프로그래머의 길
C 언어편

http://cafe.daum.net/pway

Chapter 1. 프로그래밍을 위한 언어의 선택
Chapter 2. 첫과제를 시작하기 전에

2.1 첫번째 프로그램
2.2 두번째 프로그램
2.3 세번째 프로그램
2.4 네번째 프로그램
2.4.1 for 문에 대한 이해 1
2.4.2 for 문에 대한 이해 2
2.4.3 for 문에 대한 이해 3
2.4.4 for 문에 대한 이해 4
2.5 다섯번째 프로그램
2.5.1 for 문에 대한 이해 5
2.5.2 for 문에 대한 이해 6
2.6 여섯번째 프로그램
2.7 일곱번째 프로그램
2.8 여덟번째 프로그램
2.8.1 for 문에 대한 이해 7
2.8.2 for 문에 대한 이해 8
2.8.3 for 문에 대한 이해 9
Chapter 3. 프로그램 과제소개
Chapter 4. 과제 모음 첫번째
Chapter 5. 과제 모음 두번째
Chapter 6. 다음 단계로 나가기
6.1 아홉번째 프로그램
6.1.1 이차원 배열이란 1
6.1.2 이차원 배열이란 2
6.2 열번째 프로그램
6.3 열한번째 프로그램
6.4 열두번째 프로그램
6.5 열세번째 프로그램
6.6 열네번째 프로그램
6.7 열다섯번째 프로그램
Chapter 7. 과제 모음 세번째(2 차원 배열문제)
Chapter 8. 초보를 뛰어넘기 위해 - 개미수열
Chapter 9. 과제 모음 다섯번째(1 차원 배열문제)
Chapter 10. 파일 입출력을 시작하며
Chapter 11. 파일 입출력 과제모음
Chapter 12. 함수에 관하여
Chapter 13. 초보를 마치면서 풀어보는 과제모음
Chapter 14. 중급으로 향하는 길 - C/C++, 자료구조 방향
Chapter 15. 선택의 기로 - 어디로 갈것인가?
Chapter 16. 정말로 실력을 키우기를 원한다면

Chapter 1. 프로그래밍을 위한 언어의 선택

여기 프로그래머의 길에서 주로 사용하게 되는 언어는 C 입니다. C 언어는 아주오래된 역사를 가지고 있으며 많은 프로그램들이 C 로 짜여지기는 했지만, 초보자들이 배우기에는 매우 악명이 높은 언어입니다.

한마디로 어렵다는게 그 첫번 째 이유입니다. 그래도 다른 중요한 언어들이 C 와 밀접한 관계를 가지고 있기 때문에 C 를 벗어나서 프로그램을 배우는 것도 그렇게 쉬운일 만은 아닙니다. C++, Java, C#등이 C 와 문법적인 구조가 상당히 연관성이 높습니다.

C 언어가 어려운 이유는 C 언어를 가르치는 선생님들이 배우는 사람의 이해력과 프로그래밍 능력을 벗어나는 C 언어의 구성요소들을 강제로 전달하려고 하기 때문에 생기는 경우가 태반입니다.

초보자들이 이해하고 구사할수 있는 C 언어의 구성요소 들은 사실 몇가지 정도밖에 되지 않습니다.

1. printf 함수의 사용 2. 간단한 변수의 사용 3. for 문의 사용

이것들만 가지고도 상당히 복잡한 프로그램을 작성할 수가 있습니다.

Chapter 2. 첫과제를 시작하기 전에

2.1 첫번째 프로그램

1. 다음 프로그램을 실행시켜보자.

```
#include<stdio.h>
int main()
{
    printf("This is my first program\n");
    return 0;
}
```

2. 조금더 복잡한 프로그램을 한번 돌려보자.

```
#include<stdio.h>
int main()
{
    printf("\n");
    printf("-------------------------\n");
    printf("This is my second program\n");
    printf("-------------------------\n");
    return 0;
}
```

우선 여러분이 익숙해져야 하는 것이 printf 함수이다. printf 함수를 통해서 여러분은 무엇인가를 보여줄 수 있게 되는 것이다.

2.2 두번째 프로그램

1. 다음 프로그램을 한번 실행시켜보자.

```
#include<stdio.h>
int main()
{
    int a, b, c;

    a = 1;
    b = 2;
    c = a + b;

    printf("a=%d b=%d c=%d\n", a, b, c);
    return 0;
}
```

프로그램을 짤때 변수의 개념은 매우 중요하다. 위의 프로그램에서 a, b, c 가 변수이다. 변수는

1. 이름을 가지고 있다.
2. 자료의 형(종류)이 있다.
3. 컴퓨터 내의 메모리 상에 존재한다.

(위치와 크기가 있다.)

4. 변수안에 자료를 담아서 보관할 수 있다.
5. 변수안에 있던 자료를 꺼내서 사용할 수 있다.

대개 이와같은 성질을 가진다.

2.3 세번째 프로그램

1. 다음 프로그램을 한번 실행시켜 보자.

```
#include<stdio.h>
int main()
{
    int a, b, c;

    printf("please enter two numbers:");
```

```
    scanf("%d %d", &a, &b);

    c = a + b;
    printf("\na=%d b=%d c=%d", a, b, c);
    return 0;
}
```

우리는 printf 함수를 사용하여 출력을 한다. 가장먼저 배울 입력에 관한 함수는 scanf 함수이다. scanf 함수를 사용할 때 주의해야 할점은 값을 받을 변수, 위의 경우 a, b 앞에 &문자를 붙여야 한다는 점이다. 이에관해서는 실력이 늘고 나서 알도록 하고 지금은 그냥 넘어 가도록 하자.

2.4 네번째 프로그램

1. 다음 프로그램을 실행시켜보자.

```
#include<stdio.h>
int main()
{
    int i, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    printf("\n");
    for(i=0;i<n;i++)
        printf("yes ");
    printf("\n print %d many yes",n);
    return 0;
}
```

조금 복잡한 프로그램이네요, for 문을 이해하기위한 기본이 되는 프로그램입니다.

2. 다음 프로그램을 한번 실행시켜 봅시다.

```
#include<stdio.h>
int main()
{
    int i, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    printf("\n");
    for(i=0;i<n;i++)
        printf("%d", i);
    printf("\n print the numbers from 0 to %d", n-1);
    return 0;
}
```

2.4.1 for 문에 대한 이해 1

이제 프로그램을 작성하기위해서 printf, scanf 함수 이외에 만나게된 첫번째 친구인 for 문을 알아보도록 하자.

for 문은 반복회수가 미리정해져 있는 경우에 사용하는 프로그램 제어구조이다.

10 번만 "yes "를 출력하자.

100 번만 "happy "를 출력하자.

1000 번만 "laugh "를 출력하자.

위의 경우에 사용하는 제어구조가 for 문인 것이다.

```
    for(i=0;i<10;i++)
        printf("yes ");

    for(i=0;i<100;i++)
        printf("happy ");

    for(i=0;i<1000;i++)
        printf("laugh ");
```

2.4.2 for 문에 대한 이해 2

for 문은 반복회수가 미리정해져 있는 경우에 사용한다고 했는데 그 사용회수가 꼭 숫자로 주어져야 하는 것은 아니고 변수로 주어질수도 있다.

다음의 경우를 생각해보자.

```
    for(i=0;i<n;i++)
        printf("yes ");
```

n 의 값에 따라서 출력되는 "yes "의 갯수가 달라질 것이다. for 문이 어떻게 동작하는지 이해하기 위해서 간단한 경우를 생각해 보자 우선 n 의 값을 3 이라고 가정하자.

위의 for 문이 실행되는 순서는 다음과 같다.

```
 0. i=0;
 1. i<n;  0<3 인가 테스트한다, 그렇다.
 2. printf("yes ");
 3. i++   i 가 1 증가한다 i=1 이된다.
 4  i<n;  1<3 인가 테스트한다, 그렇다.
 5. printf("yes ");
 6. i++   i 가 1 증가한다 i=2 가된다.
 7. i<n;  2<3 인가 테스트한다, 그렇다.
 8. printf("yes ");
 9. i++   i 가 1 증가한다 i=3 이된다.
10. i<n;  3<3 인가 테스트한다, 틀리다.
11. for 문을 종료한다.
```

2.4.3 for 문에 대한 이해 3

여기서는 for 문에 대해 조금더 자세히 알아보자.

```
    for(i=0;i<n;i++)
        printf("yes ");
```

for 문은 다음과 같은 구조를 가진다.

```
    for(초기화; 실행조건; 변수증가)
        실행문
```

위에서 i=0 은 초기화에 해당한다. 이 초기화는 for 문의 실행시 처음에 한번만 실행된다.

위에서 i<n 은 실행조건에 해당한다. 실행조건이 맞게되면 실행문을 실행한다음 변수증가를 실행하게된다. 다시 (실행조건-> 실행문-> 변수증가->)를 반복하게 된다. 실행조건이 틀리게 되면 for 문을 종료하게 된다.

위에서 printf("yes ")는 실행문이다. 위에서 i++ 은 변수증가에 해당한다.

for 문에서 실행문이 하나뿐이라는 것을 기억하고 있어야한다.

2.4.4 for 문에 대한 이해 4

for 문내에서 for 문을 제어하기 위해서 사용한 변수를 사용하는 경우에 대해서 알아보자.

```
    for(i=0;i<n;i++)
        printf("%d", i);
```

n 의 값을 3 이라고 가정하자.

위의 for 문이 실행되는 순서는 다음과 같다.

```
 0. i=0;
 1. i<n;  0<3 인가 테스트한다, 그렇다.
 2. printf("%d", i);
        0 을 출력한다
 3. i++   i 가 1 증가한다 i=1 이된다.
 4  i<n;  1<3 인가 테스트한다, 그렇다.
 5. printf("%d", i);
        1 을 출력한다
 6. i++   i 가 1 증가한다 i=2 가된다.
 7. i<n;  2<3 인가 테스트한다, 그렇다.
 8. printf("%d", i);
        2 를 출력한다
 9. i++   i 가 1 증가한다 i=3 이된다.
```

10. i<n; 3<3 인가 테스트한다, 틀리다.
11. for 문을 종료한다.

for 문을 종료했을때 i 의 값이 3 이 되어있다는 것을 꼭 기억하자.

2.5 다섯번째 프로그램

1. 다음 프로그램을 실행시켜보자.
```
#include<stdio.h>
int main()
{
    int i, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
        printf("%d", i+1);
    return 0;
}
```
2. 위의 program 에서

printf("%d", i+1); 대신에
printf("%d
", i+1); 로 바꿔서 실행해 보자.

3. 다음 프로그램을 실행시켜보자.
```
#include<stdio.h>
int main()
{
    int i, j, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
    {
         for(j=0;j<n;j++)
             printf("%4d", j+1);
         printf("\n");
    }
    return 0;
}
```
아니 이렇게 복잡한 프로그램을 돌려보라니, 하는 생각이 들지도 모르겠군요. 하지만 우리가 실력을 키우기 위해서 출발해야 할 장소는 바로 이곳이 되겠습니다.

2.5.1 for 문에 대한 이해 5

다섯번째 프로그램에 나왔던 다음 프로그램을 한번보자
```
 1.  #include<stdio.h>
 2.  int main()
 3.  {
 4.      int i, j, n;
 5.
 6.      printf("\nPlease enter number:");
 7.      scanf("%d", &n);
 9.
10.      for(i=0;i<n;i++)
11.      {
12.           for(j=0;j<n;j++)
13.               printf("%4d", j+1);
14.           printf("\n");
15.      }
16.      return 0;
17.  }
```
설명을 돕기위해서 각줄에 번호를 붙였다.

질문: for 문에는 실행문이 하나뿐이라고 했는데 왜 여러개가 있나요?

답변: 10 줄의 for(i=0;i<n;i++) 의 실행문은 11 줄 부터 15 줄 까지 입니다.

질문: 5 줄의 복잡한 문들이 하나의 실행문이라고 주장하시는 가요?

