430.211 Programming Methodology (프로그래밍 방법론)

Flow of Control

Lab 1 Week 2

Fall 2025

M.IN.D Lab

TA info

- TA Info
  - Yongho Kim(peterkim98@snu.ac.kr)
  - Juhyeon Park (parkjh9229@naver.com)
  - Jung Yoon Hwang(danieljyh@snu.ac.kr)

Outline

- Class Review
  - Boolean Expressions
  - Branching Mechanisms
  - Loops
- Assignment
- Attendance Check

Outline

- Class Review
  - Boolean Expressions
  - Branching Mechanisms
  - Loops
- Assignment
- Attendance Check

Boolean Expressions

- A **Boolean data type variable** can store either of the values '**true**' or '**false**'.
- **Boolean expressions** are used to control branching and looping statements.

Boolean Expressions

- Logical operators
  - '!' : NOT
  - '&&' : AND
  - '||' : OR
- Comparison operators
  - '==' : Equal to
  - '! =' : Not equal to
  - '<' ('<=') : Less than (Less than or equal to)
  - '>' ('>=') : Greater than (Greater than or equal to)

Boolean Expressions

- Exercise 1

```cpp
#include <iostream>
using namespace std;

int main(){
    int x = 3, y = 4, z = 5, r=0;
    bool result; // A Boolean data type variable
    result = (x < y) && (r < z); // A Boolean expression
    cout << "result : " << result << endl;
    return 0;
}
```

This prints "1" as C++ converts 'True' to '1'

Boolean Expressions

Short Circuit Evaluation

```
if ( (kids != 0) && ((pieces/kids) >= 2) )
    cout << "Each child may have two pieces!";
```

Boolean Expressions

Short Circuit Evaluation

```
if ( (kids != 0) && ((pieces/kids) >= 2) )
    cout << "Each child may have two pieces!";
```

Prevent
"Zero division error"

Boolean Expressions

Short Circuit Evaluation

```
if ( (kids != 0) && ((pieces/kids) >= 2) )
    cout << "Each child may have two pieces!";
```

Prevent
"Zero division error"

Do not use x>y>z

Use (x>y) && (y>z)

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7    int x,y,z;
8    x=1;
9    y=2;
10   z=2;
11   cout << (x<y<z) << endl;
12   return 0;
13 }
```

Boolean Expressions

- Precedence of Operators

Highest precedence

::	Scope resolution operator
.	Dot operator
->	Member selection
[]	Array indexing
( )	Function call
++	Postfix increment operator (placed after the variable)
--	Postfix decrement operator (placed after the variable)
++	Prefix increment operator (placed before the variable)
--	Prefix decrement operator (placed before the variable)
!	Not
-	Unary minus
+	Unary plus
*	Dereference
&	Address of
new	Create (allocate memory)
delete	Destroy (deallocate)
delete []	Destroy array (deallocate)
sizeof	Size of object
( )	Type cast
*	Multiply
/	Divide
%	Remainder (modulo)
+	Addition
-	Subtraction
<<	Insertion operator (console output)
>>	Extraction operator (console input)
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal
!=	Not equal
&&	And
||	Or
=	Assignment
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modulo and assign
? :	Conditional operator
throw	Throw an exception
,	Comma operator

Lowest precedence

M.IN.D Lab

Boolean Expressions

- What if we **change** these **two lines** as below?

bool result;
result = $(x < y)\ \&\&(r < z)$;

bool result = $!\,x$;

bool result = $(!\,0 * x)$;

int result = $(!\,0 * x)$;

bool result = $(x < 5)||(x/r)$;

bool result = $(x\ \&\&\ y) + (!\,z)$;

?

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5       int x = 3, y = 4, z = 5, r=0;
6       bool result; // A Boolean data type variable
7       result = (x < y) && (r < z); // A Boolean expression
8       cout << "result : " << result << endl;
9       return 0;
10  }
```

All non-zero values → True

Zero value → False

Boolean Expressions

- What if we **change** these **two lines** as below?

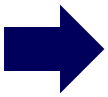
bool result = $!x$;	result = 0
bool result = $(!0 * x)$;	result = 1
int result = $(!0 * x)$;	result = 3
bool result = $(x < 5)\|\|(x/r)$;	result = 1
bool result = $(x \&\& y) + (!z)$;	result = 1

Branching Mechanisms

- If-else statement
  - Chooses one of two based on the '**Boolean expression**'.
  - If true, '**Yes statement**' is excuted.
  - Else, '**No statement**' is excuted. (← Optional)

Ex.2

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5
6       int x = 3;
7
8       if (x == 1)
9       {
10          cout << "Yes statement" << endl;
11      }
12      else
13          cout << "No statement" << endl;
14
15      return 0;
16  }
```



```
No statement

터미널이 작업에서 다시 사용됩니다.
닫으려면 아무 키나 누르세요.
```

Branching Mechanisms

- If-else statement
  - Chooses one of two based on the '**Boolean expression**'.
  - If true, '**Yes statement**' is excuted.
  - Else, '**No statement**' is excuted. (← Optional)

Ex.2

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5
6       int x = 3;
7
8       if (x == 1)
9       {
10          cout << "Yes statement" << endl;
11      }
12      else
13          cout << "No statement" << endl;
14
15      return 0;
16  }
```

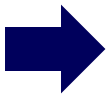
What if we make a mistake (e.g., x=1 ?)

