5장

조건 처리 명령어

5.1 조건분기문

프로그램에서 어떤 조건에 따라서 수행해야 하는 일들이 달라지는 경우는 매우 빈번하게 발생한다. 따라서 프로그램언어라면 보통 명령 수행의 흐름을 바꾸는 제어 명령어가 마련되어 있으며 C++언어도 이를 위해서 조건 검사 명령과 반복 명령어가 있다.

if 명령문은 조건을 제어하기 위해서 사용된다. 문법은 다음과 같다.

```
if (조건식) {
   실행문1;
   …
    실행문n;
}
```

보다시피 아주 간단한 문장으로 조건문이 참인지 거짓인지 판별하여 조건이 참이면 바로 뒤의 중괄호 {…}로 묶인 실행문들을 수행하고, 참이 아니면 수행하지 않고 다음으로 넘어간다.

만약 실행문이 하나라면 굳이 중괄호로 묶을 필요는 없으나 가독성을 높이기 위해서 관례적으로 중괄호를 항상 사용한다. 예를 들면 다음과 같은 두 경우는 완전히 동일하지만 후자와 같이 프로그램하는 것이 바람직하다.

```
if (ca > cb)
   cx = 10;

if (cd > cb) {
   cx = 10;
}
```

if 문을 사용할 때는 몇 가지 주의할 점이 있다. 먼저 조건은 반드시 괄호로 감싸야 한다는 점이고 괄호 안의 조건은 참과 거짓을 판별할 수 있어야 한다. **C++** 언어는 내부적으로 정수 **0**을 거짓으로 취급하고 그 이외의 수는**(**보통은 **1**값**)** 모두 참으로 취급한다는 점을 유의하자. 아래의 예에서 sa=10이라는 대입문은 무조건 수행되고 예2에서 sb=sc라는 대입문은 절대로 수행되지 않는다. (왜?)

```
if (1) { //예1
   sa = 10;
}

if (0) { //예2
   sb = sc;
}
```

또한 조건문에서 가장 하기 쉬운 실수가 '=='를 '='로 잘못 사용하는 경우인데 이 경우 논리적인 버그가 발생하게 된다. 예를 들면

```
long la = 1, lb = 1, lc;
if (la = lb) {
   lc = 10;
}
```

위와 같은 경우에는 la = lb라는 표현식은 변수 la에 변수 lb값을 대입하고 그 자체로 변수 lb값인 1을 갖게 된다. 따라서 의도와 다르게 lc=10이라는 명령은 무조건 수행되게 된다. 만약 lb변수값이 0이라면 lc=10이라는 명령은 절대로 수행되지 않는다. 따라서 아래와 같이 프로그램을 수정해야 할 것이다.

```
long la=1, lb=1, lc;
if (la == lb) {
 lc = 10;
}
```

또한 실수하기 쉬운 예가 다음과 같다.

```
if (la == 10);
   lb = lc;
```

이 예의 경우 if 조건 다음의 세미콜론 ';'에 의해 수행문이 종료되기 때문에 조건과 관계없이 lb=lc명령이 수행된다. 실제 프로그래밍을 하다보면 쉽게 하는 실수이니 눈여겨보기 바란다.

다음의 두 예는 서로 다른 프로그램이다. 첫 번째 예는 괄호가 없기 때문에 if 조건이 첫 번째 문장에만 적용되어서 ia의 값과는 상관없이 sc++이 수행되지만, 두 번째 예는 ia가 10값일 때에만 sc++이 수행된다.

```
if (ia == 10)
sb++;
sc++;
```

```
if (ia == 10) {
   sb++;
   sc++;
}
```

이번에는 if 문과 항상 같이 다니는 else문에 대해서 알아보자. 기본적인 문법은 아래와 같다.

```
if (조건문) {
   명령1;
   ...
} else {
   명령2;
   ...
}
```

else문에 포함된 명령어집합은 if 조건이 거짓일 경우 수행된다. 또한 if와 else를 조금 확장해 보면 if ~ else if 문이 된다.

```
if (조건문1) {
   명령문1;
   ...
} else if (조건문2) {
   명령문2;
   ...
} else {
   명령문3;
   ...
}
```

조건문1이 참이면 명령문1을 수행하고 조건문1이 거짓이고 조건문2가 참이면 명령문2가 수행되며, 두 조건 다 거짓일 경우 명령문 3이 수행된다.

