제2장

흐름과 제어

학습 목표

- 부울식
  - 식 세우기, 평가 & 우선순위 규칙
- 분기 메카니즘
  - if-else
  - Switch
  - 내포된 if-else
- 루프
  - While, do-while, for
  - 중첩 루프
- 파일 입출력의 소개

연산자의 우선순위 (1 of 2)

디스플레이 2.3 연산자 우선순위

::	영역 지정 연산자
.	도트 연산자
->	멤버 선택
[]	배열 인덱스
( )	함수 호출
++	후위 증가 연산자(변수 뒤에 위치)
--	후위 감소 연산자(변수 뒤에 위치)
++	전위 증가 연산자(변수 앞에 위치)
--	전위 감소 연산자(변수 앞에 위치)
!	not 연산자
-	단항 마이너스
+	단항 플러스
*	dereference(포인터가 가리키는 곳의 데이터)
&	주소
new	메모리 할당
delete	변수 반환
delete []	배열 반환
sizeof	객체 크기
( )	형 변환
*	곱하기
/	나누기
%	나머지
+	더하기
-	빼기
<<	삽입 연산(콘솔 출력)
>>	추출 연산(콘솔 입력)

가장 높은 우선순위
(먼저 수행)

연산자의 우선순위(2 of 2)

<	보다 작은
>	보다 큰
<=	보다 작거나 같은
>=	보다 크거나 같은
==	같은
!=	같지 않은
&&	논리적 and
||	논리적 or
=	할당 연산자
+=	더한 후에 할당
-=	뺀 후에 할당
*=	곱하기 후에 할당
/=	나누기 후에 할당
%=	나머지 연산 후에 할당
? :	조건 연산자
throw	예외를 발생시킴
,	콤마 연산자

가장 낮은 우선순위
(나중에 수행)

우선순위

- 논리 적용 전 연산
  - x + 1 > 2 || x + 1 < -3 의 뜻:
    - (x + 1) > 2 || (x + 1) < -3
- 단락회로 평가
  - (x >= 0) && (y > 1)
  - 증가 연산자에 주의할것!
    - (x > 1) && (y++)
- 부울 값으로 변환하기
  - 0이 아닌 값 → true
  - 0 → false

if-else 구문

- 형식 구문:

```
if (<boolean_expression>)
        <yes_statement>
else
        <no_statement>
```

- 각 선택문은 오직 하나의 실행문임을 명심할 것!
- 분기에 여러 가지 실행문을 넣고 싶다면 → 복합문을 이용

복합문의 실행

- 들여 쓰고 있음에 주의

```
if (myScore > yourScore)
{
    cout << "I win!\n";
    wager = wager + 100;
}
else
{
    cout << "I wish these were golf scores.\n";
    wager = 0;
}
```

switch문의 실행

예제

```
int vehicleClass;
double toll;
cout << "Enter vehicle class: ";
cin >> vehicleClass;

switch (vehicleClass)
{
    case 1:
        cout << "Passenger car.";
        toll = 0.50;
        break;
    case 2:
        cout << "Bus.";
        toll = 1.50;
        break;
    case 3:
        cout << "Truck.";
        toll = 2.00;
        break;
    default:
        cout << "Unknown vehicle class!";
}
```

만일 break를 생략하면, 승용차는 1.50달러를 지불한다.

조건 연산자

- “3항 연산자” 로 불리기도 한다.
  - 수식 안에 조건문을 내장하는 것 이 가능하다.
  - 근본적으로 if-else문의 표기법 변형이다.
  - 예시:

```
if (n1 > n2)
    max = n1;
else
    max = n2;
```

  - 다른 방법:

```
max = (n1 > n2) ? N1 : n2;
```

    - "?" 와":" 는 “삼항” 연산자를 구성한다.

루프

- C++에서 쓰이는 3가지 형태의 루프
  - while
    - 가장 유동적이다.
    - “제한적이지” 않다.
  - do-while
    - 가장 덜 유동적이다.
    - 항상 한번 이상 루프바디를 실행한다.
  - for
    - “숫자를 세는” 루프

콤마 연산자

- 여러 식의 리스트를 계산하고 마지막 식의 값을 반환한다.
- For 루프문에서 가장 자주 사용된다.
- 예시:

  first = (first = 2, second = first + 1);

  - first는 3의 값을 가진다.
  - second는 3의 값을 가진다.
- 어떤 순서로 식이 계산될지 장담할 수 없다.

break문과 continue문

- 제어의 흐름
  - 루프가 입력과 출력에서 어떻게 세련되고 깔끔한 제어의 흐름을 제공하는지 다시 살펴보자.
  - 드물게, 자연적인 흐름을 대체할 수 있다.
- break;
  - 루프를 즉시 탈출하도록 한다.
- continue;
  - 루프 몸체를 한차례 건너뛰고 순환한다.
- 이러한 문장들은 자연적인 흐름을 해친다.
  - 반드시 필요한 경우에면 사용된다!

파일 입력에 대한 소개

- cin을 사용하여 키보드로부터 입력받는 것과 매우 유사한 방식으로 파일을 읽어올 수 있다.
- 단 하나의 예만 여기서 다루었을 뿐, 더 많은 세부사항은 12장에서 다룬다.
  - 여러분이 텍스트 파일로부터 읽고 타이핑하기에 너무 많은 데이터를 처리할 수 있다면 충분하다.

