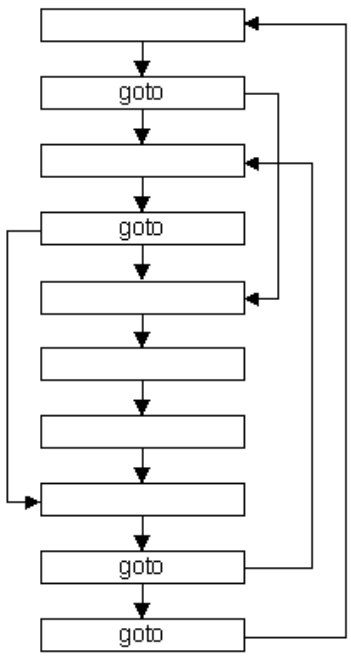
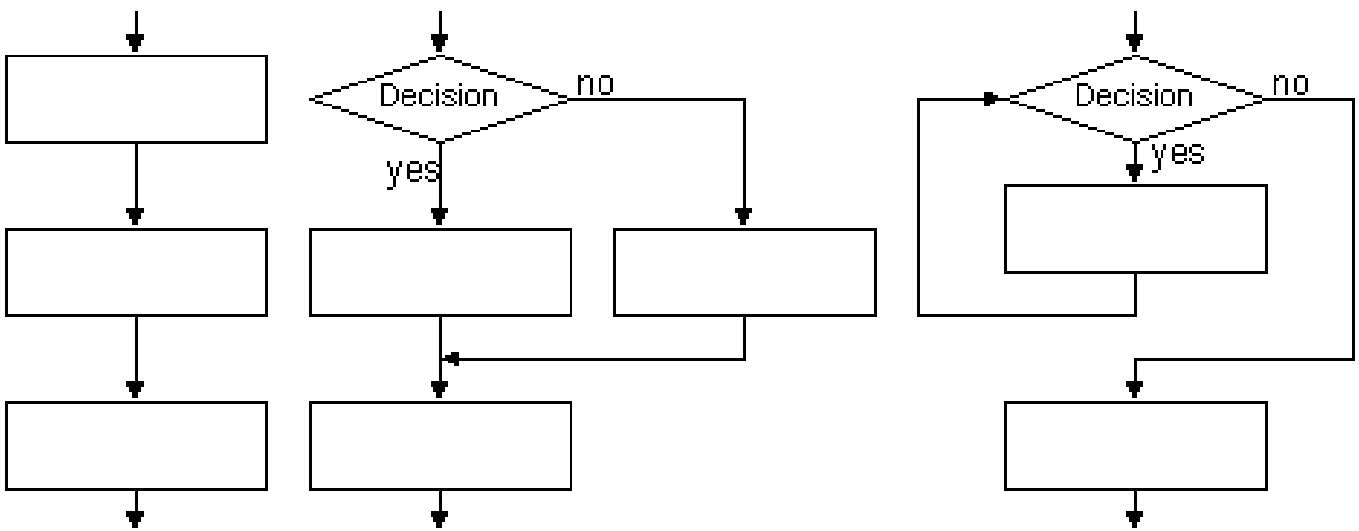
10. 제어구조

- GOTO문이 얼마나 프로그램을 복잡하게 만드는지 살펴 볼 수 있습니다.

