



# 기초 프로그래밍 실습

## 11월 2주차



---

학번 : 2016110056

학과 : 불교학부

이름 : 박승원

날짜 : 2016년 11월 11일

---

## 9장 실습 문제 1번, 2번

- 1. 덧셈, 뺄셈, 곱셈, 나눗셈을 지원하는 계산기 프로그램을 작성하여 보자. 이번에는 각 연산들이 몇 번씩 계산되었는지를 기억하게 하자. 각 연산을 지원하는 함수들은 자신이 호출된 횟수를 화면에 출력한다.

- 2. 주사위를 던져서 각각의 면이 몇 번 나왔는지를 출력하는 프로그램을 작성하라. 주사위의 면은 난수를 이용하여 생성한다. 주사위를 던지는 함수 `get_dice_face()`를 만들고 이 함수 안에서 각각의 면이 나올 때마다 그 횟수를 정적 지역 변수를 이용하여 기억하게 하라. `get_dice_face()` 호출 횟수가 100의 배수일 때

```
#include<stdio.h>
#include<ctype.h>

int add(int a, int b) {
    static int count = 0;
    count++;
    printf(" 덧셈은 총 %d번 호출되었습니다 .\n", count);
    printf(" 연산 결과 : %d\n", a+b);
    return a+b;
}

int sub(int a, int b) {
    static int count = 0;
    count++;
    printf(" 뺄셈은 총 %d번 호출되었습니다 .\n", count);
    printf(" 연산 결과 : %d\n", a-b);
    return a-b;
}

int mul(int a, int b) {
    static int count = 0;
    count++;
    printf(" 곱셈은 총 %d번 호출되었습니다 .\n", count);
    printf(" 연산 결과 : %d\n", a*b);
    return a*b;
}

int div(int a, int b) {
```

```

static int count = 0;
count++;
printf(" 나눗셈은 총 %d번 호출되었습니다.\n", count);
printf(" 연산 결과 : %d\n", a/b);
return a/b;
}

int main()
{
    int a, b;
    char c;
    int end = 0;
    while(!end) {
        printf(" 연산을 입력하시오 .");
        scanf("%d%c%d", &a, &c, &b);
        switch(c) {
            case '+': add(a, b); break;
            case '-': sub(a, b);break;
            case '*': mul(a, b);break;
            case '/': div(a, b);break;
            default: end=1;
        }
    }
}

```

----- 문제 1번 실행을 시작합니다. -----  
./1.x  
연산을 입력하시오 .3\*4  
곱셈은 총 1번 호출되었습니다.  
연산 결과 : 12  
연산을 입력하시오 .3\*2  
곱셈은 총 2번 호출되었습니다.  
연산 결과 : 6  
연산을 입력하시오 .2+3  
덧셈은 총 1번 호출되었습니다.  
연산 결과 : 5  
연산을 입력하시오 .2+7  
덧셈은 총 2번 호출되었습니다.  
연산 결과 : 9  
연산을 입력하시오 .30\*1  
곱셈은 총 3번 호출되었습니다.  
연산 결과 : 30  
연산을 입력하시오 .1:1  
----- 문제 1번 실행을 종료합니다. -----

```

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

```

```
int dice [7];

int d_rand() {
    static int count = 0;
    count++;
    dice [rand() % 6 + 1]++;
    return count;
}

int main()
{
    srand(time(NULL));

    while(1) {
        int k = d_rand();
        if(k%100 == 0) {
            for(int i=1; i<7; i++) printf ("%d ", dice [i]);
            printf ("\n");
        }
        if(k == 5000) break;
    }
}
```

```
zezeon@ubuntuZ: ~/Programming/basicProgramming
evince report.pdf
zezeon@ubuntuZ:~/Programming/basicProgramming$ ./2.x
8 19 16 18 20 19
36 33 23 32 49 27
53 45 43 53 62 44
71 56 56 75 78 64
90 72 80 84 95 79
108 84 95 102 116 95
130 104 113 117 130 106
149 123 126 134 150 118
167 136 147 155 162 133
187 155 167 176 168 147
200 168 184 197 180 171
223 181 201 211 197 187
241 203 219 222 207 208
254 220 234 239 223 230
276 237 250 252 240 245
295 251 270 267 258 259
310 268 283 283 281 275
324 283 295 304 297 297
340 293 308 324 319 316
350 312 322 347 342 327
362 332 337 369 354 346
377 351 352 386 374 360
391 367 368 402 393 379
408 378 388 420 405 401
426 392 407 439 420 416
441 413 423 454 437 432
459 431 435 472 450 453
484 446 451 485 465 469
510 460 465 500 481 484
523 476 479 515 503 504
542 492 491 531 523 521
560 509 504 552 534 541
585 526 513 570 552 554
601 543 526 588 570 572
620 558 540 601 588 593
633 576 559 620 603 609
648 591 571 642 619 629
669 611 590 655 634 641
680 629 607 670 658 656
696 638 629 689 675 673
711 655 643 711 690 690
734 673 655 724 706 708
753 686 672 743 724 722
765 702 691 760 747 735
```

