jinyPHP 1권 - 8. 조건문

**8 조건 제어문**

우리 일상 생활은 알게 모르게 수많은 선택과 결정을 합니다. 컴퓨터 프로그램 또한 하나의 일을 처리할 때 사람처럼 수많은 선택의 조건을 처리하며 수행합니다. 하지만 컴퓨터 프로그램은 이러한 모든 선택 여부를 사용자에게 물어보지 않습니다.

프로그램을 개발하고 코드를 작성하는 것은 이러한 선택과 결정을 최소화하고 자동화함으로써 의미를 부여합니다. 따라서 프로그램을 작성하면서 가장 많이 사용되는 기본 문법은 바로 조건 처리를 하는 제어문이라고 할 수 있습니다. 제어문은 프로그램이 순차적으로 해석 처리하는 과정에 어떠한 결정이 필요한 경우, 조건에 따라서 서로 다른 흐름을 변경하는 역할을 합니다.

선택과 결정을 한다는 것은 어떤 일은 할 것이며 어떤 일은 하지 않겠다는 것입니다. 프로그램 또한 개발자가 작성한 코드를 모두 실행하지는 않습니다. 어떤 코드는 동작을 하고 어떤 코드는 동작을 하지 않습니다. 이것은 수많은 코드를 작성하더라도 동작 조건에 따라서 실행을 해야 하는 경우도 있고 건너뛰어야 하는 경우도 필요하기 때문입니다.

컴퓨터 프로그램 언어에서 이러한 다채로운 동작을 구별하고 코드를 분개하는 기능을 처리하는 것이 제어문법입니다. 제어문을 또 다른 말로 조건문이라고도 많이 부릅니다.

PHP 언어는 C 언어스타일의 조건문 작성 규칙과 유사하게 작성할 수 있습니다. 이렇게 유사한 코드 작성 방법은 새로운 언어를 매우 쉽게 배울 수 있습니다.

**조건 제어문 순서도**

프로그램을 코드 이외의 다이어그램, 즉 순서도로 표현을 하기도 하는데 이때 조건 제어문은 순서도에서 다이아몬드 그림으로 표기합니다.



프로그램 순서도에서 다이아몬드 기호는 조건 선택을 하는 것이라고 이해하면 됩니다.

**8.1 참과 거짓**

우리가 선택과 결정을 할 때 많이 사용하는 기호가 yes와 no일 것입니다. 즉, ‘하다’와 ‘안 하다’입니다. 그 외 애매한 표현으로 선택을 결정하지 않습니다. 컴퓨터는 yes와 no 기호 대신에 논리기호 true와 false를 사용합니다. 이는 컴퓨터가 0과 1 값을 가지는 이진법 연산을 가진 기계이기 때문입니다.

컴퓨터에서 제어문은 프로그램의 코드를 조건에 따라서 분기를 합니다.

앞에서 변수 타입 중 논리(bool) 변수와 논리 연산에 대해서 살펴보았습니다. 논리 값과 논리 연산은 그 결과 값으로 논리 참(true)과 거짓(false)을 반환합니다. 여기서 조건이란 참(yes)과 거짓(no)이라는 이분법적인 두 가지 조건을 구분합니다.

**8.1.1 참이란?**

선택과 결정에서 yes의 의미와 비슷합니다. 어떤 결과 값이 존재하거나 유효한 경우를 **참**이라고 말합니다. 프로그램에서는 참이라는 상수명으로 true라고 표현합니다. 값으로 표현한다고 하면 1의 값을 가집니다.

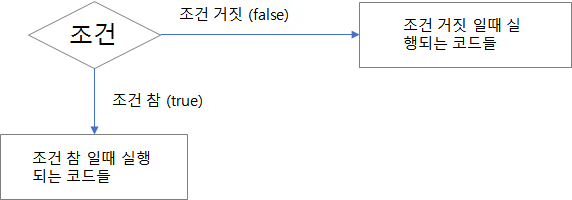
**8.1.2 거짓이란?**

선택과 결정에서 no의 의미와 비슷합니다. 어떤 결과 값이 없거나 유효하지 않은 경우를 **거짓**이라 말합니다. 프로그램에서는 거짓이라는 상수명으로 false라고 표현합니다. 값으로 표현한다고 하면 0의 값을 가집니다.

컴퓨터 언어 제어문에서는 조건을 참과 거짓으로 판별합니다. 하지만 입력되는 조건은 또 다른 논리 기호를 섞어서 입력 값으로 사용할 수 있습니다.

**조건 제어문 참과 거짓 동작에 대한 순서도 개념**

조건 제어문을 순서도로 다시 한번 표현하면 아래와 같습니다.



좀 복잡한 프로그램을 제작하거나 코딩 작성 전 논리적인 구상을 할 때 위와 같이 순서도를 그려가면서 코드를 생각해보면 더 쉽게 코드를 이해하고 작성할 수 있을 것입니다.

**8.2 if문**

if 키워드는 대부분 프로그램 언어에서 조건문으로 사용하고 있는 키워드입니다. 또한 if는 사전적 의미로 “(만약) .. 면”이라는 의미를 가지고 있습니다. PHP언어에서는 조건 명령문 키워드로 if 를 사용합니다.

**8.2.1 조건 단일 실행**

다음 문법은 가장 기본적인 조건 제어문 표현 문법 예입니다. 문법을 살펴보면 “(만약) .. 면”이라는 조건을 if 키워드 다음에 적습니다. 즉, 조건이 참(true)일 경우에 뒤에 있는 실행 코드를 실행합니다.

|문법|

|  |
| --- |
| **작성 문법** if ( 조건 ) 실행 코드; |

if라는 키워드는 C 언어 스타일의 조건 제어문으로 이미 많은 프로그램 언어에서 유사하게 사용하고 있습니다. PHP 언어에서도 쉬운 언어 학습을 위해 동일한 형태의 키워드와 문법으로 제공하고 있습니다.