답변: 그렇습니다. C 에서는 복합문 (또는 블럭) 을 하나의 실행문으로
취급합니다. 복합문이란 여러개의 문을 묶어서 사용할 수 있도록
해 줍니다.

질문: 복합문 속에 for 문이 들어가도 되는 것입니까?

답변: 그렇습니다. 복합문은 아무리 복잡한 문이라도 담을 수 있는 그릇입니다.
지금까지 우리가 만난 가장 복잡한 문은 for 문입니다. 복합문 속에
for 문이 담겨 있다는 것은 자연 스러운 일입니다.

질문: 이 프로그램이 어떻게 동작하는지 이해할 수 있는 건가요?

답변: 물론입니다. 단지 약간의 상상력이 필요할 뿐입니다.

2.5.2 for 문에 대한 이해 6

다섯번째 프로그램에 나왔던 다음 프로그램을 한번보자

```
 1.  #include<stdio.h>
 2.  int main()
 3.  {
 4.      int i, j, n;
 5.
 6.      printf("\nPlease enter number:");
 7.      scanf("%d", &n);
 9.
10.      for(i=0;i<n;i++)
11.      {
12.          for(j=0;j<n;j++)
13.              printf("%4d", j+1);
14.          printf("\n");
15.      }
16.      return 0;
17.  }
```

이 프로그램이 어떻게 동작하는지 한번 알아보도록 하자. 우선 n 을 3 이라고 가정하자.

```
10 라인 for 문에서
    초기화:    i=0             i 는 0 이 된다
    실행조건:  i<3
    실행문:    11 라인 - 15 라인 까지 실행
          12 라인 for 문에서
              초기화:   j=0                   j 는 0 이 된다
              실행조건: j<3
              실행문:   printf("%4d", j+1);   1 을 출력한다
              변수증가: j++                   j 는 1 이 된다
              실행조건: j<3
              실행문:   printf("%4d", j+1);   2 를 출력한다
              변수증가: j++                   j 는 2 가 된다
              실행조건: j<3
              실행문:   printf("%4d", j+1);   3 을 출력한다
              변수증가: j++                   j 는 3 이 된다
              실행조건  j<3
              for 문(12 라인)을  종료한다.
          14 라인에서
              printf("\n");                    줄바꿈을 출력한다
          실행문(11 라인-15 라인)  종료
    변수증가:  i++              i 는 1 이 된다
    실행조건:  i<3
    실행문:    11 라인 - 15 라인 까지 실행
          12 라인 for 문에서
```

```
            초기화:   j=0                   j는 0 이 된다
            실행조건: j<3
            실행문:   printf("%4d", j+1);   1 을 출력한다
            변수증가: j++                   j는 1 이 된다
            실행조건: j<3
            실행문:   printf("%4d", j+1);   2 를 출력한다
            변수증가: j++                   j는 2 가 된다
            실행조건: j<3
            실행문:   printf("%4d", j+1);   3 을 출력한다
            변수증가: j++                   j는 3 이 된다
            실행조건  j<3
            for 문(12 라인)을  종료한다.
      14 라인에서
            printf("\n");                    줄바꿈을 출력한다
      실행문(11 라인-15 라인) 종료
   변수증가:  i++              i는 2가 된다
   실행조건:  i<3
   실행문:    11 라인 - 15 라인 까지 실행
      12 라인 for 문에서
            초기화:   j=0                   j는 0 이 된다
            실행조건: j<3
            실행문:   printf("%4d", j+1);   1 을 출력한다
            변수증가: j++                   j는 1 이 된다
            실행조건: j<3
            실행문:   printf("%4d", j+1);   2 를 출력한다
            변수증가: j++                   j는 2가 된다
            실행조건: j<3
            실행문:   printf("%4d", j+1);   3 을 출력한다
            변수증가: j++                   j는 3 이 된다
            실행조건  j<3
            for 문(12 라인)을  종료한다.
      14 라인에서
            printf("\n");                    줄바꿈을 출력한다
      실행문(11 라인-15 라인) 종료
   변수증가:  i++              i는 3이 된다
   실행조건:  i<3
   for 문(10 라인)을 종료한다.
```

따라서

```
   1   2   3
   1   2   3
   1   2   3
```

위와 같은 출력이 화면에 보여진다.

2.6 여섯번째 프로그램

1. 다음 프로그램을 실행시켜보자.

```
#include<stdio.h>
int main()
{
    int i, j, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
    {
         for(j=0;j<n;j++)
             printf("%4d", j+1);
         printf("\n");
    }
```

```
    return 0;
}
```

2. 12 라인 printf("%4d", j+1); 대신 printf("%4d", i+1); 를 넣어 프로그램을 실행시켜 보자.

3. 12 라인에 printf("%4d", i+j+1); 을 넣어 프로그램을 실행시켜 보자.

4. 12 라인에 printf("%4d", (i+1)*(j+1)); 을 넣어 프로그램을 실행시켜 보자.

프로그램을 실행시키기전에 그 결과를 예측할 수 있다면 for 문에 대해서 약간 이해가 깊어 졌다고 볼 수 있겠다.

2.7 일곱번째 프로그램

1. 다음 프로그램을 실행시켜보자.

```
#include<stdio.h>
int main()
{
    int i, j, k, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    k = 1;
    for(i=0;i<n;i++)
    {
         for(j=0;j<n;j++)
             printf("%4d", k++);
         printf("\n");
    }
    return 0;
}
```

이 프로그램의 출력이 어떠할 지 예측할 수 있는가? 예측할 수 없다면 곰곰히 따져 보자. 그리고 다시 for 문의 동작방식을 익혀보도록 하자.

2.8 여덟번째 프로그램

1. 다음 프로그램을 실행시켜보자.

```
#include<stdio.h>
int main()
{
    int i, j, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
    {
         for(j=0;j<i+1;j++)
             printf("%4d", j+1);
         printf("\n");
    }
    return 0;
}
```

2. 다음 프로그램을 실행시켜보자.

```
#include<stdio.h>
int main()
{
    int i, j, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
    {
         for(j=0;j<n-i;j++)
             printf("%4d", j+1);
         printf("\n");
    }
```

```
    return 0;
}
```

이 프로그램의 출력이 어떠할 지 예측할 수 있는가?

2.8.1 for 문에 대한 이해 7

일곱번째 프로그램에 나왔던 다음 프로그램을 한번보자

```
 1.  #include<stdio.h>
 2.  int main()
 3.  {
 4.      int i, j, k, n;
 5.
 6.      printf("\nPlease enter number:");
 7.      scanf("%d", &n);
 9.
10.      k=1;
11.      for(i=0;i<n;i++)
12.      {
13.           for(j=0;j<n;j++)
14.               printf("%4d", k++);
15.           printf("\n");
16.      }
17.      return 0;
18.  }
```

이 프로그램이 어떻게 동작하는지 한번 알아보도록 하자. 우선 n 을 3 이라고 가정하자.

```
10 라인 에서    k=1;            k 는 1 이 된다
11 라인 for 문에서
    초기화:    i=0             i 는 0 이 된다
    실행조건:  i<3
    실행문:    12 라인 - 16 라인 까지 실행
          13 라인 for 문에서
              초기화:   j=0                   j 는 0 이 된다
              실행조건: j<3
              실행문:   printf("%4d", k++);   1 을 출력한다 k 는 2 가 된다
              변수증가: j++                   j 는 1 이 된다
              실행조건: j<3
              실행문:   printf("%4d", k++);   2 를 출력한다 k 는 3 이 된다
              변수증가: j++                   j 는 2 가 된다
              실행조건: j<3
              실행문:   printf("%4d", k++);   3 을 출력한다 k 는 4 가 된다
              변수증가: j++                   j 는 3 이 된다
              실행조건  j<3
              for 문(13 라인)을  종료한다.
          15 라인에서
              printf("\n");                    줄바꿈을 출력한다
          실행문(12 라인-16 라인) 종료
    변수증가:  i++              i 는 1 이 된다
    실행조건:  i<3
    실행문:    12 라인 - 16 라인 까지 실행
          13 라인 for 문에서
              초기화:   j=0                   j 는 0 이 된다
              실행조건: j<3
              실행문:   printf("%4d", k++);   4 를 출력한다 k 는 5 가 된다
              변수증가: j++                   j 는 1 이 된다
              실행조건: j<3
              실행문:   printf("%4d", k++);   5 를 출력한다 k 는 6 이 된다
              변수증가: j++                   j 는 2 가 된다
              실행조건: j<3
              실행문:   printf("%4d", k++);   6 을 출력한다 k 는 7 이 된다
              변수증가: j++                   j 는 3 이 된다
              실행조건  j<3
              for 문(13 라인)을  종료한다.
```

```
        15 라인에서
            printf("\n");                    줄바꿈을 출력한다
        실행문(12 라인-16 라인) 종료
    변수증가:  i++              i 는 2 가 된다
    실행조건:  i<3
    실행문:    12 라인 - 16 라인 까지 실행
        13 라인 for 문에서
            초기화:   j=0                    j 는 0 이 된다
            실행조건: j<3
            실행문:   printf("%4d", k++);    7 을 출력한다 k 는 8 이 된다
            변수증가: j++                    j 는 1 이 된다
            실행조건: j<3
            실행문:   printf("%4d", k++);    8 을 출력한다 k 는 9 가 된다
            변수증가: j++                    j 는 2 가 된다
            실행조건: j<3
            실행문:   printf("%4d", k++);    9 를 출력한다 k 는 10 이 된다
            변수증가: j++                    j 는 3 이 된다
            실행조건  j<3
            for 문(13 라인)을  종료한다.
        15 라인에서
            printf("\n");                    줄바꿈을 출력한다
        실행문(12 라인-16 라인) 종료
    변수증가:  i++              i 는 3 이 된다
    실행조건:  i<3
    for 문(11 라인)을 종료한다.
```

따라서

```
   1   2   3
   4   5   6
   7   8   9
```

위와 같은 출력이 화면에 보여진다.

2.8.2 for 문에 대한 이해 8

여덟번째 프로그램에 나왔던 다음 프로그램을 보자.

```
 1.  #include<stdio.h>
 2.  int main()
 3.  {
 4.      int i, j, n;
 5.
 6.      printf("\nPlease enter number:");
 7.      scanf("%d", &n);
 8.
 9.      for(i=0;i<n;i++)
10.      {
11.          for(j=0;j<i+1;j++)
12.              printf("%4d", j+1);
13.          printf("\n");
14.      }
15.      return 0;
16.  }
```

이프로그램은 11 번 라인의 for 문속에 나오는 실행조건이 지금까지 보던것 과는 다르다. 즉 j<n 과 같이 정해져 있는 변수가 아니고 j<i+1 과 같이 변하는 변수 i 를 사용하고 있다는 점이 다르다고 하겠다.

우리가 지금까지 사용한 for 문에 나오는 실행조건들은 바로 실행회수를 나타내고 있다. i+1 도 실행회수를 나타내는 것이다.

지금까지 for 문을 잘 이해해 왔다면 i 의 값이 0, 1, 2, 3, . . . , n-1 로 변하다가 n 이 되면 for 문을 종료하고 나왔다는 사실을 알고 있을 것이다.