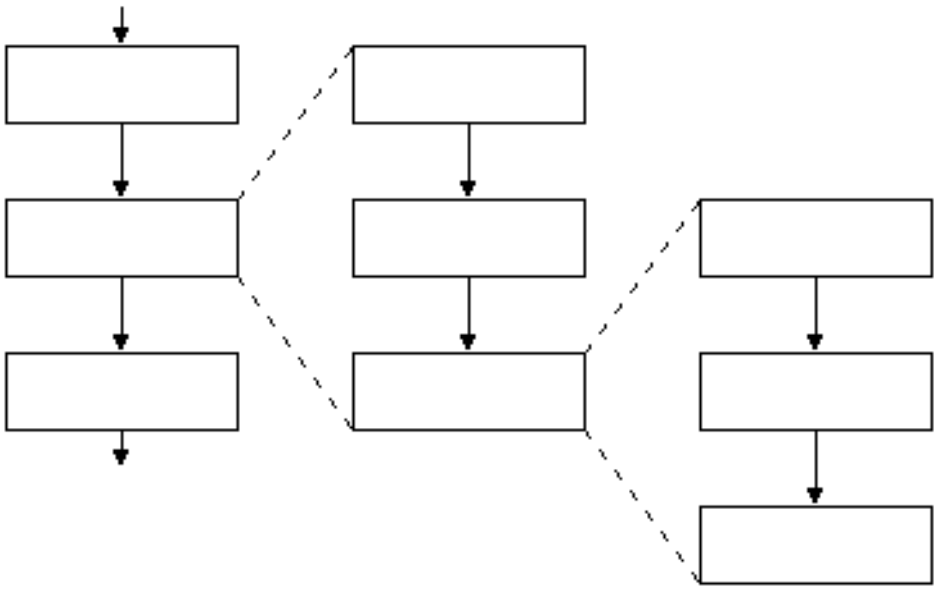


• 대부분의 프로그래밍 언어들 - C, C++ - 은 블록과 제어 구조를 지원하는데, 이처럼 블록과 제어구조를 지원하는 언어를 **구조적 언어(structured programming language)**라고 합니다.

• 구조적 언어는 흐름을 제어하기 위해 3가지의 **흐름 제어 구조(flow control structure)를** 지원합니다. 그것이 ①순차 ②비교 ③반복 구조입니다.



- 순차(sequential) 구조는 직관적(intuitive)입니다.
- 한 실행 단위는 다른 순차 구조를 호출할 수도 있습니다.



순차 구조의 의미: 순차 구조의 한 단위는 블록이며, 이것은 다른 제어구조가 될 수 있습니다.

- 비교를 지원하는 제어문에는 if, switch가 있습니다.
- 반복을 지원하는 제어문에는 for, while과 do가 있습니다.

(1) if문

```
if (표현식1)
        문장1;
[else if (표현식2)
        문장2;][...]
[else
        문장3;]
```

(2) switch문

```
switch (표현식) {
    [case 표현식:
        문장;
        [break;]][...]
    [default:
        문장;
        [break;]]
}
```

(3) for문

```
for ([표현식1]; [표현식2]; [표현식3])
    문장;
```

(4) while문

```
while（표현식）
    문장;
```

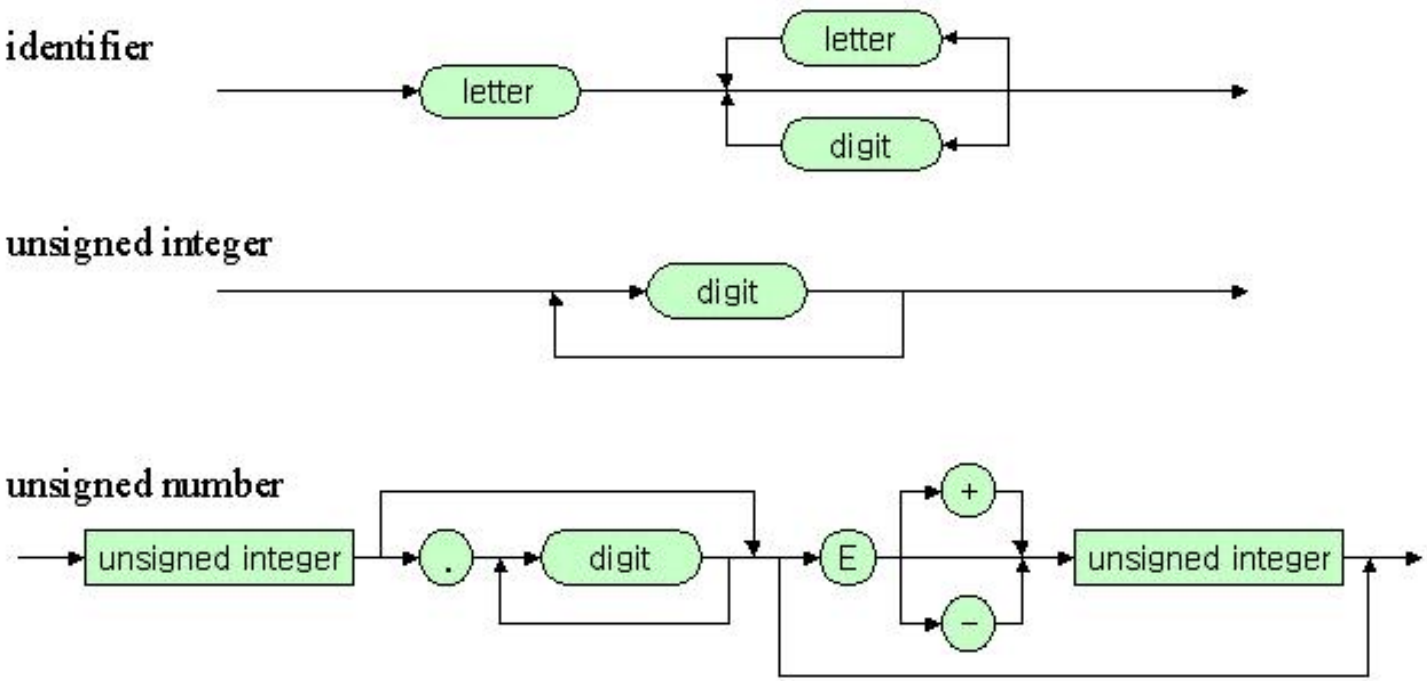
(5) do...while문

```
do
    문장;
while（표현식);
```