PSR 표준 스타일 코딩 권장에 따르면 if 키워드와 조건문 소괄호() 블록 사이에 한 칸의 공백을 추가하여 일관적으로 작성하는 것을 추천합니다.

단일 조건 실행이란 조건이 참일 경우 실행하는 문장이 한 개뿐이라는 것입니다.

**예제파일** if-01.php

|  |
| --- |
| <?php  $country = "Korea";  // 문자열을 비교합니다.  **if ($country == "Korea") echo $country . "에 오신 것을";**    echo "환영합니다.";  ?> |

위의 예는 조건에 대한 단일 실행 코드입니다. if 조건문과 같은 줄에 한 줄의 명령문만 위치하고 있습니다. $country 변수에는 국가 이름이 저장되어 있습니다. 만일 $country 값이 “Korea”와 같다면 실행 결과는 다음과 같습니다.

**결과**

Korea에 오신 것을 환영합니다.

만약 값이 틀리면 다음과 같이 출력됩니다.

**결과**

환영합니다.

**예제 파일** if-01-1.php

|  |
| --- |
| <?php  $age = 18;  // 숫자를 비교합니다.  if ($age >= 18) echo $age . "는 성인입니다.";    // 숫자를 비교합니다.  if ($age < 18) echo $age . "는 미성년입니다.";  ?> |

**결과)**

18는 성인입니다.

단일 실행은 조건이 간단한 작업을 처리할 때 매우 유용합니다. 단일 실행은 코드를 간결하게 하고 줄 단위로 가독성이 쉬운 코드로 작성할 수 있습니다.

**8.2.2 조건 다중 실행**

단일 실행 조건은 조건 만족 시 하나의 실행 코드만 처리를 합니다. 코드를 작성하다 보면 하나의 실행 코드만 실행하는 경우는 그렇게 많지 않습니다. 소스 코드의 여러 명령 블록을 묶어서 조건을 처리하게 됩니다.

조건 다중 실행이란 여려 명령들을 묶어서 하나의 조건으로 처리한다는 의미입니다. 한 개 이상의 다수의 실행 코드를 처리하기 위해서는 중괄호 { } 기호로 실행할 코드를 감싸주면 됩니다.

**|문법|**

|  |
| --- |
| if (조건 ) **{**  조건이 만족을 하면 실행됩니다…  실행 코드;  실행 코드;  실행 코드;  **}** |

다중 실행 조건을 작성할 때 주의할 점은 중괄호 시작과 끝을 짝을 맞추어 작성을 해야 합니다. 만일 중괄호의 짝을 한쌍으로 일치하지 않으면 PHP는 문법적 오류로 출력과 동작을 하지 않습니다.

**조건 다중 실행 예제**

**예제 파일)** if-02.php

|  |
| --- |
| <?php  $num = "10";  echo "num 값은 " . $num . "입니다.";  if ($num >= 0 ) {  echo " (양수입니다.) <br>";  $num --;  echo " 값을 하나 감소합니다 ($num)";  }  ?> |

**결과**

num 값은 10입니다. (양수입니다.)

값을 하나 감소합니다 (9)

위의 예는 조건에 대한 다중 실행 코드입니다. 위의 예는 $num 값이 양수인지를 판별하고, 양수면 메시지를 추가로 출력하고 값을 1만큼 감소합니다.

즉, 조건에 따라서 동작하는 명령이 세 개인 것입니다. 이런 조건 다중 처리일 경우 조건 만족 시 여러 줄을 실행을 하기 위해서 실행할 코드 부분을 중괄호 { }로 묶어주면 됩니다.

**8.3 else 문**

지금까지 조건 설명의 조건은 참(true)인 경우의 처리를 설명을 했습니다.

하지만 조건의 만족이 거짓일 때 처리해야 하는 일들도 필요할 것입니다. if 문법에서는 거짓에 대한 로직 부분을 else라는 명령으로 따로 선언할 수 있습니다. 즉 if 다음에 오는 **else 이후 문장은 거짓**으로 처리되는 코드입니다.

**|문법|**

|  |
| --- |
| if (조건) 참 동작 실행 코드; else 거짓 동작 처리; |

위의 문법을 보면 조건에 대해서 참 동작 코드와 거짓 동작 처리가 **else 명령어로 구분**이 되어 있습니다.

거짓 조건 또한 단일 실행 코드로 작성할 수 있습니다. 만일 거짓 조건의 명령이 하나인 경우 한 줄에 모든 내용을 같이 작성을 할 수 있습니다.

**예제 파일)** if-03.php

|  |
| --- |
| <?php  if ($sex == "man") $sexKorean = "남자"; **else** $sexKorean = "여자";  echo $sexKorean . "입니다. <br>";  ?> |

**결과**

여자입니다.

위의 예는 간단한 거짓 조건을 포함한 조건 처리입니다. $sex 변수에는 영문 문자열 성별을 판별해서 $sexKorean변수에 한글로 성별을 입력 처리합니다. 만일 $sex 값이 “man”이면(참) 조건이 되어 $sexKorean에 “남자” 문자열을 대입하고, 아닌 거짓경우에는 $sexKorean 변수에 “여자” 문자열을 대입합니다.

조건 다중 실행문에서 여려 명령문을 묶어서 조건을 처리하는 것을 보았을 것입니다. else 명령문 또한 필요 시 조건 다수 명령 형태로 거짓 조건에 대한 실행 코드를 중괄호 { }를 묶어서 사용할 수도 있습니다.

**|문법|**

|  |
| --- |
| if (조건) 참 동작 실행 코드;  else **{**  거짓 동작 처리;  거짓 동작 처리;  거짓 동작 처리;  **}** |

다음은 성년 여부를 체크하는 간단한 조건 예제를 실행해봅니다.