다음 예제는 하나의 정수를 입력받아서 3의 배수인지 아닌지를 판별하여 화면에 표시해주는 예제이다. 3의 배수라면 3으로 나눈 나머지가 0일 것이고 아니라면 3으로 나눈 나머지가 0이 아니라는 사실을 이용하면 쉽게 프로그램을 작성할 수 있다.

ex05-01.cpp

```
#include <stdio.h>
int main() {
   int ia;
   printf("Input an integer :");
   scanf("%d", &ia);
   if (ia%3 == 0) {
       printf("%d is multiple of 3.\n", ia);
   } else {
       printf("%d is NOT multiple of 3.\n", ia);
   }
}
```

```
Input an integer number :2
2 is NOT multiple of 3.
```

사용자가 입력받은 수의 절대값을 출력하는 프로그램 예를 들면 다음과 같다. 입력된 수가 양수냐 아니냐에 따라서 수행되는 일이 달라진다.

ex05-02.cpp

```
#include<stdio.h>
int main() {
      float fa;
      printf("input a number :");
      scanf("%f", &fa);
      if (fa>0) {
            printf("%f",fa);
      } else {
            printf("%f",-fa);
      }
}
```

실행 결과

```
input a number :-1.1
|-1.100000|=1.100000
```

이 프로그램에서 입력된 수가 양수이면 그대로 출력하고 음수라면 -1을 곱해서 출력하는 간단한 방법을 사용했다. 이 예와 같이 단순한 if-else문이라면 다음과 같이 조건연산자를 사용하는 것이 더 효율적이다.

```
#include<stdio.h>
int main() {
    float fa;
    printf("input a number :");
    scanf("%f", &fa);
    printf("|%f|=%f", (fa>0)? fa:-fa );
}
```

다음 예는 입력된 정수가 음수인지, 0인지, 양수인지를 판별하는 예이다. if-else문이 중첩되어 사용되었음을 눈여겨보아야 한다.

ex05-03.cpp

```
#include <stdio.h>
int main() {
   int ia;
   printf("Input an integer number : ");
   scanf("%d",&ia);
   if (ia < 0) {
       printf("%d is negative.\n", ia);
   } else if (ia > 0) {
       printf("%d is positive.\n", ia);
   } else {
       printf("%d is zero.\n", ia);
   }
}
```

```
Input an interger number : 0
0 is zero.
```

다음은 사용자로부터 입력 받은 문자 하나가 알파벳 소문자라면 'lower case' 라고 화면에 출력하는 예제이다.

ex05-04.cpp

```
#include <stdio.h>
int main() {
      char ch;
      scanf("%c", &ch);
      if ('a'<=ch && ch<='z' ){
            printf("lower case");
      }
}
```

```
g
lower case
```

이 예제에서 비교문 ('a'<=ch && ch<='z')는 (97<=ch && ch<=122) 와 동일하다. 문자는 내부적으로 아스키코드로 간주되기 때문이다.

위 예제에서 대문자의 경우 "upper case"라고 출력하고 숫자의 경우 'digit'이라고 출력하는 부분을 추가하면 다음과 같다.

ex05-05.cpp

```
#include <stdio.h>
int main() {
	char ch;
	scanf("%c", &ch);
	if ('a'<=ch && ch<='z' ){
		printf("lower case");
	} else if ('A'<=ch && ch<='Z' ){
		printf("upper case");
	} else if ('0'<=ch && ch<='9') {
		printf("digit");
	} else {
		printf("unknown");
	}
}
```

이와 같이 if - else if 문은 얼마든지 중첩하여 사용할 수 있다.

5.2 조건문 예제

5.2.1 예제 #1

여기에서는 if 문을 사용한 예제를 풀어보도록 하겠다.

1. 세 개의 float 형 숫자를 사용자에게 입력 받아서 가장 큰 수를 출력하는 프로그램을 작성하라.

세 float형 변수를 f1, f2, f3 라고 하고 가장 큰 수를 저장하는 변수를 fMax 라고 하 하자. 간단한 알고리듬을 다음과 같이 생각해 볼 수 있다.

(a) f1과 f2 중 큰 것을 fMax에 저장한다.

(b) f3이 fMax보다 크다면 fMax 값을 f3 값으로 갱신한다.

(c) fMax를 화면에 출력한다.

이것을 그대로 프로그램으로 구현하면 다음과 같다.

ex05-06.cpp

```
#include  <stdio.h>
int main() {

    float f1, f2, f3;
    printf("Input three numbers : ");
    scanf("%f,%f,%f",&f1, &f2, &f3);

    float fMax = (f1>f2)? f1:f2; //(a)를 구현
    if (f3 > fMax){ //(b)를 구현
         fMax = f3;
    }
    printf("The maximum value is %f.", fMax);//(c)를 구현
}
```

실행 예는 다음과 같다.

```
Input three numbers : 11,22.5,-10
The maximum value is 22.500000.
```

이 프로그램을 조금만 바꾸면 최소값을 구하는 프로그램으로 변경할 수 있다.

5.2.2 예제 #2

전 절의 예제와 약간 다른 다음 문제를 풀어보도록 하겠다..