따라서 i+1 은 1, 2, 3, ... , n 과 같은 값을 가지게 된다. 한번, 두번, 세번, . . . , n 번 같이 회수를 변화 시킬 필요가 있을때 사용할 수 있는 방법이 되겠다.

n=3 일때의 출력은 다름과 같다.

```
1
1   2
1   2   3
```

2.8.3 for 문에 대한 이해 9

여덟번째 프로그램에 나왔던 다음 프로그램을 보자.

```
 1.  #include<stdio.h>
 2.  int main()
 3.  {
 4.      int i, j, n;
 5.
 6.      printf("\nPlease enter number:");
 7.      scanf("%d", &n);
 8.
 9.      for(i=0;i<n;i++)
10.      {
11.           for(j=0;j<n-i;j++)
12.               printf("%4d", j+1);
13.           printf("\n");
14.      }
15.      return 0;
16.  }
```

n-i 는 n, n-1, n-2, ... , 3, 2, 1 과 같은 값을 가지게 된다. n 번, n-1 번, n-2 번, . . . , 3 번, 2 번 1 번 같이 회수를 변화 시킬 필요가 있을때 사용할 수 있는 방법이 되겠다.

n=3 일때의 출력은 다름과 같다.

```
1   2   3
1   2
1
```

Chapter 3. 프로그램 과제소개

이제 for 문에 대한 기초적인 무장을 하고 for 문만 사용해서 풀수 있는 과제들을 한번 풀어 보기로 하자

프로그램 과제는 자기힘으로 풀어야만 의미가 있기 때문에 다른사람이 작성한 프로그램을 통해서 실력을 키우겠다는 생각은 매우 별로 실현 가능성이 없다고 보면 되겠다.

아무리 힘들더라도 자기힘으로 프로그램과제 들을 풀어나가자 그것이 바로 프로그래머의 길이다.

프로그램 과제에 올라온 프로그램을 푸는 것은 실력향상을 원하는 각자가 꼭 해보아야 할것이다. 따라서 프로그램과제에 대한 과도한 도움을 주는것을 금지하고자 한다.

프로그램 과제에 대해 그결과가 되는 프로그램을 올리는 것은 이곳 프로그래머의 길에서 용납될수 없는 행위가 될것임을 엄숙히 경고 한다.

Chapter 4. 과제 모음 첫번째

문 1) 다음과 같이 임의의 숫자를 입력하였을 경우 숫자 값에 따른 결과를 출력하는 프로그램을 작성하시오.

1-1) number = 5

```
 1  2  3  4  5
 6  7  8  9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```

1-2) number = 5

```
21 22 23 24 25
16 17 18 19 20
11 12 13 14 15
 6  7  8  9 10
 1  2  3  4  5
```

1-3) number = 5

```
 1  3  5  7  9
11 13 15 17 19
21 23 25 27 29
31 33 35 37 39
41 43 45 47 49
```

1-4) number = 5

```
 1
 1  2
 1  2  3
 1  2  3  4
 1  2  3  4  5
```

1-5) number = 5

```
 1
 2  3
 4  5  6
 7  8  9 10
11 12 13 14 15
```

1-6) number = 5

```
 1  2  3  4  5
 1  2  3  4
 1  2  3
 1  2
 1
```

1-7) number = 5

```
 1  2  3  4  5
 6  7  8  9
10 11 12
13 14
15
```

1-8) number = 5

```
 1  2  3  4  5
 2  3  4  5  6
 3  4  5  6  7
 4  5  6  7  8
 5  6  7  8  9
```

1-9) number = 5

```
 1  2  3  4  5
 2  3  4  5  1
 3  4  5  1  2
 4  5  1  2  3
 5  1  2  3  4
```

1-10) number = 5

```
                1
             2  3
          4  5  6
       7  8  9 10
   11 12 13 14 15
```

Chapter 5. 과제 모음 두번째

두번째 과제모음에서 프로그램작성시에 for 문과 printf 문 이외에는 다른 문이나 함수를 사용하지 않고 프로그램을 작성하도록 하자.

즉 화면상에서 커서의 위치를 옮기는 함수 같은 것은 사용하지 않고 프로그램을 작성해 보도록 하자.

문 2) 다음과 같이 임의의 숫자를 입력하였을 경우 숫자 값에 따른 결과를 출력하는 프로그램을 작성하시오.

2-1) number = 5

```
*****
*****
*****
*****
*****
```

2-2) number = 5

```
*
**
***
****
*****
```

2-3) number = 5

```
    *
   **
  ***
 ****
*****
```

2-4) number = 5

```
    *
   ***
  *****
 *******
*********
```

2-5) number = 5

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

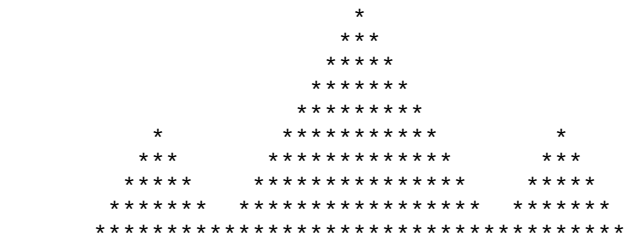
2-6) number = 5

2-7) number = 5

[n 줄의 출력 , 2-8 을 위한 보조 문제]

2-8) number = 5

[2*n 줄의 출력]



2-9) number = 5

```
$$$$$$$
$*****$
$*****$
$*****$
$*****$
$*****$
$$$$$$$
```

2-10) number = 5

```
[  n+2 줄 + n+1 줄 = 2*n + 3 줄      ]
[  *가 2 개 있는 줄과                ]
[  *가 1 개 있는 줄은 달리 찍어야 함 ]
```

```
*
**
*@*
*@@*
*@@@*
*@@@@*
*@@@@@*
*@@@@*
*@@@*
*@@*
*@*
**
*
```

문자열을 찍는 법

```
#include<stdio.h>
int main()
{
    int i, j, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("*");
        printf("\n");
    }
    return 0;
}
```

Chapter 6. 다음 단계로 나가기

이제 초보자로서 슬슬 프로그램에 재미를 붙여 가는 중입니다

for 문과 printf 문을 이용해서 여러가지의 프로그램을 작성해 보니 제법 재미가 있습니다. 프로그램이 제 적성에 맞는것 같은데요.

이런사람을 위해서 보다 흥미진진한 프로그래머의 길로 여러분을 안내해 드리려고 합니다.

for 문이 2 층 구조로된 제어구조는 상당히 흥미있는 여러가지 문제를 가능하게 합니다. 하지만 보다 재미있는 문제를 맛보려면 아무래도 다른 무언가를 새로 소개해야만 할 것 같습니다.

2 층의 for 문과 밀접한 관계를 가지고 있는 2 차원 배열을 소개합니다. 2 차원 배열은 1 차원 배열을 이해하고 나서 배우는 것이 순서입니다.

하지만 2 가지를 한꺼번에 배우도록 합시다.

배열이란 변수가 확장된 한가지 형태입니다. 동일한 형의 변수를 여러개 묶어서 사용할 수 있게 만든 것이 배열입니다.

우리주위에서 볼 수 있는 가장 가까운 배열을 닮은 꼴은 아파트입니다. 아파트 한동은 보통 15 개의 층으로 이루어져 있습니다, 각층에는 거의 같은 아파트들이 8 채 또는 6 채 이렇게 모여 있습니다.

아파트내에 있는 각각의 집들을 구분하기 위해서 1408 호 같은 숫자를 사용합니다. 14 층에 있는 8 호 라는 뜻을 보통 가지게 됩니다.

이것을 이렇게 표시한다고 해 봅시다.

수정아파트 101 동 14 층 8 호

그러면 수정아파트 101 동 101 호는 다음과 같이 표시됩니다.

수정아파트 101 동 1 층 1 호

자 층과 호를 빼고 표시해 보도록 합시다.

수정아파트 101 동 11

수정아파트 101 동 대신 data 라고 써봅시다.

data11

네 그렇습니다 바로 이것이 2 차원 배열인 것입니다.

6.1 아홉번째 프로그램

1. 다음 프로그램을 실행시켜 보자.

```
#include<stdio.h>
int main()
{
    int a[30][30];
    int i, j, k, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    k=0;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            a[i][j] = k++;

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("%4d", a[i][j]);
        printf("\n");
    }
    return 0;
}
```

6.1.1 이차원 배열이란 1

이차원 배열도 변수와 마찬가지로 다음과 같은 성질을 가진다.

1. 이름이 있다.
2. 자료의 형이 있다.
3. 컴퓨터 내의 메모리 상에 존재한다. (위치와 크기가 있다)
4. 값을 보관할 수있다.
5. 보관했던 값을 꺼내서 사용할 수 있다.

이외에도 배열은 변수에 없는 다른 요소가 있다. 즉 배열내에 있는 여러개의 방을 구분하기 위해서 방번호를 필요로 한다. 2 차원 배열의 경우에는 층번호와 방번호 이렇게 2 가지가 필요하다. 1 차원 배열의 경우에는 층번호는 없고 방번호만 필요하다.

C 에서 배열의 층번호나 방번호는 0 부터 시작한다. 10 개의 방이 있다면, 방번호는 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 이렇게 10 개가 되는 것이다. 아마 여러분은 이러한 숫자들의 모임을 여러번 본적이 있을 것이다. for 문을 돌릴 때 변수들이 변해가는 것이 바로 위의 경우와 같은 것이다.

여러분은 for 문과 배열의 연관성에 대해서 무언가 있구나 하는 것을 느낄 지도 모른다.

층번호도 0 부터 시작해서 1, 2, 3, ,,, 이렇게 변해 가는 것이다.

아홉번째 프로그램에서 이차원 배열을 선언한 것을 한번 보도록 하자.

int a3030;

이 배열의 이름은 a 이다. a 를 택한 이유는 간단하기 때문이지 특별한 이유가 있는 것이 아니다. 여러분은 배열의 이름으로 다른 이름을 택할 수가 있다. a 의 자료형은 정수이다. a 는 30 층을 가지고 있고 각층마다 30 개의 방이 있다. 모두 900 개의 방이 있는 셈이다.

왜 하필 30 개의 층 30 개의 방을 택했느냐 하면 그정도면 여러분이 앞으로 짜는 프로그램에 사용하기 충분하기 때문이다.

a 의 방들은

```
    a[ 0][ 0], a[ 0][ 1], a[ 0][ 2], a[ 0][ 3], ,,, a[ 0][28], a[ 0][29],
    a[ 1][ 0], a[ 1][ 1], a[ 1][ 2], a[ 1][ 3], ,,, a[ 1][28], a[ 1][29],
    a[ 2][ 0], a[ 2][ 1], a[ 2][ 2], a[ 2][ 3], ,,, a[ 2][28], a[ 2][29],

       . . . . . .

    a[29][ 0], a[29][ 1], a[29][ 2], a[29][ 3], ,,, a[29][28], a[29][29]
```

와 같은 순서로 메모리 상에서 위치하고 있다.

위에서는 각 방을 나타내기 위해서 숫자를 사용하는 방법을 보여 주고 있다. 하지만 항상 숫자로만 배열의 방들을 나타낼 수있다면 흥미있는 프로그램을 짜기가 어려울 것이다. 따라서 층이나 방을 나타내기 위해서 i 나 j 같은 변수를 사용한다.

6.1.2 이차원 배열이란 2

다시 아홉번째 프로그램을 보도록 하자

```
 1.  #include<stdio.h>
 2.  int main()
 3.  {
 4.      int a[30][30];
 5.      int i, j, k, n;
 6.
 7.      printf("\nPlease enter number:");
 8.      scanf("%d", &n);
 9.
10.      k=0;
11.      for(i=0;i<n;i++)
12.          for(j=0;j<n;j++)
13.              a[i][j] = k++;
14.
15.      for(i=0;i<n;i++)
16.      {
17.          for(j=0;j<n;j++)
18.              printf("%4d", a[i][j]);
19.          printf("\n");
20.      }
```

```
21.       return 0;
22.   }
```

11 라인 부터 13 라인 까지 for 문을 통해서 배열에 정수 값을 채운다 n 개의 층에 각 층마다 n 개의 방에다가 숫자를 1 씩 증가 시켜가면서 채우는 것이다.

15 라인 부터 20 라인 까지 for 문을 통해서 배열에 들어 있는 값을 찍어 층별로 한줄씩 찍어주는 것이다.

앞으로 여러분이 짜는 2 차원 배열문제는 대부분의 경우 이와 같이 값을 넣는 부분과 배열에 들어 있는 값을 찍는 부분으로 나누어서 작성하도록 한다.

6.2 열번째 프로그램

1. 다음 프로그램을 실행해 보자

```
#include<stdio.h>
int main()
{
    int a[30][30];
    int i, j, k, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            a[i][j]=j+1;   // [A]
    }

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("%4d", a[i][j]);
        printf("\n");
    }
    return 0;
}
```

어떤 결과가 나올지 미리 짐작해 보자.

2. 위의 프로그램에서

```
   가: a[i][j]=i+1; 을 넣어보자.
   나: a[i][j]=i+j+1; 을 넣어보자.
   다: a[i][j]=i-j;   를 넣어보자.
   라: a[i][j]=j-i;   를 넣어보자.
   마: a[i][j]=(i+1)*(j+1); 을 넣어보자.
```

어떤 결과가 나올지 미리 짐작해 보자.

6.3 열한번째 프로그램

1. 다음 프로그램을 실행해 보자.