**예제 파일** if-04.php

|  |
| --- |
| <?php  $age = 20;  if ($age >= 20) {  echo $age. "은";  echo "성인입니다.";  } else {  echo $age. "은";  echo "미성년자입니다.";  }  ?> |

**결과**

20은 성인입니다.

위의 예는 $age에 나이를 입력한 후에 변수 값에 따라서 성인 여부를 판별하는 조건문입니다. $age의 값이 20을 포함한 이상의 값이 만족할 때 "성인입니다." 문자열을 표기하고, 조건이 만족하지 않을 때는 "미성년자 입니다."라고 문자열을 출력합니다.

조건 제어문을 작성할 때 항상 참과 거짓에 대한 양쪽 사항에 대한 처리 로직을 같이 세트로 만드는 것이 좋습니다.

**8.4 다수의 분기 else if**

선택과 결정을 하는 조건 제어문은 참과 거짓 이분법적인 논리를 판단(결정)하여 동작합니다. 하지만 이분법적인 논리 판단은 다양한 조건을 처리하는 데는 제약이 있습니다.

만일 우리의 선택 결정할 수 있는 조건이 세 가지, 네 가지일 경우도 있습니다. 즉, 시험 문제를 풀거나 여러 가지 경우 중 하나를 선택하는 것입니다.

다양한 조건으로 판별해야 할 때는 어떻게 해야 할까요? 이런 경우를 대비하여 조건 제어문은 **else if 명령을 통해 새로운 조건을 추가**로 연결할 수 있습니다.

else if는 else 거짓 조건에 또 다른 조건문을 연장하는 것입니다. 즉, 거짓일 때의 코드 블록이 또 하나의 조건문인 것입니다.

**|문법|**

|  |
| --- |
| if (조건1) {  조건1 참 동작 실행 코드;  } **else if (조건2)** {  조건1이 거짓이고, 조건2가 참일 때 동작 처리;  } else {  조건1과 조건2가 모두 거짓일 때 동작 처리;  } |

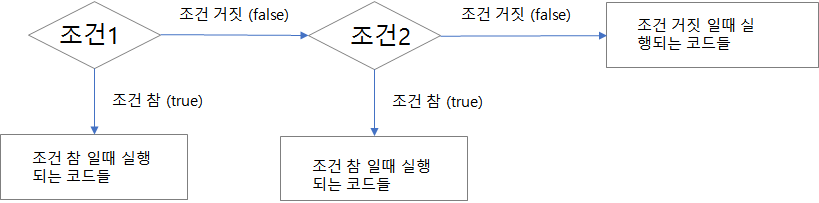
else if를 기존 방법으로 다시 풀어서 설명하면

|  |
| --- |
| if ( 조건1 ) {  조건1 참 동작 실행코드실행 코드;  } else {  if ( 조건2 ){  조건1이 거짓이고, 조건2가 참일 때 동작 처리;  } else {  조건1과 조건2가 모두 거짓일 때 동작 처리 ;  }  } |

와 같다고 표현을 재구성 해볼 수 있습니다. 하지만 이러한 표현은 다수의 중괄호{} 블록을 만들어 내고 쉽게 이해 및 이해하기 어려운 부분이 있습니다.

else if를 이용하여 조건을 연결할 때는 연결 갯수 제한이 없습니다. 필요한 만큼 연결을 추가하여 복잡한 다수의 조건들을 처리할 수 있습니다.

이러한 조건 체인을 순서도로 그려보면 다음과 같을 수 있습니다.



숫자를 구별하는 예제를 조건 체인을 통해 예제를 작성해보겠습니다. 숫자를 구분해보면 양수, 음수와 0으로 세 가지 상태로 구별해 볼 수 있습니다.

**예제 파일)** if-05.php

|  |
| --- |
| <?php  $num = 5;  if ($num > 0) {  echo "양수입니다.";  } else if ($num < 0) {  echo "음수입니다.";  } else if ($num == 0) {  echo "0 의 값입니다.";  } else {  echo "알 수 없습니다..";  }  ?> |

**결과**

양수입니다.

위의 예는 간단하게 두 개의 조건 체인을 통해 값을 판별하는 예제입니다. 먼저, $num 값이 크기 비교연산자를 통하여 0보다 큰 값인지 확인합니다. 큰 값이면 “양수입니다.” 문자열을 출력합니다. 만일 값이 양수가 아닌 거짓(false)으로 판별되면 다음 설정된 2차 조건문을 실행합니다.

2차 조건에서도 크기 비교연산자를 통하여 0보다 작은 값인지 판별합니다. 0보다 작은 값이면 “음수입니다.” 문자열을 출력합니다.

하지만 수의 표현에는 양수와 음수가 아닌 수 0이 있습니다. 만일 값이 앞의 양수도 아니고, 현재 조건인 음수도 아니면(거짓), 다음 3차 조건으로 또 다시 조건을 판단을 합니다.

비교연산자를 통하여 값이 0이 같은지를 판단합니다. 값이 0이면 “0의 값입니다.” 문자열을 출력합니다. 일반적으로 숫자가 양수나 음수가 아니면 0으로 판단할 수 있겠지만 혹시 입력되는 $num이 숫자가 아닐 수도 있습니다. 따라서 이러한 예외도 생각해서 $num의 값이 0인지 판별합니다.

마지막 else는 $num이 숫자가 아닌 양수, 음수, 0을 판별할 수 없을 때 “알 수 없습니다..” 문자열을 출력합니다.