11:02

금요일, 11월 11

## 9장 실습 문제 3번

- 3. 정적 지역 변수가 사용되는 하나의 용도는 함수가 처음 호출될 때 초기화를 딱 한번만 수행하는 것이다.`inited`는 정적 변수이기 때문에 다음번의 호출에서도 그 값을 유지한다. 따라서 초기화 코드는 함수가 처음 호출될 때 한번만 실행된다. 이러한 구조를 사용하여 맨 처음 호출되는 경우에만 초기화를 수행하는 난수 발생 함수 `get_random()`을 작성하여 테스트 하라.

```
int get_random(void) {
    static int inited = 0 ;
    if(inited == 0 ) {
        ...
        inited = 1 ;
    }
    ...
}
```

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

int get_random() {
    static int inited = 0;
    if (! inited ) {
        inited = 1;
        printf (" inited \n");
    }
}

int main()
{
    get_random();
    get_random();
    get_random();
    get_random();
}

}
```

----- 문제 3번 실행을 시작합니다. -----

./3.x  
inited

----- 문제 3번 실행을 종료합니다. -----



연결됨

와이파이 네트워크 'DGU'에 연결되었습니다.

## 9장 실습 문제 4번

- 4. 다음과 같은 무한 수열을 계산하는 순환적인 프로그램을 작성하라.
  - $1/1+1/2+1/3+\dots+1/n$

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

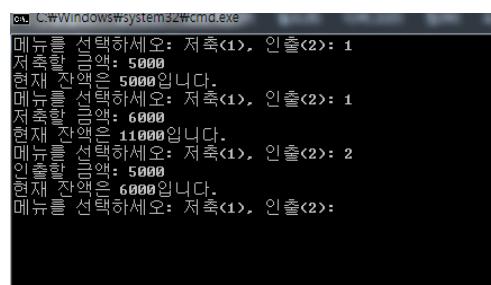
float f(int n) {
    if(n == 1) return 1;
    else return f(n-1) + (float)1/n;
}

int main(int c, char** v) {
    printf ("%f", f(atoi(v[1])));
}
```

```
zezeon@ubuntuZ: ~/Programming/basicProgramming
세 그 멘 테 이 션 오 류 (core dumped)
zezeon@ubuntuZ:~/Programming/basicProgramming$ ./4.x 30
3.994987 zezeon@ubuntuZ:~/Programming/basicProgramming$ ./4.x 3
1.833333 zezeon@ubuntuZ:~/Programming/basicProgramming$ ./4.x 2
1.500000 zezeon@ubuntuZ:~/Programming/basicProgramming$
```

## 9장 실습 문제 5번

- 5. 은행 계좌에서 저축하고 인출하는 프로그램을 작성하여 보자. `Save(int amount)` 함수는 저금할 금액 `amount`를 받으며 `save(100)`과 같이 호출된다. `draw(int amount)`은 예금 인출을 나타낸다. 사용자에게 메뉴를 보여주고 저축 또는 인출을 선택하게 한다.



```
0x0 C:\Windows\system32\cmd.exe
메뉴를 선택하세요: 저축<1>, 인출<2>: 1
저축할 금액: 5000
현재 잔액은 5000입니다.
메뉴를 선택하세요: 저축<1>, 인출<2>: 1
저축할 금액: 6000
현재 잔액은 11000입니다.
메뉴를 선택하세요: 저축<1>, 인출<2>: 2
인출할 금액: 5000
현재 잔액은 6000입니다.
메뉴를 선택하세요: 저축<1>, 인출<2>:
```

```
#include<stdio.h>

int money = 0;

void save(int m) {
    money += m;
}

void draw(int m) {
    money -= m;
}

int main() {
    int menu, m, end = 0;
    while(!end) {
        printf(" 메뉴를 선택하세요 : 저축 (1), 인출 (2)");
        scanf("%d", &menu);
        if(menu == 1) {
            printf(" 저축할 금액 :");
            scanf("%d", &m);
            save(m);
        } else if(menu == 2) {
            printf(" 인출할 금액 :");
            scanf("%d", &m);
            draw(m);
        } else end = 1;
        printf(" 현재 잔액은 %d입니다.\n", money);
    }
}
```