```
#include<stdio.h>
int main()
{
    int a[30][30]={0,};
    int i, j, k, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
    {
        a[i][i]=i+1;   // [A]
    }

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
```

```
            printf("%4d", a[i][j]);
        printf("\n");
    }
    return 0;
}
```

2. 위의 프로그램에서

```
   가: a[i][i]=n-i;     을 넣어보자.
   나: a[0][i]=i+1;     을 넣어보자.
   다: a[n-1][i]=i+1;   을 넣어보자.
   라: a[i][0]=i+i;     을 넣어보자.
   마: a[i][n-1]=i+1;   을 넣어보자.
   바: a[i][n-i-1]=i+1; 을 넣어보자.
```

어떤 결과가 나올지 미리 짐작해 보자.

6.4 열두번째 프로그램

1. 다음 프로그램을 실행해 보자.

```
#include<stdio.h>
int main()
{
    int a[30][30]={0,};
    int i, j, k, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    k=0;
    for(i=0;i<n;i++)
    {
        a[i][i]=k++;   // [A]
    }

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("%4d", a[i][j]);
        printf("\n");
    }
    return 0;
}
```

2. 위의 프로그램에서

```
   가: a[0][i]=k++;     을 넣어보자.
   나: a[n-1][i]=k++;   을 넣어보자.
   다: a[i][0]=k++;     을 넣어보자.
   라: a[i][n-1]=k++;   을 넣어보자.
   마: a[i][n-i-1]=k++; 을 넣어보자.
   바: a[n-i-1][i]=k++; 을 넣어보자.
```

어떤 결과가 나올지 미리 짐작해 보자.

6.5 열세번째 프로그램

1. 다음 프로그램을 실행해 보자.

```
#include<stdio.h>
int main()
{
    int a[30][30]={0,};
    int i, j, k, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    k=0;
    for(i=0;i<n;i++)
    {
```

```
        for(j=0;j<n;j++)
            a[i][j]=k++;   // [A]
    }

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("%4d", a[i][j]);
        printf("\n");
    }
    return 0;
}
```

2. 위의 프로그램에서

```
   가: a[j][i]=k++;           을 넣어보자.
   나: a[i][n-j-1]=k++;       을 넣어보자.
   다: a[n-i-1][j]=k++;        을 넣어보자.
   라: a[n-i-1][n-j-1]=k++;   을 넣어보자.
   마: a[j][n-i-1]=k++;       을 넣어보자.
   바: a[n-j-1][i]=k++;       을 넣어보자.
```

어떤 결과가 나올지 미리 짐작해 보자.

6.6 열네번째 프로그램

1. 다음 프로그램을 실행시켜 보자.

```
#include<stdio.h>
int main()
{
    int i, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
    {
        if((i%2)==0)
            printf("\ni=%d is even number", i);   // 짝수
        else
            printf("\ni=%d is odd number", i);    // 홀수
    }
    return 0;
}
```

이 프로그램에서 우리는 처음으로 if 문을 사용하였다. 2 차원 배열문제 부터는 if 문을 사용할 필요가 생긴다. 이제 if 문에 대해서 공부해 보기로 하자.

6.7 열다섯번째 프로그램

1. 다음 프로그램을 실행시켜보자.

```
#include<stdio.h>
int main()
{
    int a[30][30];
    int i, j, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(i==j)
                a[i][j]=9;
            else
                a[i][j]=1;
        }
    }
```

```
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("%4d", a[i][j]);
        printf("\n");
    }
    return 0;
}
```

위의 프로그램은 number=5: 일때 다음과 같은 출력을 준다.

```
   9   1   1   1   1
   1   9   1   1   1
   1   1   9   1   1
   1   1   1   9   1
   1   1   1   1   9
```

가만히 보면 a00, a11, a22, a33, a44 에만 9 가 들어있고 나머지 칸에는 1 이 들어 있음을 볼 수 있다. 왜 그리 될까 한번 짐작해 보기로 하자. 많은 경우에 스스로 짐작하여 이러 이러 하리라고 생각하고 그것이 맞는지 확인하는 방법이 도움을 줄 경우가 있다.

2. 다음 프로그램을 실행시켜보자.

```
#include<stdio.h>
int main()
{
    int a[30][30];
    int i, j, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(i>j)
                a[i][j]=9;
            else
                a[i][j]=1;
        }
    }

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("%4d", a[i][j]);
        printf("\n");
    }
    return 0;
}
```

위의 프로그램은 number=5: 일때 다음과 같은 출력을 준다.

```
   1   1   1   1   1
   9   1   1   1   1
   9   9   1   1   1
   9   9   9   1   1
   9   9   9   9   1
```

왜 그리 될까 한번 짐작해 보기로 하자.

위의 프로그램에서 if(i>j) 대신 if(i<j)를 넣으면 어떤 출력이 나올지 짐작해 보자. 또 if(i>=j)일 경우와, if(i<=j) 일 경우에 대해서도 생각해보자.

다시 열네번째 프로그램으로 돌아가 보면

`if((i%2)==0)` 라는 조건이 나온다.

`%` 는 나머지 연산자이다. `i%2` 는 `i` 를 `2` 로 나눈 나머지를 나타낸다. `i` 가 짝수라면 나마지가 `0` 이 되고, 홀수라면 나머지가 `1` 이 된다. 따라서 `if((i%2)==0)` 이라는 조건은 `i` 가 짝수일때 참이 되는 조건인 것이다.

열다섯번째 프로그램에서 if(i==j)대신에 if((i%2)==0)를 넣으면 어떤 출력이 나올지 한번 짐작해 보기로 하자.

이제 이차원 배열을 이용한 과제를 시작할 준비가 되었다. 이토록 지리한 연습을 통해서 우리는 이차원 배열이 어쩌면 그렇게 어려운 것만은 아닐 것 같다는 희망의 냄새를 맡게 되었다.

지금부터 더욱 흥미가 더해지는 프로그램의 세계로 떠나보자.

Chapter 7. 과제 모음 세번째(2차원 배열문제)

문 3) 2 차원 배열에서 다음과 같이 임의의 숫자를 입력하였을 경우 숫자 값에 따른 결과를 출력하는 프로그램을 작성하시오.

3-1) 1 씩 증가하는 변수를 사용하여 배열에 값을 넣도록 프로그램을 작성하시오.

```
number = 5

 1  2  3  4  5
 6  7  8  9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```

3-2) 1 씩 증가하는 변수를 사용하여 배열에 값을 넣도록 프로그램을 작성하시오.

```
number = 5

 1  6 11 16 21
 2  7 12 17 22
 3  8 13 18 23
 4  9 14 19 24
 5 10 15 20 25
```

3-3) 1 씩 증가하는 변수를 사용하여 배열에 값을 넣도록 프로그램을 작성하시오.

```
number = 5

21 22 23 24 25
16 17 18 19 20
11 12 13 14 15
 6  7  8  9 10
 1  2  3  4  5
```

3-4) 1 씩 증가하는 변수를 사용하여 배열에 값을 넣도록 프로그램을 작성하시오. i 의 값이 짝수인 지 홀수 인지에 따라서 오른쪽으로 진행할 지 왼쪽으로 진행할 지 달라 지도록 프로그램을 작성 하시오.

```
number = 5

 1  2  3  4  5
10  9  8  7  6
11 12 13 14 15
20 19 18 17 16
21 22 23 24 25
```

3-5) 테두리와 대각선에는 9 를 채우고 나머지는 0 으로 채우는 프로그램을 작성하시오.

```
number = 7

 9  9  9  9  9  9  9
 9  9  0  0  0  9  9
 9  0  9  0  9  0  9
 9  0  0  9  0  0  9
 9  0  9  0  9  0  9
 9  9  0  0  0  9  9
 9  9  9  9  9  9  9
```

3-6) (i ㅁ j)의 값이 짝수인 지 홀수 인지에 따라서 0 이나 1 을 출력하는 프로그램을 작성하시오. ㅁ 는 사칙연산 (+,-,*,/) 중의 하나임.

```
number = 5

 1  0  1  0  1
 0  1  0  1  0
 1  0  1  0  1
 0  1  0  1  0
 1  0  1  0  1
```

3-7) 3-6 문제의 응용편

```
number = 6

 1  1  0  0  1  1
 1  1  0  0  1  1
 0  0  1  1  0  0
 0  0  1  1  0  0
 1  1  0  0  1  1
 1  1  0  0  1  1
```

3-8) 3-6 문제의 응용편

```
number = 6

 1  1  1  0  0  0
 1  1  1  0  0  0
 1  1  1  0  0  0
 0  0  0  1  1  1
 0  0  0  1  1  1
 0  0  0  1  1  1
```

3-6, 3-7, 3-8) 은 거의 같은 문제입니다.

3-9) 1 씩 증가하는 변수를 사용하여 배열에 값을 넣도록 프로그램을 작성하시오.

```
number = 5

 1  2  3  4  5
16  0  0  0  6
15  0  0  0  7
14  0  0  0  8
13 12 11 10  9
```

3-10) 1 씩 증가하는 변수를 사용하여 배열에 값을 넣도록 프로그램을 작성하시오.

```
number = 5

 1  2  3  4  5
16 17 18 19  6
15 24 25 20  7
14 23 22 21  8
13 12 11 10  9
```

3-11) number 가 홀수 일때만 풀도록 하시오.

```
number = 5

 3  3  3  3  3
 3  2  2  2  3
 3  2  1  2  3
 3  2  2  2  3
 3  3  3  3  3
```

3-12) 1 씩 증가하는 변수를 사용하여 배열에 값을 넣도록 프로그램을 작성하시오.

```
number = 5

 1  3  6 10 15
 2  5  9 14 19
 4  8 13 18 22
 7 12 17 21 24
11 16 20 23 25
```

3-13) 홀수 마방진 프로그램을 작성하시오

```
number = 3 일때:

 8  1  6
 3  5  7
 4  9  2

number = 5 일때:

17 24  1  8 15
23  5  7 14 16
 4  6 13 20 22
10 12 19 21  3
11 18 25  2  9
```

홀수 마방진을 만드는 규칙

1. 주어진 배열(3X3, 또는 5x5)의 맨위줄의
   가운데 칸에서 시작한다.
2. 현재위치에서 대각선으로 오른쪽 위로 진행하면서
   다음 값을 넣는다.
3. 첫 줄 다음은 맨아래 줄이 된다.
4. 마지막 칸의 다음은 첫 칸이 된다.
5. 진행방향에 값이 이미 들어있을 경우에는
   원래의 칸에서 한칸 밑으로 내려온다.

예를 들어서 3x3 홀수 마방진의 경우를 보자.

1. a[0][1] 에서 출발한다.
2. 오른쪽 위의 대각선 방향으로 진행한다.
   a[-1][2] 로 진행해야 하지만 -1 줄은 없는경우이다.
   따라서 a[2][2]로 진행한다.
3. 오른쪽 위의 대각선 방향으로 진행한다.
   a[1][3] 으로 진행해야 하지만 3 칸은 없다.
   따라써 a[1][0] 으로 진행한다.
4. 오른쪽 위의 대각선 방향으로 진행한다.
   a[0][1] 로 진행해야 하지만 이미 값이 들어있다.
   따라서 a[2][0] 으로 한칸 내려 온다.
5. 오른쪽 위의 대각선 방향으로 진행한다.
   a[1][1] 로 진행한다.
6. 오른쪽 위의 대각선 방향으로 진행한다.
   a[0][2] 로 진행한다.
7. 오른쪽 위의 대각선 방향으로 진행한다.
   a[-1][3] 으로 진행한다. -1 줄, 3 칸은 없는 경우이다.
   따라서 a[2][0]으로 진행한다. 이미 값이 들어있다.
   따라서 원래의 위치 a[0][2]에서 한칸 내려온다.
   a[1][2]로 진행한다.
8. 오른쪽 위의 대각선 방향으로 진행한다.
   a[0][3] 으로 진행한다. 3 은 없는 칸이다.
   따라서 a[0][0] 으로 진행한다.