하지만 조건문을 많이 사용할수록 프로그램의 실행 성능은 차이가 있을 수 있습니다. Low-level 언어인 어셈블리로 보면 조건문은 cmp 기호로 CPU의 많은 리소스를 차지하게 됩니다. 그렇다고 해서 조건문을 줄여서 최적화하여 개발하라고 애기하는 것은 아닙니다. 요즘은 컴퓨터의 성능이 많이 좋아졌기 때문에 수많은 조건을 처리하기에 부족함이 없습니다.

선택 조건 제어문은 프로그램 개발 경험이 많을수록 다양한 예외에 대한 처리를 준비하고 코드로 작성하는 것이 중요합니다. 컴퓨터는 사람이 일반적으로 생각하는 것과 달리 모든 상황별 예외를 정의해야 합니다.

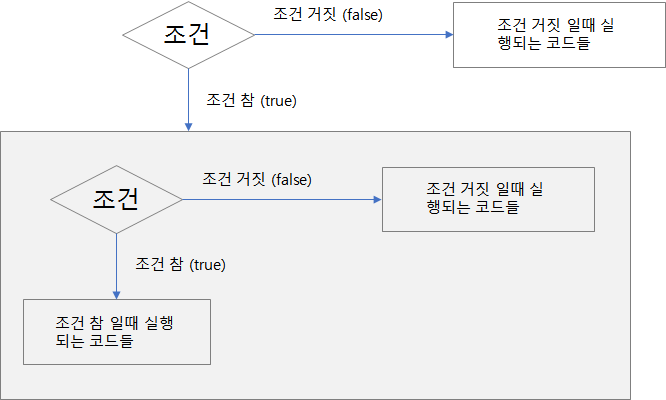
**8.5 조건문의 중복**

선택과 결정if 조건 제어문은 참(true)과 거짓(false), 또 다른 조건(else if)을 통해 계속 참과 거짓 조건을 연결하여 처리할 수 있습니다. 하지만 이렇게 1차원적인 조건이 아니라 어떤 조건을 만족하면서 또 다른 조건을 판별할 수 있는 2차원적인 조건은 어떻게 작성을 할까요?

else if조건문을 설명할 때 또 다른 표현법으로 if 조건 안에 또 다른 if 조건문이 들어간 문법적 표현을 보았을 것입니다. if 조건문은 또 다른 if 조건문을 포함할 수 있습니다. 즉, 하나의 if 조건문 상태 안에 또 다른 조건을 추가할 수 있습니다.

다수의 선택에 있어서 참의 조건을 계속 판별하거나 거짓의 조건을 계속 판별해가는 다중 조건을 처리하는 데 이중 조건 중복은 매우 유용합니다.

다음 그림은 이중 중복 조건을 순서도로 그려서 표현하면 다음과 같습니다.



이중 조건문은 참 조건 블록 안에 넣을 수도 있고, 거짓 조건 블록 안에 넣을 수도 있습니다. 즉, 중괄호 블록으로 둘러싸인 코드 안에는 중복으로 코드를 이중화할 수 있습니다.

**프로그램 계층화**

우리의 프로그램은 하나의 명령을 순차적으로 실행합니다. 즉 한 줄 한 줄 실행합니다. 하지만 조건 제어문을 만나게 되면 코딩 로직이 조건에 따라 분기됩니다.

중복 조건의 처리는 프로그램이 작성이 1단계로 순차적으로 작성되는 것과 달리 계층적으로 단계(계단)화되는 것과 유사합니다. 위의 이중 조건을 문법적으로 다음과 같이 작성을 해볼 수 있습니다.

**|문법|**

|  |
| --- |
| if (조건1) **{**  조건1 만족할 때 실행 코드;  if (조건2) {  조건2 만족할 때 실행 코드;  }  **}** |

조건의 중복 제한의 수는 없습니다. 필요한 만큼 2차, 3차로 세분화하여 작성할 수 있습니다. 조건이 많을수록 프로그램은 복잡해지고 고도화된 작업들을 처리합니다.

**|문법|**

|  |
| --- |
| if (조건1) {  조건1 만족할 때 실행 코드;  if (조건2) {  조건2 만족할 때 실행 코드;  }  } else {  조건1 만족하자 않을 때 실행 코드;  if (조건3) {  조건3 만족할 때 실행 코드;  }  } |

다음은 이중 조건문을 통해 성별과 성년 나이를 확인하는 간단한 예제입니다.

**예제 파일** if-06.php

|  |
| --- |
| <?php  $age = 20;  $sex = "woman";  $type = "coperation";  if ($sex == "man") {    if ($age >= 20) {  echo "성인 남자입니다.";  } else {  echo "미성년자 남자입니다.";  }  } else if ($sex == "woman") {    if($age >= 20){  echo "성인 여자입니다.";  } else {  echo "미성년자 여자입니다.";  }  } else if ($sex == "company") {    if ($type == "personal") {  echo "개인 기업입니다.";  } else {  echo "법인 기업입니다.";  }  } else {  echo "성별을 구분할 수 없습니다.";  }  ?> |

**결과**

성인 여자입니다.

위의 예제는 성별 변수 $sex를 세 가지 상태로 구분합니다. 또한 각각의 나이 $age 변수에 따라서 “성년”과 “미성년자”를 구분합니다. 기업회원인 경우에는 “법인”과 “개인” 기업으로 구분합니다.

중복 제어문은 이처럼 다양한 조건과 상태에 따라서 복합적으로 선택과 결정을 할 수 있습니다.

