
    int n;

    scanf("%d", &n);

    printf("%d", fibonacci(n));

    return 0;
}
```

실습 & 과제

실습 / 과제

- 과제 내용
  - 당일 진행한 실습과 관련된 문제
- 제출 기한
  - 실습 : 당일 자정까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
  - 과제 : 다음 실습 수업 전날 23:59 까지
- 제출 방법
  - Elice의 Submit 기능 활용