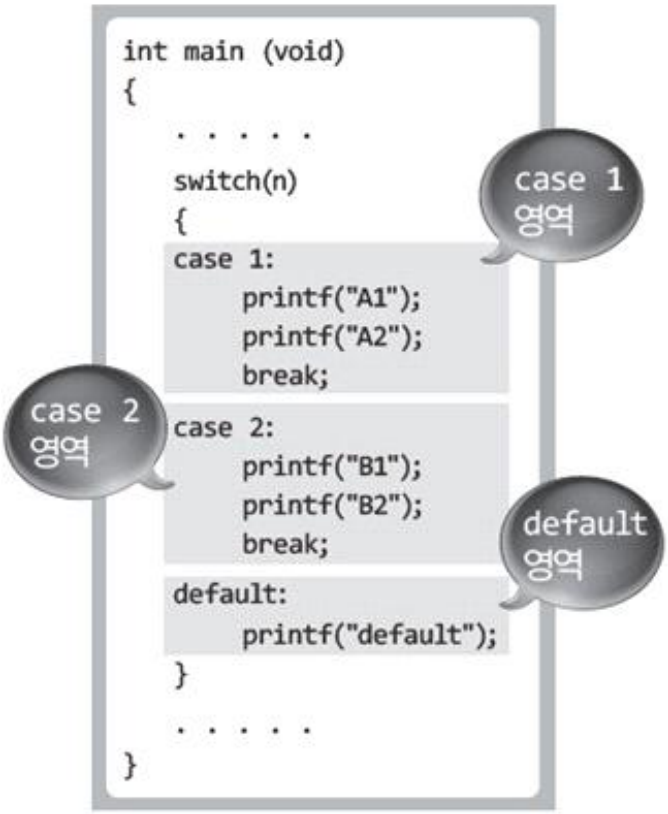
5주차 2차시 switch 문

【학습목표】

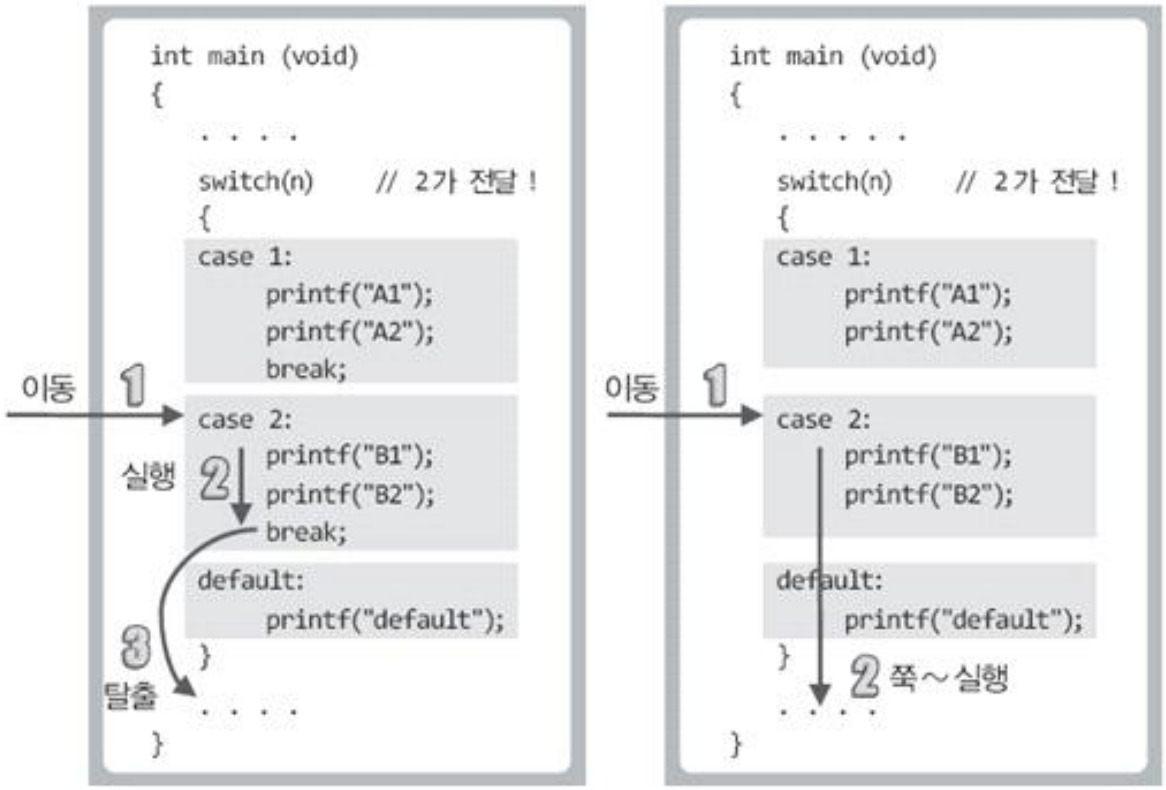
1. switch, doto문을 설명할 수 있다.
2. 함수의 정의에 대해 설명할 수 있다.