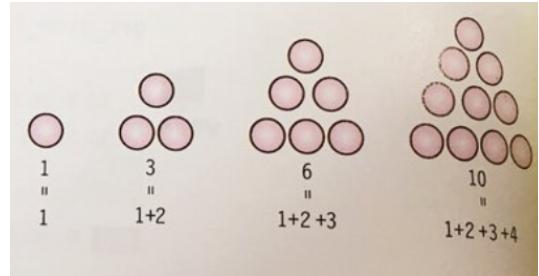
```

----- 문제 5번 실행을 시작합니다. -----
./5.x
메뉴를 선택하세요 : 저축(1), 인출(2)1
저축할 금액 : 3000
현재 잔액은 3000입니다.
메뉴를 선택하세요 : 저축(1), 인출(2)1
저축할 금액 : 300
현재 잔액은 3300입니다.
메뉴를 선택하세요 : 저축(1), 인출(2)2
인출할 금액 : 2500
현재 잔액은 800입니다.
메뉴를 선택하세요 : 저축(1), 인출(2)1
저축할 금액 : 50000
현재 잔액은 50800입니다.
메뉴를 선택하세요 : 저축(1), 인출(2)3
현재 잔액은 50800입니다.
----- 문제 5번 실행을 종료합니다. -----

```

## 9장 실습 문제 6번

- 6. 오른쪽과 같은 n 번째 삼각수를 계산하는 함수 `get_tri_number(int n)`을 순환 호출을 이용하여 작성하여 보자.



```

#include<stdio.h>

int get_tri_number(int n) {
    if(n == 1) return 1;
    return get_tri_number(n-1) + n;
}

int main()
{
    for(int i=1; i<10; i++)
        printf ("%d 번째 삼각수는 %d입니다.\n", i, get_tri_number(i));
}

```

1번째 삼각수는 1입니다.  
2번째 삼각수는 3입니다.  
3번째 삼각수는 6입니다.  
4번째 삼각수는 10입니다.  
5번째 삼각수는 15입니다.  
6번째 삼각수는 21입니다.  
7번째 삼각수는 28입니다.  
8번째 삼각수는 36입니다.  
9번째 삼각수는 45입니다.

----- 문제 6번 실행을 종료합니다. -----

## 9장 실습 문제 7번

- 7. 이항계수를 계산하는 순환 함수를 작성하라. 이항 계수는 다음과 같이 순환적으로 정의 된다. 반복함수로도 구현해보라.

$${}_nC_k = \begin{cases} {}_{n-1}C_{k-1} + {}_{n-1}C_k & \text{if } 0 < k < n \\ 1 & \text{if } k = 0 \text{ or } k = n \end{cases}$$

```
#include<stdio.h>

int nCr(int n, int r) {
    if(r == 0 || r == n) return 1;
    return nCr(n-1, r-1) + nCr(n-1, r);
}

int factorial (int n) {
    int r = 1;
    for(int i=1; i<=n; i++) r *= i;
    return r;
}

long nCk(int n, int k) {
    long r = 1;
    for(int i=k+1; i<=n; i++) r *= i;
    for(int i=1; i<=n-k; i++) r /= i;
    return r;
}
```

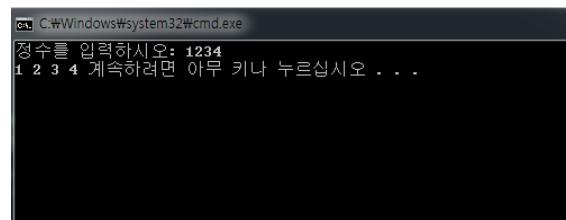
```
}
```

```
int main() {
    printf ("%d\n", nCr(10, 3));
    printf ("%d\n", nCk(10, 3));
}
```

```
./7.x
120
120
----- 문제 7번 실행을 종료합니다.
```

## 9장 실습 문제 8번

- 8. 순환 호출을 이용하여 정수의 각 자리 수를 출력하는 함수 `show_digit(int x)`를 작성하고 테스트 하라. 즉 정수가 1234이면 화면에 1 2 3 4 와 같이 출력한다. 함수는 일의 자리를 출력하고 나머지 부분을 대상으로 다시 같은 함수를 순환 호출한다. 예를 들어서 1234의 4를 출력하고 나머지 123을 가지고 다시 같은 함수를 순환 호출한다. 1234를 10으로 나누면 123이 되고 4는 1234를 10으로 나눈 나머지 이다.



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

```
void show_digit(int n) {
    printf ("%d ", n%10);
    if(n >9) show_digit(n/10);
}
```

```
int main() {
    show_digit(2346);
}
```

```
evince report.pdf
zezeon@ubuntuZ:~/Programming/basicProgramming$ rm 8.png
zezeon@ubuntuZ:~/Programming/basicProgramming$ make tex
----- 문제 8번 실행을 시작합니다.
./8.x
6 4 3 2 ----- 문제 8번 실행을 종료합니다.
```