파일 입력 예시(1 of 2)

- 아래와 같이 텍스트 파일 이름이 player.txt 인 것을 참고

디스플레이 2.10 샘플 텍스트 파일, player.txt, 플레이어의 높은 점수와 이름 저장하기

```
100510
Gordon Freeman
```

파일 입력 예시 (2 of 2)

디스플레이 2.11 디스플레이 2.10의 텍스트 파일을 읽는 프로그램

```
1   #include <iostream>
2   #include <fstream>
3   #include <string>

4   using namespace std;
5   int main( )
6   {
7       string firstName, lastName;
8       int score;
9       fstream inputStream;

10      inputStream.open("player.txt");

11      inputStream >> score;
12      inputStream >> firstName >> lastName;

13      cout  << "Name: " << firstName >> " "
14            << lastName << endl;
15      cout  << "Score: " << score << endl;
16      inputStream.close( );

17      return 0;
18  }
```

Sample Dialogue

실습 문제 – 문제해결능력 향상

- 마름모 출력

크기(3보다 큰 홀수) : 5

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

크기(3보다 큰 홀수) : 9

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

- 구구단 자유롭게 출력

```
몇단까지 출력할까요? :12
몇까지 곱할까요? : 9
한줄에 몇단씩 출력할까요? :4
 1 x  1 =    1      2 x  1 =    2      3 x  1 =    3      4 x  1 =    4
 1 x  2 =    2      2 x  2 =    4      3 x  2 =    6      4 x  2 =    8
 1 x  3 =    3      2 x  3 =    6      3 x  3 =    9      4 x  3 =   12
 1 x  4 =    4      2 x  4 =    8      3 x  4 =   12      4 x  4 =   16
 1 x  5 =    5      2 x  5 =   10      3 x  5 =   15      4 x  5 =   20
 1 x  6 =    6      2 x  6 =   12      3 x  6 =   18      4 x  6 =   24
 1 x  7 =    7      2 x  7 =   14      3 x  7 =   21      4 x  7 =   28
 1 x  8 =    8      2 x  8 =   16      3 x  8 =   24      4 x  8 =   32
 1 x  9 =    9      2 x  9 =   18      3 x  9 =   27      4 x  9 =   36

 5 x  1 =    5      6 x  1 =    6      7 x  1 =    7      8 x  1 =    8
 5 x  2 =   10      6 x  2 =   12      7 x  2 =   14      8 x  2 =   16
 5 x  3 =   15      6 x  3 =   18      7 x  3 =   21      8 x  3 =   24
 5 x  4 =   20      6 x  4 =   24      7 x  4 =   28      8 x  4 =   32
 5 x  5 =   25      6 x  5 =   30      7 x  5 =   35      8 x  5 =   40
 5 x  6 =   30      6 x  6 =   36      7 x  6 =   42      8 x  6 =   48
 5 x  7 =   35      6 x  7 =   42      7 x  7 =   49      8 x  7 =   56
 5 x  8 =   40      6 x  8 =   48      7 x  8 =   56      8 x  8 =   64
 5 x  9 =   45      6 x  9 =   54      7 x  9 =   63      8 x  9 =   72

 9 x  1 =    9     10 x  1 =   10     11 x  1 =   11     12 x  1 =   12
 9 x  2 =   18     10 x  2 =   20     11 x  2 =   22     12 x  2 =   24
 9 x  3 =   27     10 x  3 =   30     11 x  3 =   33     12 x  3 =   36
 9 x  4 =   36     10 x  4 =   40     11 x  4 =   44     12 x  4 =   48
 9 x  5 =   45     10 x  5 =   50     11 x  5 =   55     12 x  5 =   60
 9 x  6 =   54     10 x  6 =   60     11 x  6 =   66     12 x  6 =   72
 9 x  7 =   63     10 x  7 =   70     11 x  7 =   77     12 x  7 =   84
 9 x  8 =   72     10 x  8 =   80     11 x  8 =   88     12 x  8 =   96
 9 x  9 =   81     10 x  9 =   90     11 x  9 =   99     12 x  9 =  108
```

실습 문제 – 문제해결능력 향상

- 입력된 숫자 이하의 완전수를 출력하는 프로그램 작성
  - 완전수란 자신을 제외한 약수들의 합(6=1+2+3)으로 자신이 계산되는 수
  - 배열 쓰지 말고, 비효율적이라도 반복문 반복해서 사용
  - 아래와 똑같이 출력되게 할 것(“1+2+3+”처럼 출력되면 안 됨)

```
입력 : 10
6(1+2+3)
```

```
입력 : 32
6(1+2+3)
28(1+2+4+7+14)
```

중첩 반복문을 빠져나가는 여러가지 방법

1. break 쓰기

2. 반복 변수 변경하기

3. flag 변수 사용하기

4. goto 쓰기
   – 최고 수준의 고수(高手)가 되면 사용? 아니 그래도 하지 말것.

```
flag = 1 ;
for (i = 0 ;  i < MAX1 ; i++)
    for ( j = 0 ; j < MAX2 && flag ; j++)
        for ( k = 0 ; k < MAX3 && flag ; k++)
        {
            ....
            if (CONDITION)
            {
                j = MAX2 ; i = MAX1 ;
                flag = 0 ;
                break ;
            }
        }
```