학습내용1 : switch문에 의한 선택적 실행과 goto문

1. switch문의 구성과 기본기능



switch문의 기본 구성



삽입되어 있는 break문이 갖는 의미

2. switch문 관련 예제

```
int main(void)
{
    int num;
    printf("1이상 5이하의 정수 입력: ");
    scanf("%d", &num);
    switch(num)
    {
    case 1:
        printf("1은 ONE \n");
        break;
    case 2:
        printf("2는 TWO \n");
        break;
    case 3:
        printf("3은 THREE \n");
        break;
    case 4:
        printf("4는 FOUR \n");
        break;
    case 5:
        printf("5는 FIVE \n");
        break;
    default:
        printf("I don't know! \n");
    }
    return 0;
}
```

```
1이상 5이하의 정수 입력: 3
3은 THREE
```

```
1이상 5이하의 정수 입력: 5
5는 FIVE
```

```
1이상 5이하의 정수 입력: 7
I don't know!
```

3. break문을 생략한 형태의 switch문 구성

```
int main(void)
{
    char sel;
    printf("M 오전, A 오후, E 저녁 \n");
    printf("입력: ");
    scanf("%c", &sel);

    switch(sel)
    {
    case 'M':
    case 'm':
        printf("Morning \n");
        break;
    case 'A':
    case 'a':
        printf("Afternoon \n");
        break;
    case 'E':
    case 'e':
        printf("Evening \n");
        break;  // 사실 불필요한 break문!
    }
    return 0;
}
```

```
M 오전, A 오후, E 저녁
입력: M
Morning
```

위의 예제와 같은 경우 다음과 같이 두 case 레이블을 한 줄에 같이 표시하기도 한다.

case ‘M’: case ‘m’:
. . . . .
case ‘A’: case ‘a’:
. . . . .
case ‘E’: case ‘e’:

4. switch vs. if...else if...else

```
if(n ==1)
   printf("AAA");
else if(n==2)
   printf("BBB");
else if(n==3)
   printf("CCC");
else
   printf("EEE");
```

VS.

```
switch(n)
{
 case1:
    printf("AAA");
    break;
 case2:
    printf("BBB");
    break;
 case3:
    printf("CCC");
    break;
 default:
    printf("EEE");
}
```

if...else if...else보다 switch문을 선호한다. switch문이 더 간결해 보이기 때문이다.

```
if (0<=n && n<10)
   printf("0이상 10미만");
else if(10<=n && n<20)
   printf("10이상 20미만");
else if(20<=n && n<30)
   printf("20이상 30미만");
else
   printf("30이상 ");
```

```
switch(n)
{
 case ??? :
    printf("0이상 10미만");
    break;
 case ??? :
    printf("10이상 20미만");
    break;
 case ??? :
    printf("20이상 30미만");
    break;
 default:
    printf("30이상 ");
}
```

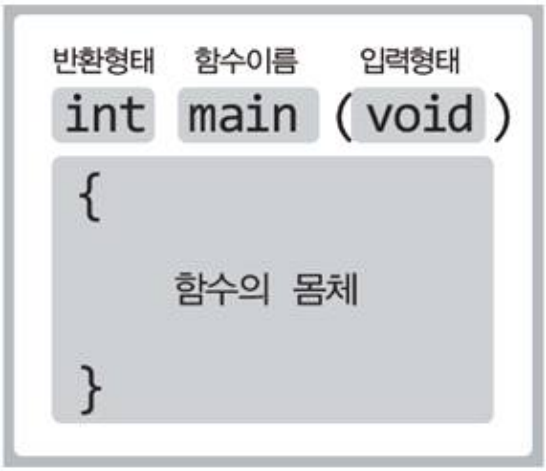
모든 if...else if...else문을 switch문으로 대체할 수 있는 것은 아니다.

학습내용2 : 함수의 정의

1. 함수를 만드는 이유

"Divide and Conquer!"

√다수의 작은 단위 함수를 만들어서 프로그램을 작성하면 큰 문제를 작게 쪼개서 해결하는 효과를 얻을 수 있다. 그러나 함수를 만드는 이유 및 이점은 이보다 훨씬 다양하다.



main 함수를 포함하여 함수의 크기는 작을수록 좋다. 무조건 작다고 좋은 것은 아니지만, 불필요하게 큰 함수가 만들어지지 않도록 주의해야 한다.