2. 세 개의 float 형 숫자를 사용자에게 입력 받아서 가운데 수를 출력하는 프로그램을 작성하라.

이 문제는 앞의 문제보다는 한 번 더 생각해야 한다. 여기서도 세 float형 변수를 f1, f2, f3 라고 하고 결과값(중간 수)를 저장하는 변수를 fMid 라고 하고 임시값을 저장하는 변수명을 fTmp라고 하자. (Tmp는 temporary 를 줄인 단어임) 알고리듬은 다음과 같이 생각해 볼 수 있다.

(a) f1과 f2 중 작은 수를 fMid 에 저장하고 큰 수를 fTmp에 저장한다.
(b) f3가 fMid보다 크다면 현재 fMid 값이 최소값이라는 의미다. 따라서 f3와 fTmp중 작은 것이 중간값이므로 그것을 fMid에 저장한다.
(c) f3가 fMid 보다 작다면 (a)에서 구한 fMid 값이 중간값이다.
(d) fMid를 출력한다.

이제 이 알고리듬을 프로그램으로 작성해 보면 다음과 같다.

ex05-07.cpp

```
#include  <stdio.h>
int main() {

    float f1, f2, f3;
    printf("Input three numbers : ");
    scanf("%f,%f,%f", &f1, &f2, &f3);

    float fMid, fTmp;

    if (f1>f2) { // 알고리듬 (a)를 구현한 것임
          fMid = f2;
          fTmp = f1;
    } else {
          fMid = f1;
          fTmp = f2;
    }

    if (f3 > fMid) { // (b)를 구현한 것임
          fMid = (f3>fTmp)? fTmp:f3;
    }

    printf("The middle value is %f.", fMid);
}
```

위에서 보면 알고리듬 (c)가 성립한다면 두 번째 if 문이 실행되지 않고 바로 printf()문이 실행이 되므로 올바르게 작동하는 것을 이해할 수 있을 것이다. 실행 예는 다음과 같다.

```
Input three numbers : -11,22.5,100.345
The middle value is 22.500000.
```

5.3 다중조건 분기문

만약 if 명령으로 여러 가지의 경우를 따져서 수행하려고 하면 if ~ else 문이 다중으로 중첩되어 프로그램의 가독성을 떨어뜨린다. 예를 들어서 어떤 정수형 변수의 값이 0일 때, 1일 때, 2일 때, 등등에 수십 가지에 대해서 동작이 다르게 수행되어야 하는 경우 if ~ else 명령보다는 여기에서 소개할 switch ~ case 명령을 사용하는 것이 가독성이나 수행 속도 면에서 훨씬 효율적이다.

switch ~ case 명령의 기본적인 문법은 다음과 같다.

```
switch (정수형변수) {

   case 값1:
        명령문1;
        break;

   case 값2:
        명령문2;
        break;

   ...

   default:
        명령문n;
        break;
}
```

switch명령 바로 다음에 오는 변수가 case 다음의 값1이나 값2에 해당하는 값이 있는지 판별한 후에 해당하는 값이 있으면 거기에 속한 명령문을 수행한다. case뒤에 오는 것은 반드시 하나의 값이어야 하며 조건이나 여러 값은 올 수 없다. 해당하는 값이 없을 경우에는 default 로 설정된 '명령문n'을 수행하게 된다. 필요에 따라 default문은 생략할 수도 있다.

이 명령을 쓸 때 주의할 점은 다음과 같다.

1. **switch**문 뒤에 오는 변수는 반드시 정수형 (char, short, int, long과 각각의 unsigned형) 이어야 한다
2. case 에 포함된 명령어들의 끝에는 반드시 break문을 써야 한다.

다음 예제는 정수(명령)를 하나 입력받아서 1이면 “Robot turned left.”이라고 표시하고, 2라면 “Robot turned right.”라고 표시하고 3이면 “Robot stopped.”이라고 표시한다. 만약 1, 2, 3중 아무 것도 아니라면 “illegal command.”라고 표시하는 간단한 프로그램이다. 흔히 하기 쉬운 실수가 case문이 끝나는 곳에 break문을 빼먹는 것인데 초보자들은 유의해야 한다.

ex05-08.cpp

```
#include <stdio.h>
int main() {

   int iA;
   printf("1. Turn left.\n");
   printf("2. Turn right.\n");
   printf("3. Stop.\n ");
   printf("Choose one :");
   scanf("%d", &iA);

   switch(iA) {
       case 1:
           printf("Robot turned left.\n");
           break;
       case 2:
           printf("Robot turned right.\n");
           break;
       case 3:
           printf("Robot stopped.\n");
           break;
       default:
           printf("Illegal command.\n");
           break;
   }

}
```

```
1. Turn left.
2. Turn right.
3. Stop
Choose one : 2
Robot turned right.
Press any key to continue...
```

이 예제와 같이 세 가지 정도는 if ~ else 문으로 구현해도 상관 없으나 경우의 수가 많아 지는 곳에는 switch ~ case 문이 훨씬 더 효율적이다.