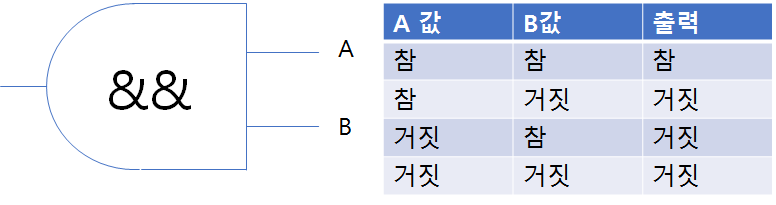
**8.6 논리 조건**

선택 결정 조건 제어문의 if 조건 블록 소괄호 ( ) 하나로 조건 값에 따라 참(true)과 거짓(false)으로 구분하여 동작합니다. 만일 조건이 한 가지 조건이 아닌 두 가지 조건을 모두 처리해야 한다면 앞에서 배운 조건의 중첩 또는 else if문을 통해 코드가가 분기되어 복잡할 수 있습니다.

if문의 조건은 연산 기호 &&, || 와 같은 논리 기호를 통해 여러 조건들을 연결하여 하나의 조건으로 묶어서 작성이 가능합니다. 이렇게 연결된 조건은 하나의 결과 값으로 처리되어 전체 연산 값에 대해서 참/거짓으로 판단하여 논리 값을 판별할 수 있습니다.

**8.6.1 조건 결합 and : &&**

논리 연산 기호 and 기호는 &&를 통해 두 가지 참 조건을 연결할 수 있습니다. AND는 디지털 논리 조건으로 입력되는 두 개의 입력 조건 모두가 참(true)인 경우에 결과도 참(true)을 출력합니다.



즉, 두 개의 값을 곱셈 형태로 연결하는 것과 같습니다. 두 개 조건 모두 참(true)을 확인할 때 사용할 수 있습니다.

&& 조건 연산 결합은 다음과 같은 문법으로 작성할 수 있습니다.

**|문법|**

|  |
| --- |
| if (조건A **&&** 조건 B) {  참 동작 실행 코드;  } else {  거짓일 때 동작 처리 ;  } |

**예제 파일** if-07.php

|  |
| --- |
| <?php  $sex = "man";  $age = 21;    if ($sex == "man" && $age >= 20) {  echo "20세 이상의 남성입니다.";  }  ?> |

**결과**

20세 이상의 남성입니다.

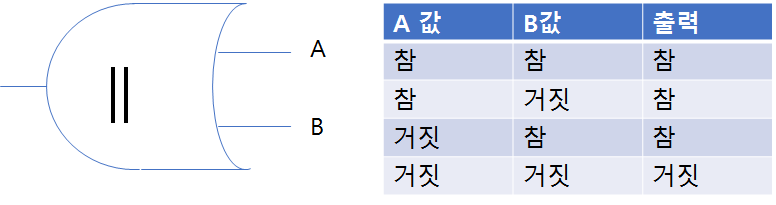
위 예제는 이중 조건처리를 하지 않고 두 가지 조건을 동시에 검사하여 성별과 성년을 모두 한 번에 조건을 검사할 수 있습니다. && 조건의 입력은 A, B 두 개 값보다 더 많이 연결하여 아래와 같이 작성할 수 도 있습니다.

if(조건A **&&** 조건B **&&** 조건C **&&** 조건D …)

필요한 만큼 연결하여 사용하면 됩니다.

**8.6.2 조건 결합 or : ||**

논리 연산 기호 or 기호는 ||를 통해 두 가지 조건을 연결하여 확인할 수 있습니다. 디지털 논리 조건 OR는 입력되는 두 개의 조건 중 하나만 참(true)인 경우 결과도 참(true)을 출력합니다.



즉, 두 개의 값을 덧셈 형태로 연결하는 것과 같습니다. 두 개 조건 중 하나만 참(true)을 확인할 때 사용할 수 있습니다.

|| 조건 연산 결합은 다음과 같은 문법으로 작성할 수 있습니다.

**|문법|**

|  |
| --- |
| if (조건A **||** 조건 B) {  참 동작 실행 코드;  } else {  거짓일 때 동작 처리 ;  } |

**예제 파일** if-08.php

|  |
| --- |
| <?php  $age = 17;    if ($age >= 18 || $age <= 65) {  echo "18세 이상, 65세 이하의 경제 가능 나이 층입니다.";  }  ?> |

**결과**

18세 이상, 65세 이하의 경제 가능 나이 층입니다.

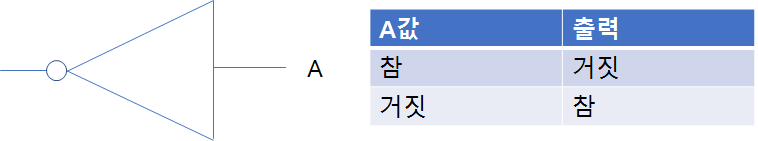
위 예제는 이중 조건 처리를 하지 않고 두 가지 조건을 동시에 검사하여 나이의 범위를 정해 참과 거짓을 한 번에 구별할 수 있습니다. || 조건의 입력은 A, B 두 개 값보다 더 많이 연결하여 다음과 같이 작성할 수 도 있습니다.

if(조건A **||** 조건B **||** 조건C **||** 조건D …)

필요한 만큼 연결하여 사용을 하면 됩니다.

**8.6.3 조건 부정 not : !**

조건 제어문의 문법 if는 조건이 참(true)이면 다음 실행문을 동작하고, 거짓이면 else 이후의 실행문이 동작합니다. 하지만, 논리표현 NOT의 ! 기호를 사용하여 조건 부정문을 통해 조건 결과의 참과 거짓을 뒤집어서 반대로 동작을 하게 만들 수 있습니다.



**작성 문법**

|  |
| --- |
| if ( !조건) {  거짓일 때 동작 처리;  } else {  참 동작 실행 코드;  } |

**예제 파일** if-09.php

|  |
| --- |
| <?php  // $adult 미선언되어 있습니다.  // NELL => false 값  if (!$adult) {  // true 조건  // false 조건이 부정으로 true로 변경됨  echo "성년 판별 논리 값이 비어 있습니다.<br>";  $age = 18;  if($age >= 18){  echo "성인입니다. <br>";  } else {  echo "미성년입니다. <br>";  }  } else {  // false 조건  echo "성년 조건을 판별합니다. <br>";  $adult = false;  if($adult){  echo "성인입니다. <br>";  } else {  echo "미성년입니다. <br>";  }  }  ?> |

**결과**

성년 판별 논리 값이 비어 있습니다.