하나의 함수는 하나의 일만 담당하도록 디자인 되어야 한다. 물론 하나의 일이라는 것은 매우 주관적인 기준이다. 그러나 이러한 주관적 기준 역시 프로그래밍에 대한 경험이 쌓이면 매우 명확한 기준이 된다.

2. 함수의 입력과 출력: printf 함수도 반환을 합니다.

```
int main(void)
{
    int num1, num2;
    num1=printf("12345\n");
    num2=printf("I love my home\n");
    printf("%d %d \n", num1, num2);
    return 0;
}
```

```
12345
I love my home
6 15
```

printf 함수도 사실상 값을 반환한다. 다만 반환값이 필요 없어서 반환되는 값을 저장하지 않았을 뿐이다.
printf 함수는 출력된 문자열의 길이를 반환한다.

함수가 값을 반환하면 반환된 값이 함수의 호출문을 대체한다고 생각하면 된다.
예를 들어서 아래의 printf 함수 호출문이 6을 반환한다면,

num1=printf("12345 ₩n");

함수의 호출결과는 다음과 같이 되어 대입연산이 진행된다.
num1=6;

3. 함수의 구분

√전달인자와 반환 값의 유무에 따른 함수의 구분!
- •유형 1: 전달인자 있고, 반환 값 있다! 전달인자(○), 반환 값(○)
- •유형 2: 전달인자 있고, 반환 값 없다! 전달인자(○), 반환 값(X)
- •유형 3: 전달인자 없고, 반환 값 있다! 전달인자(X), 반환 값(○)
- •유형 4: 전달인자 없고, 반환 값 없다! 전달인자(X), 반환 값(X)

4. 전달인자 반환 값 모두 있는 경우

전달인자는 int형 정수 둘이며, 이 둘을 이용한 덧셈을 진행한다.
덧셈결과는 반환이 되며, 따라서 반환형도 int형으로 선언한다.
마지막으로 함수의 이름은 Add라 하자!

```
A.  B.  C.
int Add (int num1, int num2)
{
    int result = num1 + num2;
D.  return result;
}
```

A. 반환형
B. 함수의 이름
C. 매개변수
D. 값의 반환

함수호출이 완료되면 호출한 위치로 이동해서 실행을 이어간다.

```
int Add(int num1, int num2)
{
    return num1+num2;
}

int main(void)
{
    int result;
    result = Add(3, 4);
    printf("덧셈결과1: %d \n", result);
    result = Add(5, 8);
    printf("덧셈결과2: %d \n", result);
    return 0;
}
```

덧셈이 선 진행되고
그 결과가 반환됨

```
덧셈결과1: 7
덧셈결과2: 13
```

5. 전달인자나 반환 값이 존재하지 않는 경우

```
void ShowAddResult(int num)   // 인자전달 (O), 반환 값 (X)
{
    printf("덧셈결과 출력: %d \n", num);
}
```

```
int ReadNum(void)   // 인자전달 (X), 반환 값 (O)
{
    int num;
    scanf("%d", &num);
    return num;
}
```

```
void HowToUseThisProg(void)   // 인자전달 (X), 반환 값 (X)
{
    printf("두 개의 정수를 입력하시면 덧셈결과가 출력됩니다. \n");
    printf("자! 그럼 두 개의 정수를 입력하세요. \n");
}
```

6. 4가지 함수유형을 조합한 예

```
int Add(int num1, int num2)    // 인자전달 (O), 반환 값 (O)
{
    return num1+num2;
}

void ShowAddResult(int num)    // 인자전달 (O), 반환 값 (X)
{
    printf("덧셈결과 출력: %d \n", num);
}

int ReadNum(void)    // 인자전달 (X), 반환 값 (O)
{
    int num;
    scanf("%d", &num);
    return num;
}

void HowToUseThisProg(void)    // 인자전달 (X), 반환 값 (X)
{
    printf("두 개의 정수를 입력하시면 덧셈결과가 출력됩니다. \n");
    printf("자! 그럼 두 개의 정수를 입력하세요. \n");
}
```

```
두 개의 정수를 입력하시면 덧셈결과가 출력됩니다.
자! 그럼 두 개의 정수를 입력하세요.
12 24
덧셈결과 출력: 36
```

【학습정리】

1. 함수의 이름을 정하고, 기능을 구현하고, 전달인자와 반환 값을 결정짓는 것이 함수를 정의하는 것이다.

2. 함수의 구성은 반환형, 함수의 이름, 매개변수, 값의 반환으로 구성된다.

3. switch문으로 구현된 모든 예제는 if...else if...else를 통해서도 구현할 수 있다.

4. switch문의 장점은 if...else if...else에 비해서 간결해 보인다는 것이다. 따라서 분기의 수가 많아지면 switch문을 사용하는 것이 좋다.