if

```
if (표현식1)
        문장1;
[else if (표현식2)
        문장2;][...]
[else
        문장3;]
```

- [와]로 둘러쌓인 부분은 그 부분이 생략될 수 있음(option)을 의미합니다.
- [...]는 앞부분이 0번 이상의 반복 가능함(0 or more repetition)을 의미합니다.



위의 그림은 언어의 문법을 나타내는 도식(diagram)입니다. C++ 언어의 문법은 CFG(context free grammar)를 이용하여 나타낼 수 있습니다. 하지만, 이 그림은 CFG와 대등하며, CFG보다 이해하기 쉽습니다.

- if문을 사용할 때 우리는 아래의 사항들을 주의해야 합니다.

(1) if나 else if에서 명시된 표현식이 2가지 이상이 참이라면 제일 먼저 만난 표현식이 평가되므로, 2번째 이하부터의 문장은 실행되지 않습니다. 즉 여러 개의 조건이 참일지라도 첫 번째 조건만이 실행됩니다.

(2) else가 없는 if문일 경우, 한 문장도 실행되지 않을 수도 있습니다.

(3) else가 있는 if문은 최소한 1문장은 실행됩니다.

```
#include <stdio.h>

void main() {
    int i=2,j=3,k=4;

    if (i<j)
        printf("i is less than j\n");//이 문장이 실행된다.
}//main
```

- 위의 예에서, 단독으로 사용된 if는 조건이 만족되므로, printf()가 실행됩니다.

```
#include <stdio.h>

void main() {
    int i=5,j=3,k=4;

    if (i<j)
        printf("i is less than j\n");
    else
        printf("i is greater or equal than j\n");//0이 문장0이 실행된다.
}//main
```

- 위의 예에서 if의 조건은 거짓(0)이므로, else 의 printf()가 실행됩니다.

```
#include <stdio.h>

void main() {
    int i=5,j=4,k=3;

    if (i<j)
        printf("i is less than j\n");
    else if (j<=k)
        printf("j is less than or equal to k\n");
    else if (i==k)
        printf("i is equal k\n");
}//main
```

- 위의 예에서는 if나 else if의 어느 조건도 거짓이므로, 어느 문장도 실행되지 않습니다.

• 아래의 예에서는 두 조건이 참이지만, 첫 번째 비교된 문장만이 실행된다는데 주의해야 합니다.

```
#include <stdio.h>

void main() {
    int i=2,j=3,k=2;

    if (i<j)//이 조건은 참이다.
        printf("i is less than j\n");//이 문장이 실행된다.
    else if (j<=k)
        printf("j is less than or equal to k\n");
    else if (i==k)//이 조건도 참이다.
        printf("i is equal k\n");
}//main
```

• 아래의 예에서는 else가 쓰인다면, 최소한 한 문장은 실행된다는 것을 보여줍니다.

```
#include <stdio.h>

void main() {
    int i=5,j=4,k=3;

    if (i<j)
        printf("i is less than j\n");
    else if (j<=k)
        printf("j is less than or equal to k\n");
    else if (i==k)
        printf("i is equal to k\n");
    else
        printf("i is greatest\n");//이 문장이 실행된다.
}//main
```

• 두 문장 이상이 실행되게 하려면, 블록**(block: 복합문(compound statements))**을 만들어야 합니다.

```
#include <stdio.h>

void main() {
    int i=5,j=4;

    if (i<j)
        printf("i is less than j\n");
        printf("this string will be printed always\n");
          //잘못된 들여 쓰기
}//main
```

• 위의 예에서 두 번째 printf()는 항상 실행됩니다.

• 단순한 들여쓰기(indentation)가 문법을 결정짓는 것은 아닙니다.