성인입니다.

즉, 논리표현 NOT은 참과 거짓을 바꾸는 의미를 가지고 있습니다.

**8.6.4 조건 혼용**

앞의 세 개의 논리표현 결합은 하나의 기호만 가지고 조건을 결합했습니다. 하지만 조건 결합은 서로 다른 논리 조건 &&와 ||, !를 소괄호( )를 통해 각각의 다른 논리 값을 결합할 수 있습니다.

(조건 A **&&** 조건B) **||** 조건 C

위의 예는 조건 C가 참(true)이거나, 조건 A나 조건 B 모두 참(true)이어야 결과를 참(true)으로 출력합니다. 만일 서로 다른 논리표현으로 다수의 조건을 결합할 때는 이렇게 산술 연산 우선순위 기호로 사용하는 소괄호를 통해 ( )를 결합이나 우선순위를 정할 수 있습니다.

**8.7 조건문 응용 처리**

이처럼 조건 제어문은 프로그램의 논리적인 처리와 연산을 위하여 작성을 하는데 매우 유용합니다. 프로그램에서 조건문을 이해하면 이미 프로그램의 개발 방법을 ⅓ 정도는 배운 것이라 볼 수 있습니다.

이처럼 조건 제어문은 프로그램의 상당한 양을 차지하는 중요한 문법입니다. 조건 제어문은 프로그램이 고도화하면서 다양한 환경을 제어하는 환경 설정 처리에도 많이 응용됩니다.

흔히 우리가 프로그램을 설치 후 환경 설정 등을 하게 되는데, 이런 경우 이러한 환경을 외부에 저장 후 상태에 따라 조건을 분기하며 프로그램이 동작하게 됩니다.

조건 제어문을 잘 사용하면 고급화된 프로그램을 작성을 할 수 있습니다.

**예제 파일)** if-10.php

|  |
| --- |
| <?php  $debug = true;    if($debug){  echo "개발자 모드입니다.";  } else {  echo "서비스 모드입니다.";  }  ?> |

**결과**

개발자 모드입니다.

조건 제어문은 데이터 처리 이외에도 다양한 환경을 변수를 통해 전체 동작을 제어하는 용도로도 많이 사용합니다. 개발하다 보면 대부분의 소스 코드들은 조건 제어문으로 처리 되는 것을 볼 수 있습니다.

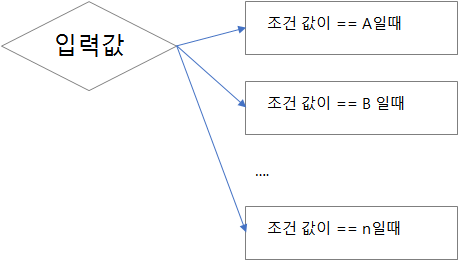
**8.8 switch**

조건 제어문 if의 변형으로 많이 사용하는 제어 문법이 하나 더 있습니다. 바로 switch라는 키워드입니다. if 문법은 다양한 비교 연산자를 통해 조건을 처리할 수 있습니다.

하지만 실제적으로 가장 많이 사용하는 비교 연산자로는 값이 같은지를 판별하는 == 기호 연산자입니다. switch 문법은 다수의 조건을 ==로 비교가 많을 때 이를 보다 쉽게 처리할 수 있는 멀티 분기 조건문 입니다.

PHP 또한 C 언어 스타일의 프로그램 언어처럼 조건 제어문 if문 이외에 멀티 분기 조건 제어문 switch라는 문법을 추가로 제공합니다.

switch 문법을 그림으로 표현하면 다음과 같습니다. 이전의 if문은 yes 또는 no 형태로 두 가지 상태의 값으로 분기되는 것과 달리, 조건값이 같은 다수의 값으로 분기되는 것을 확인할 수 있습니다.



만일 여러 조건을 분개하는 구조를 만들어야 할 때 if elseif를 많이 쓰면 코드상에서 보기 힘든 부분이 있습니다. 여러 개의 if와 else로 구성되는 복잡한 구성을 보다 이해하기 쉬운 코드 형태인 switch 문법으로 작성할 수 있습니다.

즉 switch문은 하나의 입력 값과 동일한 값으로 적용되는 다수의 블록 소스로 구분하고 또한 실행합니다.

switch 문법은 다음과 같습니다. switch 키워드 다음에 조건의 값을 가지는 소괄호 ()와 전체 소스의 조건 분기 블록 중괄호 { }로 구성됩니다.

**|문법|**

|  |
| --- |
| **문법)**  switch ($n) {  case label1:  echo "n 값이 label1일 경우 실행됩니다.";  break;  case label2:  echo "n 값이 label2일 경우 실행됩니다.";  break;  case label3:  echo "n 값이 label3일 경우 실행됩니다.";  break;  default:  echo "n 값이 일치하는 게 없는 경우 실행됩니다.";  } |

switch에서 분기되어 처리되는 소스들은 중괄호 { 로 시작해서 중괄호 }로 끝납니다. 조건의 처리 코드를 삽입하는 { } 안에 두 가지의 서브 명령어 case와 default를 이용하여 여러 일치되는 조건을 작성할 수 있습니다.

**case 조건**

case 서브 명령 키워드는 switch에 입력된 값이 일치되었을 때 동작하는 코드의 시작 부분을 정의합니다. case 키워드 다음에 일치하는 값을 지정하고 뒤에 콜론(:)을 정의한 다음 작성할 코드를 넣어줍니다. 만일 switch 입력 값이 일치되면 프로그램 로직은 해당 case로 점프하여 하단에 있는 코드를 실행합니다.