9. 오른쪽 위의 대각선 방향으로 진행한다.
   a[-1][1]로 진행한다. -1 줄은 없는 줄이다.
   따라서 a[2][1]로 진행한다.
끝.

이와같이 마방진을 채워나간다.

Chapter 8. 초보를 뛰어넘기 위해-개미수열

마방진까지도 프로그램을 마친 당신은 이제 무서울 것이 없는 기분이다. 하지만 당신이 초보를 면하기 위해서는 정말 넘어야할 산이 당신을 기다리고 있다. 이 산을 당신 스스로의 힘으로 넘는 다면 당신은 이제 자신있게 초보를 면했다고 외칠 수 있다.

1. 다음 프로그램을 보도록 하자.

```
#include<stdio.h>
int main()
{
    int i, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    i=0;
    while(i<n) {
        printf("\ni=%4d", i);
        i++;
    }
    return 0;
}
```

위의 프로그램은 다음 프로그램과 같은 일을 한다.

```
    for(i=0;i<n;i++)
        printf("\ni=%4d", i);
```

실행하는 순서도 꼭 같을 뿐만 아니라 for 문이나 while 문을 마치고 난후에 i 가 가지는 값도 같다.

while 문은 for 문을 대신하기 위해서 사용하는 것이 아니라 실행시 발생하는 조건에 따라서 반복할 지 그만둘 지가 정해지는 경우에 사용하는 문이다.

```
    while(배가고플때) {
        밥을 먹는다;
    }
```

위의 while 문은 배가 불러지면 밥을 그만 먹는다는 것을 알 수 있을 것이다. 몇 숟가락을 먹어야 배가 부를지는 경우에 따라 다를 것이다.

2. 다음 프로그램을 보도록 하자.

```
#include<stdio.h>
int main()
{
    int a[100]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, };
    int i, n;

    printf("\nPrint the contents of array a[]");

    i=0;
    while(a[i]>0) {
        printf("%4d", a[i]);
        i++;
    }
    return 0;
}
```

지금까지 설명을 하지 않고 있었던 것중의 하나가 배열을 초기화 하는 부분이다. 위의 경우에 1 차원배열 a[]는 프로그램이 시작하기전에 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0, 0, 0, . . ., 끝까지 0 으로 초기화가 된다.

위의 프로그램을 실행하면 다음과 같은 출력이 나온다.

1 2 3 4 5 6 7 8 9 10

왜 그런지 한번 생각해보자.

3. 다음 프로그램을 보도록 하자.

```
#include<stdio.h>
int main()
{
    int a[100]={11, 12, 13, 14, 15, 6, 7, 8, 9, 10, };
    int i, n;

    printf("\nPrint the contents of array a[]");
```

```
    i=0;
    while(a[i]>0) {
        if(a[i]>10)
            printf("big ");
        else
            printf("small ");
        i++;
    }
    return 0;
}
```

위의 프로그램을 실행하면 다음과 같은 출력이 나온다.

big big big big big small small small small small

왜 그런지 한번 생각해보자.

4. break 를 소개하기 위하여 위의 프로그램을 다음과 같이 고쳐보도록 하자.

```
#include<stdio.h>
int main()
{
    int a[100]={11, 12, 13, 14, 15, 6, 7, 8, 9, 10, };
    int i, n;

    printf("\nPrint the contents of array a[]");

    i=0;
    while(1) {
        if(a[i]<=0)
            break;
        if(a[i]>10)
            printf("big ");
        else
            printf("small ");
        i++;
    }
    return 0;
}
```

위의 while(1)에서 1 은 항상 참인 조건이다. 따라서 위의 while 문은 영원히 돌수 있는 문이다. 이러한 경우에 while 문을 빠져 나오기 위하여 break 문을 사용할 수 있다. 위의 프로그램은 3.의 프로그램과 꼭 같이 실행된다.

break 문이 while 루프 내에서 처음에 나오는 경우는 흔한 것은 아니다. 단지 예를 보여주다보니 그렇게 된 것임을 이해하기 바란다.

5. 다음 프로그램을 보도록 하자.

```
#include<stdio.h>
int main()
{
    int a[100]={11, 12, 13, 14, 15, 6, 7, 8, 9, 10, };
    int b[100]={0,};
    int i, n;

    i=0;
    while(a[i]>0) {    // a[]배열을 b[]배열에 복사 하는 프로그램
        b[i]=a[i];
        i++;
    }
    printf("\nPrint the contents of array b[]");

    i=0;
    while(b[i]>0) {
        printf("%4d", b[i]);
        i++;
    }
    return 0;
}
```

위의 프로그램은 a]배열에 들어있는 값들을 b[배열로 복사하는 프로그램이다.

자 우리는 초보를 뛰어넘기위해 넘어야 하는 진정한 프로그램 과제를 마주칠 준비가 되었다. 한번 문제를 만나 보기로 하자.

지금까지 일차원 배열에 관해서 몇가지의 예가 나왔으므로 아마도 다음 문제는 일차원 배열에 관한 문제일 것이라는 것을 알아 차린 사람은 눈치까 빠르다고 하겠다.

우리가 풀어야할 문제는 베르나르 베르베르의 소설 "개미" 에 나오는 개미수열이다. 이 수열은 다음과 같다.

```
1
1 1
1 2
1 1 2 1
1 2 2 1 1 1
1 1 2 2 1 3
1 2 2 2 1 1 3 1
1 1 2 3 1 2 3 1 1 1
1 2 2 1 3 1 1 1 2 1 3 1 1 3
1 1 2 2 1 1 3 1 1 3 2 1 1 1 3 1 1 2 3 1
. . . . . .
```

이와 같이 계속해서 무한대로 나간다.

첫째줄의 1 로 부터 둘째줄의 1 1 이 나오고 둘째줄의 1 1 로 부터 세째줄의 1 2 가 나온다. 마찬가지로 세째줄로 부터 네째줄이 나오고, 이렇게 계속되는 것이다. 개미수열을 잘모르는 사람은 위의 예로 부터 다음 수열이 어떻게 나오는지 알아 내도록 하자. 개미수열의 규칙을 알아내는 것은 재미 있는 퍼즐이다.

일차원 배열 2 개를 사용하여 개미수열을 구하는 프로그램을 작성하라 입력받은 n 만큼의 줄수를 출력하도록 프로그램을 작성하면 된다.

개미수열 프로그램을 작성할 때 주된 제어구조로 while 문을 사용하도록 하고, break 문은 사용하지 않도록 하자.

Chapter 9. 과제 모음 다섯번째(1차원 배열문제)

개미수열을 푼 당신은 초보를 넘었다는 기쁨에 환희 의 소리를 지른다.

너무 좋아하기 전에 정말 초보를 뛰어넘은 실력이 있는지 다음 문제를 풀어 보도록 하자.

1. 파스칼의 삼각형

```
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5   10  10  5   1
        .  .  .  .  .  .  .  .  .  .  .  .
```

일차원 배열 2 개를 사용하여 파스칼의 삼각형을 출력하는 프로그램을 작성하시오.

2. 체를 이용한 소수 구하기

크기가 1000 인 일차원 배열을 이용하여 1000 까지의 소수를 구하는 프로그램을 작성하라. 방법은 다음과 같다.

배열을 0 부터 999 까지 초기화 한다.

```
    for(i=0;i<1000;i++)
        a[i]=i;
```

2 는 소수이다.
    4 부터 시작하여 2 보다 큰 2 의 배수를 모두 지운다.
    a[]배열에 0 을 넣으므로써 지운다.
    4, 6, 8, 10, 12, 14, ,,, 등을 지운다.
    2 에서 다음 칸으로 가면서 지워지지 않은 수를 찾는다.
    3 이 지워 지지 않았다.
3 은 소수이다.
    6 부터 시작하여 3 보다 큰 3 의 배수를 모두 지운다.
    6, 9, 12, 15, 18, 21, ,,, 등을 지운다.
    3 에서 다음 칸으로 가면서 지워지지 않은 수를 찾는다.
    5 가 지워 지지 않았다.
5 는 소수이다.
    10 부터 시작하여 5 보다 큰 5 의 배수를 모두 지운다.
    10, 15, 20, 25, 30, 35, ,,, 등을 지운다.
    5 에서 다음칸으로 가면서 지워지지 않은 수를 찾는다.
    7 이 지워 지지 않았다.
7 은 소수이다.
    14 부터 시작하여 7 보다 큰 7 의 배수를 모두 지운다.
    14, 21, 28, 35, 42, 27, ,,, 등을 지운다.
    7 에서 다음칸으로 가면서 지워지지 않은 수를 찾는다.
    11 이 지워 지지 않았다.
11 은 소수이다.
.....

이러한 순서로 지워나가면 남아있는 수가 모두 소수이다. 마지막으로 2 부터 997 까지의 소수를 순서대로 찍는다.

배수를 구할 때는 덧셈만으로 가능하기 때문에 곱셈을 쓰지 않도록 프로그램을 작성하도록 하자.

3. 소인수 분해를 이용한 소수 구하기

크기가 1000 인 배열을 이용하여 1000 개의 소수를 구하는 프로그램을 작성해 보자. 방법은 다음과 같다.

2 를 제외한 소수는 모두 홀수이므로 3 이상의 홀수들만 조사하여 소수를 구한다. 홀수는 자신보다 작은 소수로 나누어 떨어지지 않으면 소수가 된다.

예를 들면 17 이 소수인지 알기 위하여 3, 5, 7, 11, 13 으로 모두 나누어 보면 되는 것이다. 하지만 조금만 생각한다면 5 로 나누어 볼 필요가 없다는 사실을 알 수 있을 것이다. 5 로 나누어 떨어 진다면 그 몫이 5 보다 클

때에만 고려하면 될 것이다. 만약 몫이 5 보다 작다면 3 이어야 하는 데 이는 이미 3 으로 나누어 볼 때 고려되었기 때문이다.

먼저 a0=2; a1=3; 을 넣어둔다.

```
    5 부터 출발하여 모든 홀수를 검사한다.
    5 가 3 으로 나누어 떨어지는지 검사한다. 검사하기 전에
        3 을 제곱하면 9 이다. 9 는 5 보다 크다
        따라서 5 는 소수이다. a[2]=5;
    7 이 3 으로 나누어 떨어지는지 검사한다. 검사하기 전에
        3 을 제곱하면 9 이다. 9 는 7 보다 크다
        따라서 7 은 소수이다. a[3]=7
    9 가 3 으로 나누어 떨어지는지 검사한다. 검사하기 전에
        3 을 제곱하면 9 이다. 9 는 9 와 같다
        따라서 9 는 소수가 아니다.
    11 이 3 으로 나누어 떨어지는지 검사한다. 검사하기 전에
        3 을 제곱하면 9 이다. 9 는 11 보다 작다
        11 이 3 으로 나누어 떨어지는지 검사한다.
        나누어 떨어지지 않는다.
    11 이 5 로 나누어 떨어지는지 검사한다. 검사하기 전에
        5 를 제곱하면 25 이다. 25 는 11 보다 크다.
        따라서 11 은 소수이다. a[4]=11
    13 이 3 으로 나누어 떨어지는지 검사한다. 검사하기 전에
        3 을 제곱하면 9 이다. 9 는 13 보다 작다.
        13 이 3 으로 나누어 떨어지는지 검사한다.
    13 이 5 로 나누어 떨어지는지 검사한다. 검사하기 전에
        5 를 제곱하면 25 이다. 25 는 13 보다 크다.
        따라서 13 은 소수이다. a[5]=13
    15 가 3 으로 나누어 떨어지는지 검사한다. 검사하기 전에
        3 을 제곱하면 9 이다. 9 는 15 보다 작다.
        15 는 3 으로 나누어 떨어진다. 소수가 아니다.
    17 이 3 으로 나누어 떨어지는지 검사한다. 검사하기 전에
        3 을 제곱하면 9 이다. 9 는 17 보다 작다.
        17 은 3 으로 나누어 떨어지지 않는다.
    17 이 5 로 나누어 떨어지는지 검사한다. 검사하기 전에
        5 를 제곱하면 25 이다. 25 는 17 보다 크다.
        따라서 17 은 소수이다. a[6]=17
    19 가 3 으로 나누어 떨어지는지 검사한다.
      . . . . .
```

이런 방법으로 소수 1000 개를 구하여 출력하는 프로그램을 작성하라. 나누어 떨어지는지를 알아내기 위하여 %연산자를 사용하면 된다.