- 두 문장을 하나의 문장처럼 만들어 주어야 합니다.

```
#include <stdio.h>

void main() {
    int i=5,j=4;

    if (i<j) {
        printf("i is less than j\n");
        printf("this string will be printed always\n");
    }//if
}//main
```

switch

```
switch（표현식）{
    [case 표현식:
        문장;
        [break;]][...]
    [default:
        문장;
        [break;]]
}
```

• 아래의 소스는, 사용자가 입력한 문자(character)가 a, b 혹은 c인지 검사합니다.

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

void main() {
    int i;

    printf("Enter a char:");
    i=getch();
    if (i=='a')
        printf("a pressed\n");
    else if (i=='b')
        printf("b pressed\n");
    else if (i=='c')
        printf("c pressed\n");
    else
        printf("whick key pressed?\n");
}//main
```

• 위 문장은 switch를 사용하여 아래와 같이 대등하게 바꿀 수 있습니다.

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

void main() {
    int i;

    printf("Enter a char:");
    i=getch();
    switch (i) {
        case 'a':
            printf("a pressed\n");
            break;
        case 'b':
            printf("b pressed\n");
            break;
        case 'c':
            printf("c pressed\n");
            break;
```

```
        default:
            printf("whick key pressed?\n");
            break;
    }//switch
}//main
```

- switch 는 괄호 안의 표현식을 평가하여, 평가 값이 case 뒤에 명시된 값과 일치하는지 검사합니다.
- 값이 일치하면, case 이후의 문장을 실행하기 시작합니다.
- break를 만나면 switch문을 탈출(exit)합니다.

- break는 선택 사항이므로 프로그래머의 의도에 따라 생략할 수 있음을 주의하세요.
- 대문자인 경우도 의도대로 동작하기 위해서는 소스를 아래와 같이 수정합니다.

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

void main() {
    int i;

    printf("Enter a char:");
    switch (i=getch()) {
        case 'A':
        case 'a':
            printf("a pressed\n");
            break;
        case 'B':
        case 'b':
            printf("b pressed\n");
            break;
```

```
        case 'C':
        case 'c':
            printf("c pressed\n");
            break;
        default:
            printf("whicK key pressed?\n");
            //break;//마지막의 break는 생략 가능하다.
    }//switch
}//main
```

for

```
for ([표현식1]; [표현식2]; [표현식3])
    문장;
```

- for문은 일반적으로 정해진 반복을 수행하기 위해 사용합니다.

① for문에 들어오기 전 *표현식1*을 실행합니다.
② *표현식2*를 평가합니다. 참이면, 단계 ③으로 갑니다. 아니면, for를 탈출합니다.
③ *문장*을 수행합니다.
④ *표현식3*을 수행합니다.
⑤ 단계 ②로 갑니다.

• 이와 같은 for문의 특징을 적용하여, for의 문법을 새로 쓰면, 아래와 같습니다.

```
for ([초기문]; [비교문]; [증가문])
    문장;
```

• for문의 표현식들은 생략할 수 있다는 것에 주의하세요. 문장은 생략해도 되지만, 세미콜론 ; 은 생략할 수 없습니다.

```
초기문;
for (;[비교문];) {
    문장;
    증가문;
}
```

- 아래의 예제는 1부터 10까지의 정수를 출력합니다.

```
#include <stdio.h>

void main() {
    int i;

    for (i=1;i<=10;++i)
        printf("%d\n",i);
}//main
```

• 비교문이 생략되면, 컴파일러는 참으로 간주합니다(권장하지 않습니다). 그러므로 아래의 코드는 **무한 루프(infinite loop)**를 만듭니다.

```
for (;;)
    문장;
```

• 1부터 100까지의 합을 구하는 프로그램을 for를 사용하여 만들어 봅시다.

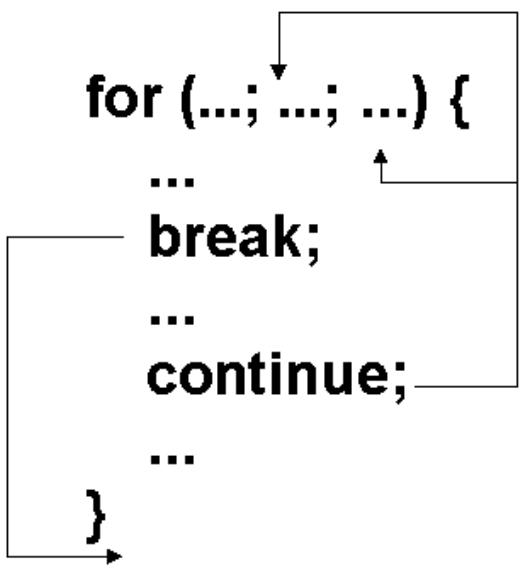
```
#include <stdio.h>