만약 두 개 이상의 정수값에 대해서 동일한 동작을 수행하는경우라면 다음과 같이 case문을 작성하면 된다.

```
case val1:
case val2:
    실행문
    break;
```

첫 번째 case문은 아무런 내용이 없다. 이 경우 조건값이 val1값이라면 실행문이 수행되고 종료된다. val2값이어도 동일한 동작을 수행한다. 따라서 조건값이 val1이거나 val2 일 경우 실행문이 실행되고 종료된다.

```
switch(n) {
    case -2:
    case -1:
        printf("valid but negative");
        break;
    case 0:
    case 1:
    case 2:
        printf("valid");
        break;
    default:
        printf("invalid");
        break;
}
```

이 예에서 만약 정수형 변수 n 이 -2 혹은 -1이라면 “valid but negative”라고 출력된다. 변수 n이 0, 1, 2 중 하나라면 “valid”라고 출력될 것이고 이 외의 값이라면 “invalid”라고 출력된다. 이 예를 if 문으로 다시 작성하면 다음과 같을 것이다.

```
if (n==-2 || n==-1) {
  printf("valid but negative");
} else if (n==0 || n==1 || n==2) {
  printf("valid");
} else {
  printf("invalid");
}
```

어느 쪽을 사용해도 되지만 일반적으로 if-else 문보다는 switch-cae문이 실행 효율이 더 좋다고 알려져 있으니 if문을 switch-case문으로 대체가 가능하다면 그것을 사용하는 것이 더 낫다.

5장 연습문제

[pb05-01] 사용자가 입력한 int형 변수 x와 y에 대해 x를 y로 나눈 몫(quotient)과 나머지(remainder)를 화면에 각각 출력하는 프로그램을 작성하라. 단, 나머지가 0일 경우 몫만 출력해야 한다.

[pb05-02] 나이(char형)를 입력받아서 11살 이하면 “child”, 12살 이상이고 19살 이하면 “teenager”, 20살 이상이면 “adult” 라고 출력하는 프로그램을 작성하라.

[pb05-03] 월소득(monthly income)을 int형으로 입력 받아서 그것에 대한 소득세(income tax)를 계산하여 출력하는 프로그램을 작성하라. 소득세율은 다음과 같다.

소득이 500,000원 미만은 0%
2,500,000원 미만은 0.5%
4,000,000원 미만은 1.0%
7,000,000원 미만은 2.0%,
그 이상은 4.0%

[pb05-04] 양의 정수(int형)를 입력받아서 1이면 "1st", 2라면 “2nd", 3이면 ”3rd" 나머지의 경우는 숫자 뒤에 “th”를 붙여서 화면에 출력하는 프로그램을 작성하라.

[pb05-05] 국영수 세 과목의 점수(float형)를 입력 받아서 그 평균을 출력한 후 평균 점수가 90점 이상이라면 “excellent!”라고 출력하는 프로그램을 작성하라.

[pb05-06] 각도(short형)를 하나 입력받아서 예각(acute angle, 직각 보다 작은 각도), 둔각(obtuse angle, 직각보다 큰 각도), 직각(right angle) 여부를 표시하도록 하라.

[pb05-07] 세 내각이 모두 예각인 삼각형을 예각 삼각형(acute triangle), 한 내각이 둔각인 삼각형을 둔각 삼각형(obtuse triangle) 그리고 한 각이 직각인 삼각형을 직각 삼각형(right triangle)이라고 한다. 세 내각(int형)을 입력받아서 어떤 삼각형인지를 출력하는 프로그램을 작성하라. 단 세 각의 합이 180도가 아니라면 “not a triangle” 이라고 출력해야 한다.

[pb05-08] 네 개의 int형 숫자를 입력받아서 두 개씩 서로 같을 경우 “bingo”라고 화면에 출력하는 프로그램을 작성하라. 예를 들어 10, 20, 20, 10 을 입력하면 “bingo”라 출력되어야 하고 10, 20, 20, 30 이라면 아무 것도 출력되지 않는다.

[pb05-09] 정수 a, b를(int형) 입력받아서 일차 방정식 $ax = b$ 의 해를 구하는 프로그램을 작성하라. 다음과 같은 경우들을 고려해야 한다.

(a) a가 0이고 b도 0인 경우 (부정, indeterminate)

(b) a가 0이고 b는 0이 아닌 경우 (불능, impossible)

(c) a도 b도 0이 아닌 경우 (x = b/a)