**default**

default 서브 명령 키워드는 switch에 입력된 값과 일치되는 case 값이 없을 때 처리하는 실행 코드입니다. 대부분 switch 처리 맨 하단에 위치하며, 이때 default 키워드와 콜론(:)만 작성합니다. 만일 일치되는 값이 없을 경우에는 프로그램은 default로 정의된 부분으로 점프하여 코드를 실행합니다.

멀티 분기 조건문 switch는 if 조건문보다 빠르게 조건을 분기하여 처리할 수 있습니다. 많은 양의 비교 판단하여 처리하는 조건이 있을 경우 if 명령문보다 자주 사용합니다. if문과 switch문을 비교해보면 단순한 == 비교에 대한 다수 조건 분기는 switch문이 유리합니다.

만일 if문을 사용하여 모든 조건을 비교한다고 하면 각각 비교되는 코드는 모두 실행될 것입니다. if문으로 작성한 조건 제어문이 있을 경우에 다음처럼 작성할 수 있습니다.

**예제 파일** switch-01.php

|  |
| --- |
| <?php  $n = "label2";  if($n == "label1"){  echo "n 값이 label1일 경우 실행됩니다.";  } else if($n == "label2"){  echo "n 값이 label2일 경우 실행됩니다.";  } else if($n == "label3"){  echo "n 값이 label3일 경우 실행됩니다.";  } else {  echo "n 값이 일치하는 게 없는 경우 실행됩니다.";  }  ?> |

**결과**

n 값이 label2일 경우 실행됩니다.

입력된 값에 대해서 모든 조건을 하나씩 비교 처리하여 코드를 실행합니다. 만일, 일치하는 조건이 마지막에 있을 경우 모두 조건을 검사해야 처리 루틴이 모두 실행하는 최악의 결과를 발생하며 성능적인 부담감이 발생할 수도 있습니다.

단순한 == 일치하는 비교로 처리되는 if문은 switch문을 통해 다시 작성할 수 있습니다. switch문은 if문과 달리 동일한 값의 비교만 가능합니다. 특정한 값이 크다, 작다거나 같지 않다라는 산술적인 비교를 처리할 수 없습니다.

switch조건 분기 명령은 조건이 만족하는 곳으로 바로 점프하여 처리를 하기 때문에 빠른 실행과 결과를 처리할 수 있습니다. 중간에 일치하지 않는 코드들은 건너뛰고, 일치하는 작업만 수행을 합니다.

**예제 파일** switch-02.php

|  |
| --- |
| <?php  $n = "label1";    switch ($n) {  case label1:  echo "n 값이 label1일 경우 실행됩니다.";  break;  case label2:  echo "n 값이 label2일 경우 실행됩니다.";  break;  case label3:  echo "n 값이 label3일 경우 실행됩니다.";  break;  default:  echo "n 값이 일치하는 게 없는 경우 실행됩니다.";  }  ?> |

**결과**

n 값이 label1일 경우 실행됩니다.

switch조건 분기문은 입력된 값에 대해서 중괄호 { } 안에서 일치하는 case로 점프하여 break; 또는 제어문 종료 지점 }까지 실행합니다.

앞서 설명을 한 것처럼 case문에서는 >= 나 <= 등 크기를 비교하는 문장을 사용할 수 없습니다.

**8.9 break**

switch 조건 제어문을 설명하면서 빼놓을 수 없는 기능이 바로 break;입니다. break는 사전적으로 ‘깨다’, ‘부수다’라는 의미이지만 프로그램에서는 종료나 탈출(exit) 같은 의미를 가지고 있습니다.

break; 명령 키워드는 switch 문법에서 거의 필수적으로 꼭 사용되는 명령입니다. 또한 9장에서 배울 반복문에서도 종종 사용할 수 있는 명령입니다.

기본적으로 switch 조건에서 실행되는 코드의 시작은 case 또는 default로 서브 키워드로 정의된 부분입니다. 여기서 시작해서 실행의 끝은 swtich의 중괄호가 끝나는 } 기호를 만날 때까지입니다. 따라서 조건 실행의 끝을 의미하는 break; 조건 제어 흐름 명령어가 필요합니다.

switch 문에서는 break; 문장을 만나게 되면 switch의 중괄호 { }를 끝내고 다음 문장으로 실행 제어권이 이동됩니다.

break 키워드는 PHP, C 언어 이외의 대부분의 언어에서도 동일한 형태의 기능을 수행합니다.

브레이크 키워드는 단독으로 사용합니다. 또한 break 키워드와 명령 구분자 세미콜론(;)을 바로 붙여서 사용합니다.

**|문법|**

break; 즉, 프로그램이 실행 도중에 break;를 만나게 되면 중괄호 { }로 묶여 있는 하나의 그룹을 탈출(종료)하게 됩니다.

**예제 파일)** break-01.php

|  |
| --- |
| <?php  $n = 1;  switch ($n) {  case 1:  echo "n값는 ".$n."입니다.<br>";  break;    case 2:  echo "n값는 ".$n."입니다.<br>";  break;  case 3:  echo "n값는 ".$n."입니다.<br>";  break;  default:  echo "n값는 ".$n."입니다.<br>";  }    echo "switch문 종료";  ?> |

**결과**

n값는 1입니다.

switch문 종료

위의 예에서 $ 변수 값에 따라서 조건과 일치하는 문자열을 한 개만 출력합니다. 그 이유는 $n 값에 따라서 조건을 분기하고, 한 줄의 문자열을 출력한 다음에 break문이 있어서 switch문을 탈출하게 됩니다.

만일 break문을 모두 제거한다고 하면 출력값은 다음과 같습니다.