Chapter 10. 파일 입출력을 시작하며

지금까지 우리는 멀고먼 길을 왔다. 초보자를 면하기가 이렇게 어려울줄 몰랐다. 그러나 아직 당신은 초보자를 진정 면한게 아니다. 그동안 미루어 왔던 몇가지 숙제들이 있기때문이다.

여기 프로그래머의 길에서는 프로그래밍의 실력과 언어에대한 이해가 서로 조화되어 상호 보조작용을 할 수 있도록 남들이 택하지 않는 길을 택하여 왔다. 그러다보니 벌써 알고 있어야 하는 것들도 미루는 것이 더 좋다고 여겨져서 미루어 온것들이 여러가지 있는 것이다. 하지만 프로그래머의 길을 충실히 따라온 여러분은 이미 상당한 실력을 갖추었다. 따라서 미루어 놓았던 숙제도 바로 마칠 수 있으리라 기대한다.

1. 다음 프로그램을 실행시켜 보도록 하자.

```
#include<stdio.h>
int main()
{
    FILE *fin;
    char finname[80];
    int c;

    printf("\nPlease enter File name:");
    scanf("%s", finname);

    fin = fopen(finname, "r");

    c = fgetc(fin);
    while(c != EOF) {
        printf("%c", c);
        c = fgetc(fin);
    }
    fclose(fin);
    return 0;
}
```

프로그램을 실행시키면 File 이름을 입력하라고 할것이다. 그때 현재 directory 에 있는 적당한 C 프로그램 파일이름을 주도록 하자.

그러면 이 프로그램은 파일이름을 입력해준 바로 그 C 프로그램을 화면에 출력하게 될 것이다.

지금부터 여러분은 프로그램의 동작에 관련된 여러가지 지식을 습득해야 한다. 그중 한가지가 파일의 개념인 것이다. 우선 시작하기 위해서 파일 이란 외부에 있는 문자의 배열이라고 생각하자. 프로그램 밖에 매우 긴 문자의 배열이 존재하고 있다. 이것을 한 문자씩 가져올 수 있게 하는게 파일이라고 일단 이해하면 되겠다.

우리는 다음과 같이 그 배열에 있는 문자들을 하나씩 프로그램 내부로 가져올 수있다.

c = fgetc(fin); 문이 바로 그것을 나타낸다.

비록 읽어 들어 오는 값이 문자이지만 값을 받을 때는 정수로 받는다. 정수로 문자값을 받는데는 그에 합당한 이유가 있기 때문이다. 여기서 알아두자 "함부로 문자형으로 값을 받는 일"을 하지 않도록 한다. 다시한번 강조하지만 c 의 자료형을 char 형으로 바꾸지 않아야 한다.

밖에서 입력을 받아 들이다 보면 여러가지로 특별한 경우가 발생할 수 있다. 그중 한가지가 파일의 끝에 다달았을 경우이다. 프로그램에서는 자꾸 다음 문자를 달라고 하는데 이미 끝에 왔기 때문에 더이상 줄게 없다. 이런 경우를 나타내기 위하여 특별한 값 EOF 를 되돌려 주는 것이다.

이 EOF 값은 문자가 나타내는 범위를 벗어난 값을 가져야 한다. 그래야 정상적인 입력 문자와 구분할 수 있기 때문이다. 따라서 c 의 형은 정수형 이어야 한다. 정수형은 문자형 보다 더 넓은 범위를 나타낼 수 있기 때문이다.

한가지 더 이야기 하고 넘어가도록 하자.

위의 프로그램에서 보여준 한문자를 먼저읽고 while 문내에서 읽은 문자를 처리(출력)한다음 맨 밑에서 다음 문자를 읽어 들이는 구조는 매우 중요하다.

다시 보도록 하자.

```
    c = fgetc(fin);
    while(c != EOF) {
        printf("%c", c);
        c = fgetc(fin);
    }
```

이 제어구조를 반드시 기억하도록 하자. 최소한 프로그래머의 길에서 프로그램을 익힌 사람이라면 이 제어구조가 가지는 심오한 뜻을 얼마 지나지 않아서 알게 될 것이다. 위의 제어구조는 프로그램의 여러 곳에서 심심치 않게

튀어나오는 아주 보편적인 제어구조인 만큼 이에 대해서 다시 말하는 기회가 오더라도 잔소리로 여기지 말기를 부탁드린다.

==========

파일 입출력과 문자

파일은 프로그램 밖에 있는 문자의 배열로 생각하자고 했다. 우리가 알고있는 문자라는 것은 화면상에서 표시되는 것만을 의미한다. 그러나 파일에서 문자라고 하는 것은 화면상에서 표시할 수 있는 문자만을 의미하지는 않는다. 문자중에는 화면에 표시할 수 없는 문자도 많이 있다. 이에 대해서 자세히 알아보기로 하자. C 에서 char 형은 1 바이트의 크기를 가지며 그값의 범위는 -128 에서 127 사이의 값을 가지는 정수값과 같다. 부호를 빼고 0 에서 255 사이의 값을 가지는 정수로 취급하고자 할 때에는 unsigned char 형을 택해 주어야 한다. 우리가 PC 에서 보통 사용하는 문자라고 부르는 것들은 ASCII 코드 로된 문자를 나타낸다. 예를 들어서 설명해 보자. 알파벳 중 A 는 65 라는 숫자와는 직접적인 관계가 없다. 그러나 컴퓨터 내부에서 A 를 나타내는 방법으로 65 라는 값을 가지는 한 바이트를 A 라고 정하여 사용하는 것이다. 마찬가지로 B 는 66 라는 값을 가지며 C 는 67 이라는 값을 가진다. 이것을 정한 사람들이 있고 우리는 그대로 따라서 사용하고 있는 것이다. 우리는 A 의 ASCII 코드값이 65, B 의 ASCII 코드값이 66, C 의 ASCII 코드값이 67 이다 라고 말하곤 한다.

다음 예를 보자.

```
#include<stdio.h>
int main()
{
    unsigned char a, b, c;

    a = 'A';
    b = 'B';
    c = 'C';

    printf("\n%c %c %c ", a, b, c);
    return 0;
}
```

위의 프로그램의 출력은

```
    A B C  이다.

    printf("\n%d %d %d ", a, b, c);로 바꾸어 출력을 하면
    65 66 67 이 출력된다.
```

다음 예를 보자.

```
#include<stdio.h>
int main()
{
    unsigned char a, b, c;

    a = 65;
    b = 66;
    c = 67;

    printf("\n%c %c %c ", a, b, c);
    return 0;
}
```

위 프로그램의 출력도 A B C 이다.

지금까지 몇가지 예를 통해 본것 처럼 문자는 범위를 가진 정수로 생각할 수 있다. 따라서 계산을 하는데 사용하거나 배열의 인덱스로 사용할 수도 있는 것이다. 지금 이야기 한것을 잘 기억하고 있도록 하자.

==========

다음 예를 포함한 파일입출력은 Windows 하에서 적용되는 경우를 다루고 있다. 대부분의 파일입출력에 관한 것은 Linux 나 Windows 나 가리지 않고 적용되는 것이지만 줄바꿈에 관한 것은 차이가 있다. 일단 Windows 하의 텍스트 파일에서 적용되는 것이라고 알아두자.

1. 다음 프로그램을 실행 시켜보자.

```
#include<stdio.h>
int main()
{
    FILE *fin, *fout;
    char  finname[80];
    char  foutname[80];
    int   c;
```

```
    printf("\nPlease enter input file name:");
    scanf("%s", finname);
    printf("\nPlease enter output file name:");
    scanf("%s", foutname);

    fin = fopen(finname, "rb");
    if(fin==NULL) {
        printf("\nCannot open file:%s", finname);
        exit(1);
    }
    fout = fopen(foutname, "wb");
    if(fout==NULL) {
        printf("\nCannot open file:%s", foutname);
        exit(1);
    }

    c = fgetc(fin);
    while(c != EOF) {
        fputc(c, fout);
        c = fgetc(fin);
    }

    fclose(fin);
    fclose(fout);
    return 0;
}
```

위의 프로그램을 실행 시키면 입력 파일이름을 넣으라는 요청이 나온다. 이때 먼저번의 프로그램 때와 마찬가지로 현재 디렉토리 에 있는 적당한 C 프로그램 파일 이름을 주도록 하자. 다시 출력파일 이름을 넣으라는 요청이 나온다. 이때 출력을 위한 다른 파일이름을 주도록 하자. 이미 있던 파일을 보존하려면 다른 이름을 주어야 한다.

위의 프로그램은 실행이 제대로 되고 나면 원래 있던 파일을 새로 준 파일이름으로 복사하는 프로그램이 되겠다.

지금 프로그램은 먼저번의 예와는 몇가지 점에서 차이가 있다. 우선 파일을 2 개 사용하고 있다는 점이다. 한개는 입력 파일로 사용하고 한개는 출력파일로 사용하고 있다.

또 fopen 문에서 fin = fopen(finname, "rb"); 에서 "rb" 라는 옵션을 사용하고 있다. rb 는 입력용 바이너리 파일이라는 속성을 나타낸다. 바이너리 파일에 대응되는 파일은 텍스트 파일이다. 텍스트 파일이란 우리가 눈으로 보고 읽을 수 있는 텍스트와 제어 문자를 포함한 파일이며, 바이너리 파일이란 텍스트 파일이 아닌 다른 파일들을 통털어 이야기 한다.

벌써 어려운 이야기가 나오고 있다. 그러나 어쩌랴 잘 소화가 되지 않더라도 듣고 넘어가도록 하자.

우리가 작성한 C 프로그램은 보통 텍스트파일로 되어있다. 그래서 파일을 열때 텍스트 파일로 여는것이 보통이다. 하지만 텍스트 파일도 바이너리 파일로 열수가 있다. 바이너리 파일로 열었을 때는 입력을 한문자씩 할때 한개의 문자도 빠뜨리지 않고 우리가 받아서 처리할 수가 있다. 그러나 텍스트 파일로 열었을 때는 문자를 받을 때 원래의 파일에는 들어 있지만 그냥 무시하고 넘어가서 우리에게 주지 않는 문자가 있다. 줄바꿈을 위해서 2 개의 문자가 쓰이고 있다. 그러나 텍스트 파일로 열면 한개의 문자만을 우리에게 전해 준다.