void main() {
    int s=0,i;

    for (i=1;i<=100;++i)
        s=s+i;
    printf("%d\n",s);
    //       5050
}//main
```

• 위의 예에서 for문을 탈출했을 때의 i는 101이라는 사실을 주의하세요. **경계 조건 (boundary condition)**은 모든 제어문에서 대부분 중요합니다.

• 반복문 - for,while과 do문 - 에서 공통적인 사항은 **break**와 **continue**에 관한 것입니다. 각각의 용도는 아래와 같습니다.

① break는 자신을 둘러싼 **가장 가까운 반복문**을 탈출합니다.

② continue는 자신을 둘러싼 가장 가까운 반복문의 처음 - 조건 검사 부분 - 으로 분기합니다.



• for문에서는 continue가 바로 비교문으로 분기하지 않습니다. 마지막 증가문을 실행한 다음, 비교문으로 분기합니다.
• 아래의 소스는 1부터 10까지를 출력하지만, 5는 출력하지 않습니다.

```
#include <stdio.h>

void main() {
    int i;

    for (i=1;i<=10;++i) {
        if (i==5) continue;
        printf("%d\n",i);
    }//for
}//main
```

• 흔히 범하는 오류는 break나 continue가 블록을 탈출한다고 생각하는 것입니다. 아래의 문장에서 break는 if를 탈출할까요?

```
#include <stdio.h>

void main() {
    int i;

    for (i=1;i<=10;++i) {
        if (i==5) {
            i=i+1;
            break;
        }//if
        printf("%d\n",i);
    }//for
}//main
```

• 그렇지 않습니다. break는 자신을 둘러싼 가장 가까운 반복문을 탈출합니다.

• 연속된 두 개의 블록을 탈출하기 위해서는 어떻게 해야 할까요? 한 가지 해결책은 **깃발 변수(flag variable)**를 사용하는 것입니다.

```
#include <stdio.h>
#include <conio.h>
void main() {
    int i,j,flag=0;

    for (i=2;i<=100;++i) {
        for (j=2;j<=9;++j) {
            if (kbhit()) {
                flag=1;
                break;
            }//if
            printf("%d*%d=%d\n",i,j,i*j);
        }//for
        if (flag==1)
            break;
    }//for
}//main
```

while

while (표현식)
    문장;

- while문은 *표현식*이 참(1)인 동안 *문장*을 실행합니다.
- for문과는 다르게 *표현식*을 생략할 수 없습니다.

• 아래의 소스는 1부터 10까지의 정수를 출력합니다.

```
#include <stdio.h>

void main() {
    int i=1;

    while (i<=10) {
        printf("%d\n",i);
        ++i;
    }//while
}//main
```

• while문을 탈출했을 때, i의 값이 11이라는 것에 주의하세요.

• while문 역시 반복문이므로 **break**와 **continue**를 가질 수 있습니다.

• break는 while문을 탈출합니다. continue는 while문의 처음 - 표현식이 있는 조건 검사문 - 으로 분기합니다.

do...while

```
do
        문장;
while (표현식);
```

- while과 do문은 매우 비슷합니다.
- while이 조건을 먼저 검사하고 문장을 실행하는 반면, do문은 *문장*을 먼저 실행하고 *표현식*을 검사합니다.

• 아래의 예는 1부터 10까지의 정수를 출력합니다.

```
#include <stdio.h>

void main() {
    int i=1;

    do {
        printf("%d\n",i);
        ++i;
    } while(i<=10);
}//main
```

• continue가 do쪽으로 분기하는 것이 아니라, while의 조건문 쪽으로 분기한다는데 주의하세요.

실습문제

1. for,while,do...while문에 사용된 break와 switch에 사용된 break의 차이점을 설명하세요. 왜 이것은 다른가요? switch문에는 왜 continue를 사용할 수 없는가요?

2. 무조건 분기 명령/함수 중 return; break; exit()에 대해서 비교 설명하세요. exit()의 파라미터는 무엇을 명시하는가요?

3. 재귀 함수(recursive function)를 비재귀 함수로 바꾸는 알고리즘은 goto를 사용합니다. 이 알고리즘을 기술하고, goto가 꼭 필요한지에 대해서 설명하세요.

4. 다음 for문을 동등한 while, do...while문으로 바꾸세요.

```
for(i=0;i<10;i+=2)
    s=s*i+i;
```

@