**예제 파일** break-02.php

|  |
| --- |
| <?php  $n = 1;  switch ($n) {  case 1:  echo "n값는 ".$n."입니다.<br>";    case 2:  echo "n값는 ".$n."입니다.<br>";    case 3:  echo "n값는 ".$n."입니다.<br>";    default:  echo "n값는 ".$n."입니다.<br>";  }    echo "switch문 종료";  ?> |

**결과**

n값는 1입니다.

n값는 1입니다.

n값는 1입니다.

n값는 1입니다.

switch문 종료

break문이 없기 때문에 조건 1, 2, 3과 default를 모두 처리하게 됩니다.

따라서 switch의 각각의 case문 뒤에는 조건 코드의 실행 종료 의미로 break;로 단락을 마감해야 합니다. 만일 switch문이 조건이 일치한 case로 점프하여 코드를 실행하는 데 break;를 만나지 않으면 switch 실행의 끝 중괄호 }가 나오는 시점까지 모두 실행됩니다.

그럼 위의 예제에서 label2 의 break;를 주석으로 하나 삭제해 보겠습니다. 다음 예제를 확인해서 실행해 보기를 바랍니다.

**예제 파일** break-03.php

|  |
| --- |
| <?php  $n = "label2";    switch ($n) {  case label1:  echo "n 값이 label1일 경우 실행됩니다.<br>";  break;  case label2:  echo "n 값이 label2일 경우 실행됩니다.<br>";  // break;  case label3:  echo "n 값이 label3일 경우 실행됩니다.<br>";  break;  default:  echo "n 값이 일치하는 게 없는 경우 실행됩니다.<br>";  }  ?> |

**결과**

n 값이 label2일 경우 실행됩니다.

n 값이 label3일 경우 실행됩니다.

위처럼 두 줄의 실행 결과가 출력될 것입니다. $n변수의 값의 label3인 경우는 바로 case label3로 분기하여 **echo "n 값이 label3일 경우 실행됩니다.";** 명령이 실행될 것입니다.

하지만 $n 변수의 값이 label2인 경우 case label2 **분기하여 echo "n 값이 label2일 경우 실행됩니다.";** 를 실행하고 **break;가 없기 때문**에 그 밑에 있는 case label3 부분도 같이 실행됩니다.

**echo "n 값이 label3일 경우 실행됩니다.";**

**break;**

그리고 case label3의 **break;**를 만나서 switch 문을 끝나게 되어 밖으로 빠져나오게 됩니다.

break;를 통해 switch의 조건 실행의 코드 영역을 유연하게 범위를 정할 수 있습니다.

**8.9 3항 연산자**

새롭게 등장하는 3항 연산자는 if조건문을 간략하게 약식 형태의 연산자 처럼 표현을 할 수 있는 방법입니다.

3항 연산자는 물음표(?) 기호와 콜론(:) 기호로 구성됩니다.

|문법|

|  |
| --- |
| 조건식 **?** true의 처리 로직 **:** false의 처리 로직 |

위와 같이 조건식?공식1:공식2 형태로 표현한 것을 3항 연산자라고 합니다. 3항 연산자는 두 개의 연산 기호를 사용합니다. 바로 ? 기호와 : 기호입니다.

물음표(?) 연산 기호 앞에 조건을 입력하거나 조건 값을 가지는 변수를 넣을 수 있습니다. 물음표(?) 뒤에는 조건에 따른 처리 로직을 작성합니다. 기본적으로 다음에 있는 조건은 참(true)일 경우 실행됩니다.

다음의 콜론(:) 기호는 else의 의미를 가집니다. 기존 if의 경우 참/거짓의 로직을 else 키워드로 구분했으나 3항 연산자에서는 else 대신에 콜론(:) 기호를 작성합니다.

콜론(:) 앞쪽에는 참(true) 처리 로직, 뒤쪽에는 거짓(false) 처리 로직을 작성합니다. 수식을 작성할 수도 있고 상수나 변수 값을 입력하여 값을 반환할 수 있습니다.

3항 연산자는 복잡한 다수의 조건을 처리하는 것이 아니라 간단한 값의 대입이나 처리하는 데 주로 사용을 합니다.

**예제 파일** if-11.php

|  |
| --- |
| <?php  $lang = "ko";  $title = ($lang == "ko") ? "한국어" : "korean";  echo $title;  ?> |

**결과**

한국어

위의 예제는 $lang 변수의 값이 “ko”인 경우에는 참(true) 조거인 “한국어” 문자열을 출력합니다. 그 이외의 경우에는 else의 의미인 콜론 기호 다음의 “Korean” 문자열을 출력합니다.

**8.9.1 참(true) 생략**

3항 연산자에서 만일 참(true) 조건이 없는 경우에는 참 조건의 식은 생략 가능합니다. 이 기능은 PHP 5.3부터 지원합니다.

|문법|

|  |
| --- |
| 조건식 **?:** false의 처리 로직 |

다음은 참 조건을 생략하는 예제입니다.

**예제 파일** if-12.php

|  |
| --- |
| <?php    $title = $lang?:"english";  echo $title;  echo "<br>";  $lang = "Korean";  $title = $lang?:"english";  echo $title;  ?> |

**결과**

english

Korean

위의 예제에는 두 번의 3항 연산자를 사용했습니다. 처음의 3항 연산자의 경우 $lang 변수의 값이 미지정되어 미정의된 상태입니다. 따라서 3항 연산자의 첫 번째 로직의 조건은 거짓(false)으로 false 조건인 “english” 문자열이 출력됩니다.

두 번째 3항 연산자의 경우 먼저 $lang 변수에 값을 미리 지정합니다. 따라서 두 번째 3항 연산자의 조건은 참(true)이 되어서 “Korean” 문자열을 출력합니다. 즉, $lang의 값이 반환됩니다.