==========

줄바꿈에 대하여

```
    23 69 6E 63 6C 75 64 65   3C 73 74 64 69 6F 2E 68
    #  i  n  c  l  u  d  e    <  s  t  d  i  o  .  h
    3E 0D 0A 69 6E 74 20 6D   61 69 6E 28 29 0D 0A 7B
    >        i  n  t     m    a  i  n  (  )        {
    0D 0A 20 20 20 20 69 6E   74 20 69 2C 20 6E 3B 0D
                     i  n    t     i  ,     n  ;
    0A . . . . . . . . . . .
```

위의 16 진수들은 다음과 같은 파일을 한문자씩 읽어 들일때 의 각각의 문자를 나타낸다.

```
#include<stdio.h>
int main()
{
    int i, n;
. . . . . .
```

첫번째 줄바꿈인 <stdio.h> 다음을 보면 0D 0A 두개의 문자가 있음을 볼수 가있다. 마찬가지로 main() 다음에도 0D 0A 두개의 문자가 나온다. int i, n; 다음에도 0D 0A 두개의 문자가 나옴 을 알수가 있다. 이 것이 우리가 파일을 바이너리 파일로 열었을때 볼수 있는 텍스트 파일의 줄바꿈이다.

그런데 우리가 파일을 텍스트파일로 열었을때는 0D 문자는 우리에게 보이지가 않는다. 중간에 있는 무었인가가 우리에게 0D 문자를 보여 주지 않는 것이다.

자 다음프로그램을 돌려 보도록 하자.

```
#include<stdio.h>
int main()
{
    FILE *fin;
    char  fname[80];
    int   c;

    printf("\nPlease enter input file name:");
    scanf("%s", fname);

    fin = fopen(fname, "rb");
    if(fin==NULL) {
        printf("\nCannot open file:%s", fname);
        exit(1);
    }

    c = fgetc(fin);
    while(c != EOF) {
        printf("%02X  ", c);
        c = fgetc(fin);
    }
    fclose(fin);
    return 0;
}
```

파일의 길이가 너무 크지 않은 파일을 주고 돌려 보도록 하자. 그러면 한 화면안에 출력이 넘치지 않기 때문에 결과를 쉽게 확인할 수 있을 것이다.

주의할 점은 "%02X " 에서 빈칸을 2 칸으로 맞추어주어야 출력의 모양이 반듯하게 나올것이다.

위의 프로그램에서 fin = fopen(fname, "r"); 와 같이 텍스트 파일로 열게 되면 줄바꿈이 달라지는 것을 쉽게 알수 있을 것이다.

이렇게 줄바꿈에 대해서 길게 이야기 하는 것은 줄바꿈을 이용한 문제를 내기 위해서 했다고 보면 되겠다.

자 파일 입출력 문제를 한번 풀어보기로 하자.

Chapter 11. 파일 입출력 과제모음

문 4-1) 파일을 읽어서 한 줄씩 뒤집어 출력하는 프로그램을 작성 하시오. (일차원 배열을 이용하여 한줄씩 보관했다가 뒤집어서 출력하시오)

```
    >h.oidts<edulcni#
    )(niam tni
    {
    ;n, j, i tni
    . . . . . .
```

문 4-2) 파일을 읽어서 아래와 같은 형식으로 출력하는 프로그램을 작성 하시오. (이차원 배열을 이용하여 모두 보관했다가 출력하시오)

```
   . . . . . .
   . . . . . .
   . . . . . .
      int i, j, n;
   {
   int main()
   #include<stdio.h>
```

문 4-3) 파일을 읽어서 아래와 같은 형식으로 출력하는 프로그램을 작성 하시오. (이차원 배열을 이용하여 모두 보관했다가 출력하시오)

```
     #  i  {         . . . . . .
     i  n            . . . . . .
     n  t            . . . . . .
     c               . . . . . .
     l  m      i     . . . . . .
     u  a      n     . . . . . .
     d  i      t     . . . . . .
     e  n            . . . . . .
     <  (      i     . . . . . .
     s  )      ,     . . . . . .
     t               . . . . . .
     d         j     . . . . . .
     i         ,     . . . . . .
     o               . . . . . .
     .         n     . . . . . .
     h         ;     . . . . . .
     >               . . . . . .
     . . . . . .
```

문 4-4) 두 개의 파일을 읽어서 하나의 파일로 출력하는 프로그램을 작성하시오. 단 줄단위로 합쳐서 한줄씩으로 만들도록 하시오. 파일을 구분하기 위해서 각 파일의 줄과 줄 사이에는 $$$를 넣으시오. (일차원 배열을 이용하여 한줄씩 보관했다가 출력하시오)

```
    #include<stdio.h>$$$#include<stdio.h>
    int main()$$$int main()
    {$$${
    int i, j, k;$$$int a[10]={0,};
    . . . . . .
    . . . . . .
```

문 4-5) 문제 4-4 번의 문제 중 두 번째 파일에 해당하는 문자열을 뒤집어 출력하는 프로그램을 작성 하시오.

```
    #include<stido.h>$$$>h.oidts<edulcni#
    int main()$$$)(niam tni
    {$$${
    int i, j, k;$$$;},0{=]01[a tni
    . . . . . .
    . . . . . .
```

문 4-6) 임의의 파일을 읽어서 hexa 값으로 표현하는 프로그램을 작성 하시오. (크기가 16 인 일차원 배열한개만 사용하고 입력파일은 한개만 사용하여 문제를 풀도록 한다.) 출력방식은 다음과 같이 하도록 한다.

1. 처음 6 자리에는 상대적인 위치를 표시한다.
2. 다음에 16 문자씩 잘라서 출력을 한다.
3. 처음에는 헥사값을 찍고 다음에는 아스키문자로 찍는다.
4. 8 개의 헥사값을 찍고는 한칸을 더 띄운다.
5. 헥사값과 아스키문자 사이에는 3 칸을 띄운다.
6. 찍을 수 없는 문자는 .으로 표시한다.
   (찍을수 있는 문자인지 아닌지를 알기위해서 isprint 함수를 사용한다.)
7. 마지막 줄처리를 아래에서 보이는 것처럼 출력을 한다.

```
000000 23 69 6E 63 6C 75 64 65  3C 73 74 64 69 6F 2E 68   #include<stdio.h
```

```
000010 3E 0D 0A 69 6E 74 20 6D  61 69 6E 28 29 0D 0A 7B   >..int main()..{
000020 0D 0A 20 20 20 69 6E 74  20 69 2C 6A 2C 6E 2C 6B   ..   int i,j,n,k
000030 2C 6C 3B 0D 0A 20 20 20  69 6E 74 20 61 5B 33 30   ,l;..   int a[30
000040 5D 5B 33 30 5D 3D 7B 30  2C 7D 3B 0D 0A 0D 0A 20   ][30]={0,};.....
000050 20 20 70 72 69 6E 74 66  28 22 6E 75 6D 62 65 72   printf("number
000060 3D 22 29 3B 0D 0A 0D 0A  20 20 20 73 63 61 6E 66   =");....   scanf
000070 28 22 25 64 22 2C 26 6E  29 3B 0D 0A 0D 0A 20 20   ("%d",&n);.....
000080 20 66 6F 72 28 69 3D 30  3B 69 3C 6E 3B 69 2B 2B    for(i=0;i<n;i++
000090 29 0D 0A 20 20 20 7B 0D  0A 20 20 20 20 20 20 66   )..   {..      f
0000A0 6F 72 28 6A 3D 30 3B 6A  3C 6E 3B 6A 2B 2B 29 0D   or(j=0;j<n;j++).
0000B0 0A 20 20 20 20 20 20 7B  0D 0A 09 09 61 5B 69 5D   .      {....a[i]
     .     .    .    .     .    .    .    .
000230 20 20 20 7D 0D 0A 0D 0A  20 20 20 72 65 74 75 72      }....   retur
000240 6E 20 30 3B 0D 0A 7D 0D  0A                        n 0;..}..
```

파일 입출력을 돕기 위해서 도움이 될 수 있도록 다음 프로그램을 보자.

```
#include<stdio.h>
int main()
{
    FILE *fin;
    char fname[80];
    unsigned char line[100];
    int  c;

    printf("\nPlease enter file name:");
    scanf("%s", fname);

    fin = fopen(fname, "rb");
    if(fin == NULL) {
        printf("\nCannot open file: %s", fname);
        exit(1);
    }

    c = fgetc(fin);
    while(c ! = EOF) {
        if(c != 13) {
            배열에 넣는다
        }
        else {
             13 이다. 즉 헥사로 0D 문자이다.
             줄바꿈을 처리하기 위해서 0D 문자를 버려야한다.
             마찬가지로 0A 문자도 읽어서 버려야 한다.
             다음줄을 읽을 준비가 되었다.
             줄의 끝에서 해야 할일을 한다.
        }
        c = fgetc(fin);
    }

    파일의 맨끝에 도달하였다.
    줄바꿈이 없이 끝나는 경우가 있음을 알아야 한다.
    이런 경우에는 보관해 두었던 한줄을 처리 하지 않은 채로
    while 문 밖으로 빠져 나왔다. 뒷처리를 해야 한다.

    파일 입출력 뿐아니라 대부분의 경우 이와 같은 경우가
    많이 발생하므로 이 제어구조를 알고 있도록 한다.

    fclose(fin);
    return 0;
}
    다시 간추려서 이야기 해 보도록 하자

    c = fgetc(fin);
    while(c != EOF) {
        읽은 문자를 처리한다.
        c = fgetc(fin);
    }
    뒤처리를 한다.

    읽은 문자처리는 다음과 같다.

    if(c != 13) {
        배열에 보관한다.
```

```
    }
    else {
        13 (헥사문자 0D) 을 만났다. 뒤에 따라나오는 0A 문자를
        처리해야 한다. 그래야 다음줄의 시작을 읽을 수 있다.
        배열에 보관했던 문자들을 출력한다.
    }
```

두개의 파일을 읽는 경우를 생각해 보자.

```
#include<stdio.h>
int main()
{
    FILE *fin1, fin2;
    char fname1[80], fname2[80];
    unsigned char line[100];
    int  c, d;

    printf("\nPlease enter file name:");
    scanf("%s", fname1);
    printf("\nPlease enter file name:");
    scanf("%s", fname2);

    fin1 = fopen(fname1, "rb");
    if(fin1 == NULL) {
        printf("\nCannot open file: %s", fname1);
        exit(1);
    }

    fin2 = fopen(fname1, "rb");
    if(fin2 == NULL) {
        printf("\nCannot open file: %s", fname2);
        exit(1);
    }

    c = 0;
    d = 0;
    while((c != EOF) && (d != EOF)) {
        c = fgetc(fin1);
        while((c ! = EOF)&&(c != 13)) {
            배열에 넣는다
            c = fgetc(fin1);
        }
        // while loop 를 빠져 나왔다. EOF 가 아니면 13 을 만난 것이다.
        if(c != EOF) {
             13 이다. 즉 헥사로 0D 문자이다.
             줄바꿈을 처리하기 위해서 0D 문자를 버려야한다.
             마찬가지로 0A 문자도 읽어서 버려야 한다.
             다음줄을 읽을 준비가 되었다.
             줄의 끝에서 해야 할일을 한다.
        }
        else {
            // 보관한 줄이 있을 때는 줄의 끝에서 해야 할일을 한다.
        }

        d = fgetc(fin2);
        while((2 ! = EOF)&&(2 != 13)) {
            배열에 넣는다
            d = fgetc(fin2);
        }
        // while loop 를 빠져 나왔다. EOF 가 아니면 13 을 만난 것이다.
        if(c != EOF) {
             13 이다. 즉 헥사로 0D 문자이다.
             줄바꿈을 처리하기 위해서 0D 문자를 버려야한다.
             마찬가지로 0A 문자도 읽어서 버려야 한다.
             다음줄을 읽을 준비가 되었다.
             줄의 끝에서 해야 할일을 한다.
        }
        else {
            // 보관한 줄이 있을 때는 줄의 끝에서 해야 할일을 한다.
        }

    }
```

```
    파일의 맨끝에 도달하였다. file1 이나 file2 의 끝에 도달한 것이다.

    남은 파일에 대해서 처리를 해주도록 한다. 위의 구조가 거의 반복해서
    나올 것이다.

    fclose(fin1);
    fclose(fin2);
    return 0;
}
```

Chapter 12. 함수에 관하여

이제 미루워 왔던 두번째 숙제인 함수에 대해서 알아보도록 하자.

우리는 지금까지 작성한 프로그램에서 함수를 작성하지 않고 main 함수 만을 사용해서 프로그램을 작성하는 방식으로 프로그램을 짜 왔다. 그러나 조금 프로그램작성 능력이 생기고 나면 자연스럽게 함수를 만들어서 프로그램을 기능별 단위로 쪼개고 싶은 필요성이 생기게 된다. 여러분은 지금 그러한 단계에 와 있다고 믿는다.

무엇이든지 자기가 필요하다고 느꼈을때 주는 것이 아주 고맙게 여겨지는 것인데, 이런 이유로 함수를 소개하는 것을 지금까지 미루워 왔다.

함수는 우리가 main 함수를 작성했던 것과 마찬가지로 작성하면 된다. 단지 차이가 있다면, 우리가 작성한 main 함수에는 인수가 없는 경우 의 프로그램 예제만 보아왔다. 사용자가 작성하여 사용하는 함수는 보통 인수를 필요로 하며 이에 대해서 제대로 이해하는 것이 중요하다.

다음 예를 보도록 하자.

```
#include<stdio.h>
// oneline 함수를 정의하는 곳
int oneline(int n)
{
    int i;

    for(i=0;i<n;i++)
        printf("%4d", i+1);
    return 0;
}

int main()
{
    int i, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++) {
        oneline(i+1);  // oneline 함수의 사용
        printf("\n");
    }
    return 0;
}
```

위의 프로그램은 다음 프로그램과 동일한 일을 하는 프로그램이다.

```
#include<stdio.h>
int main()
{
    int i, j, n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    for(i=0;i<n;i++) {
        for(j=0;j<i+1;j++)
            printf("%4d", j+1);
        printf("\n");
    }
    return 0;
}
```

위의 프로그램에서 함수 oneline 을 사용한 것은 자연스러운 경우는 아니며 단지 함수가 이러한 것이라는 것을 보여주기 위한 것이다.

원래 함수란 수학에서 사용하듯이 y = f(x) 의 경우와 같이 주워진 값 x 로부터 함수를 적용하여 나온 결과값 y 를 만들어 내는 것을 의미한다. 따라서 함수란 결과적으로 만들어 지는 것만 의미가 있고 결과를 만들어 내는 과정은 중요하지 않아야 한다고 생각할 수도 있다. 그러나 위의 경우에 보듯이 oneline 이라는 함수가 돌려준 결과값 0 는 별로 의미가 없고 그 과정에서 화면에 출력한 숫자들이 의미를 가진다.

따라서 C 언어에 있어서 함수란 결과값 뿐아니라 중간에 생긴 부수효과 들도 중요한 역할을 할 수가 있다.

다음 프로그램을 보자.

```
#include<stdio.h>

int factorial(int n)
```

```
{
    int i, fact;

    fact = 1;
    for(i=2;i<=n;i++)
        fact = i*fact;
    return fact;
}

int square(int n)
{
    return n*n;
}

int main()
{
    int n;

    printf("\nPlease enter number:");
    scanf("%d", &n);

    printf("\nnumber=%4d factorial = %4d square = %4d ",
        n, factorial(n), square(n));
    return 0;
}
```

위의 프로그램에서 우리는 두개의 함수 factorial 과 square 를 사용하였다. 위의 경우에는 결과값이 의미가 있는 경우가 되겠다.

Chapter 13. 초보를 마치면서 풀어보는 과제모음

이제 함수의 사용까지 공부함으로써 우리는 초보과정을 마쳤다. 모두 지금까지 열심히 온 것을 스스로 축하하면서 이제 중급과정으로 넘어가도록 하자. 그전에 초보과정을 마친 사람들을 위한 조금 어려운 과제들을 풀어보도록 하자.

아래 프로그램 들은 Turbo C 에서 작성하여 돌리는 것을 가정하고 낸 문제들이다.

```
      #
  #### #
     # #
     # ##
     # #
       #
```

1. 위와 같은 글자를 출력하는 프로그램을 작성하시오. (2 차원 배열에 필요한 정보를 넣어서 초기화 한다음 출력할 것)

2. 위의 '가'자를 문자형 일차원배열 6 에 넣어서 같은 방법으로 출력할 것. 한줄에 필요한 정보는 7 가지 위치에 따라 #가 있느냐 없느냐를 구별해야 한다. 이것을 한문자가 8 비트 이기 때문에 모두 표시할 수있다.

3 "가"글자의 모양을 좌,우 스크롤 하는 프로그램을 작성 하시오. 키보드 방향키(좌화살표키, 우화살표키)에 따라 좌우 스크롤 하는 프로그램을 작성. 스크롤이란 같은장소에서 회전하는 것을 의미한다.

```
1 2 3 4 5   가

2 3 4 5 1   이 되었다가

3 4 5 1 2   로 변해가는 것을 의미한다.
```

4. 일차원 배열에 저장된 '＼', '|', '/' '-' 의 문자가 회전하는 프로그램을 작성하시오.

5. 위의 문제에서 속도및 회전 방향을 조절하는 프로그램을 작성하시오. 속도의 증가 를위해서 위화살표키, 감소를 위해서 아래화살표키, 방향의 전환을 위해서 우화살표키, 좌화살표키를 사용하도록 하시오.

6. 만년달력 프로그램을 작성하시오.

7. 세자리수 야구 게임을 작성하시오.

8. 팝업 메뉴 프로그램을 작성하시오.

9. 사다리 게임을 위한 프로그램을 작성하시오.

10. 오델로 게임 프로그램을 작성하시오.

11. 지뢰찾기 게임 프로그램을 작성하시오.

12. 푸쉬푸쉬 게임 프로그램을 작성하시오.

Chapter 14. 중급으로 향하는 길-C/C++, 자료구조 방향

초보를 뛰어넘은 당신은 이제 프로그래밍에 대해서 어느정도 자신이 붙었다. 그러나 여전히 무엇을 어 떻게 배워나가야 하는지 어떤 목표를 가지고 공부 해야 하는지 잘 모르고 있다.

우선 파일입출력에 어느정도 익숙해질 필요가 있다. 그리고 정렬알고리즘들을 배우고 구현해 보아야 한다. 보통 초보자 시절에 정렬알고리즘을 배우고 구현해 보는 경우가 많은데, 필자는 초보자들에게 정렬 알고 리즘을 구현하라는 것은 조금 무리라고 생각하고 있다. 그러나 초보를 뛰어넘은 사람들, 특히 프로그래머의 길에서 초보를 뛰어넘은 사람들은 정렬 알고리즘들을 구현하는데에 큰 어려움을 느끼지 않을 것이다.

바블정렬, 선택정렬, 삽입정렬, 등은 비교적 쉽게 이해하고 구현할 수가 있다. 단지 예외가 있다면 퀵정렬이 되겠다. 이 퀵정렬은 재귀호출(순환호출)을 사용하기 때문에 재귀함수의 사용에 대해서 익숙해 지지 않으면 아주 큰 어려움을 주는 장애물이 되겠다. 도전문제에서 기술한대로 하노이탑문제를 종이카드를 이용해서 익숙하도록 풀어보고 또 프로그램으로 구현 도 해보도록 하자.

당신이 공부해야할 것들을 한번 살펴보자. 당신은 이제 여러가지를 알아야 한다. C 언어에서 빠뜨려온 구조체와 포인터를 제대로 알아야 한다. 그래야 자료구조를 제대로 공부하고 이해하고 구현해 볼수 있는 것이다. 그렇다. 구조체와 포인터를 알아야 하는 이유는 프로그램내에서 복잡한 일을 하려고 할때 반드시 알아야 하는 자료구조를 구현할 수 있는 실력을 키우기 위해서인 것이다.

그런다음 C 로 배우는 알고리즘 (이재규지음)책등을 통해서 자료구조를 제대로 배우는 것이 좋겠다. 자료구조는 스택,큐

배열,연결리스트를 먼저 배우고 또 이를 구현해 보도록 한다. 배열을 이용한 선형검색과 이분검색을 구현한 다음 트리를 배우고 이진검색나무를 구현하고 그리고 균형잡힌나무(2-3 나무, 2-3-4 나무, 빨강-깜장 나무, B-나무)등을 공부하고 구현해 본다.

그리고 해싱을 공부한다. 해싱은 매우중요한 것으로 이해할 뿐아니라 구현해 보는것이 매우 중요하다. 다소 여유가 있는 사람은 힙정렬도 공부하는것이 좋을 것이다.

이정도면 대개 중급과정을 마쳤다고 볼수 있다. 중급과정을 마친 여러분은 도전문제중 아빠의 퍼즐 문제를 구현할수 있는 실력이 생겼다고 볼수 있다.

Chapter 15. 선택의 기로-어디로 갈것인가?

정렬과 자료구조에 대해서 배우고 구현해본 당신은 이제 현실세계에 보다 가까이 다가가야 한다. 본격적으로 실제 사용되는 프로그램을 작성하는 쪽으로 가야하는 것이다. 즉 프로 프로그래머로서의 준비를 해야하는 것이다.

프로 프로그래머의 길에는 여러가지 방향이 있을 수있다.

Visual Basic 을 주로 사용하는 길이 있다.

Windows API 와 C/C++을 사용하는 길이 있다. 조금 다른길 로는 MFC 를 사용하는 길이 있다.

Java 를 주로 사용하는 길이 있다. C#는 마이크로소프트가 제시하는 길이다.

그외에 많은 다양한 길들이 있고 웹관련 프로그래밍, DB 관련 프로그래밍으로 가는길, 네트워크관련, 하드웨어 제어 관련, 모바일 관련, 등등 길에 따라서 공부하고 익혀야 하는 것들이 다를 수 있다.

다른 무엇보다도 객체지향적인 훈련을 쌓을 필요가 있다. 그동안의 경험과 추세를 보아서 객체지향적인 개발환경에 익숙해져 있지않으면 안된다고 본다. 물론 Visual Basic 프로그램 만으로 충분한 사람도 있고 Power Builder 만으로 충분한 사람들이 있을 것이다. 그러나 큰 흐름은 OOP 훈련을 받아야 한다는 것이다. OOP 개발환경은 대체로 크게 3 가지 로 나눌 수 있겠다. C++ 계열, Java 계열, C#계열 로 볼수 있다. 물론 Delphi 도 있고 다른 객체지향적 언어들도 있다.

필자는 위의 세가지 중에서 Java 를 택해서 훈련을 쌓는것 이 무난하다고 생각한다. 위의 글에서 언급한 중급의 실력을 쌓기 위해서 해야하는 훈련과정, 파일 입출력, 정렬, 구조체-포인터, 자료구조, 연결리스트, 트리, 균형잡힌 나무, 해싱 등을 Java 를 통해서 배우고 익히며 동시에 OOP 적인 훈련을 쌓을수 있다고 생각한다.

물론 중급의 실력을 가진사람이 Java 를 시작해도 무난할 것이라고 보인다. Java 에서 받은 훈련들이 C++이나 C# 에도 거의 대부분이 유효할 것이라고 보이기 때문에 어떤길을 택해서 가야하는가에 대한 고민은 실력이 늘고 나서 해도 늦지 않을 것이라고 본다.

Chapter 16. 정말로 실력을 키우기를 원한다면

앞으로 이곳 프로그래머의 길에 나오는 과제나 도전문제에 대한 답이나 소스들이 인터넷상에 돌아다닐지도 모른다는 불안감이 벌써 들기 시작한다. 충분히 예상되는 일이기는 하다.

그렇다면 당신은 어떻게 할것인가? 문제는 안풀린다. 벌써 1 주일째 이 문제를 붙들고 있다. 짜증이 나기도 하고 열도 난다. 답이 보고 싶다. 정말 보고 싶다.

아니다. 보지 않겠다. 결코 보지 않겠다. 내힘으로 반드시 풀고야 말겠다.

어떤 태도를 가졌느냐에 따라 당신의 앞길이 달라질것이다.

선택은 당신에게 달렸다. 답을 볼것인가 말것인가. 실력있는 프로그래머가 될것인가 말것인가. 당신의 미래,,가