Branching Mechanisms

- If-else statement
  - In C++, an assignment statement **returns** a transferred value.
  - Recall that **nonzero numbers** are converted to **true.**
  - Unlike '$x = 1$', '$1 = x$' will produce an **error message**.

Ex.2

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5
6       int x = 3;
7
8       if (x = 1)
9       {
10          cout << "Yes statement" << endl;
11      }
12      else
13          cout << "No statement" << endl;
14
15      return 0;
16  }
```



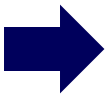
Code	Result
"$x == 1$"	No statement
"$x = 1$"	Yes statement
"$1 == x$"	No statement
"$1 = x$"	Error

Branching Mechanisms

- If-else statement
  - Check that **nonzero numbers** are converted to **true.**

Ex.3

```cpp
#include <iostream>
using namespace std;

int main(){
    int i;
    double d;

    cout << true << endl;
    cout << false << endl;
    cout << "-------" << endl;
    i = 0;
    if(i){cout << "True" << endl;} else{cout << "False" << endl;}
    i = 1;
    if(i){cout << "True" << endl;} else{cout << "False" << endl;}
    i = -1;
    if(i){cout << "True" << endl;} else{cout << "False" << endl;}
    cout << "-------" << endl;
    d = 0.1;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}
    d = 0;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}
    i = d;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}
    d = 1.2;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}
    d = -1.2;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}

    return 0;
}
```



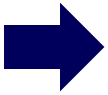
?

Branching Mechanisms

- If-else statement
  - Check that **nonzero numbers** are converted to **true.**

Ex.3

```cpp
#include <iostream>
using namespace std;

int main(){
    int i;
    double d;

    cout << true << endl;
    cout << false << endl;
    cout << "-------" << endl;
    i = 0;
    if(i){cout << "True" << endl;} else{cout << "False" << endl;}
    i = 1;
    if(i){cout << "True" << endl;} else{cout << "False" << endl;}
    i = -1;
    if(i){cout << "True" << endl;} else{cout << "False" << endl;}
    cout << "-------" << endl;
    d = 0.1;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}
    d = 0;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}
    i = d;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}
    d = 1.2;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}
    d = -1.2;
    if(d){cout << "True" << endl;} else{cout << "False" << endl;}

    return 0;
}
```



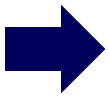
```
1
0
-------
False
True
True
-------
True
False
False
True
True

터미널이 작업에서 다시 사용됩니다.
 닫으려면 아무 키나 누르세요.
```

Branching Mechanisms

- Multiway if-else statement
  - Chooses **one of many** statements.
  - Boolean expressions are evaluated **in order.**
  - Action corresponding to the **first true** is executed.

Ex.4

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5
6       int x = 3, y = 5, z = 4;
7
8       if (x == y){
9           cout << "case 1" << endl;
10      }
11      else if (x == z){
12          cout << "case 2" << endl;
13      }
14      else {
15          cout << "case 3" << endl;
16      }
17      return 0;
18  }
```



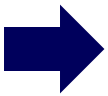
?

Branching Mechanisms

- Multiway if-else statement
  - Chooses **one of many** statements.
  - Boolean expressions are evaluated **in order.**
  - Action corresponding to the **first true** is executed.

Ex.4

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5
6       int x = 3, y = 5, z = 4;
7
8       if (x == y){
9           cout << "case 1" << endl;
10      }
11      else if (x == z){
12          cout << "case 2" << endl;
13      }
14      else {
15          cout << "case 3" << endl;
16      }
17      return 0;
18  }
```



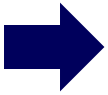
```
case 3

터미널이 작업에서 다시 사용됩니다.
닫으려면 아무 키나 누르세요.
```

Branching Mechanisms

- Switch statement
  - Also implements **multiway branches.**
  - Determined by a **controlling expression.**
  - Ends when either a **'break'** or the end is reached.

Ex.5

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5
6       int x;
7       cout << "Please enter an integer :" << endl;
8       cin >> x;
9
10      switch(x){
11          case 1:
12              cout << "You entered 1." << endl;
13              break;
14          case 2:
15              cout << "You entered 2." << endl;
16              break;
17          default :
18              cout << "Please enter either 1 or 2." << endl;
19      }
20      return 0;
21  }
```



```
Please enter an integer :
1
You entered 1.

터미널이 작업에서 다시 사용됩니다.
닫으려면 아무 키나 누르세요.
```

```
Please enter an integer :
3
Please enter either 1 or 2.

터미널이 작업에서 다시 사용됩니다.
닫으려면 아무 키나 누르세요.
```

Branching Mechanisms

- Switch statement
  - Also implements **multiway branches.**
  - Determined by a **controlling expression.**
  - Ends when either a **break** or the **end** is reached.

Ex.5

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5
6       int x;
7       cout << "Please enter an integer :" << endl;
8       cin >> x;
9
10      switch(x){
11          case 1:
12              cout << "You entered 1." << endl;
13              //break;
14          case 2:
15              cout << "You entered 2." << endl;
16              break;
17          default :
18              cout << "Please enter either 1 or 2." << endl;
19      }
20      return 0;
21  }
```

What if we remove the break and enter an integer 1 ?

Branching Mechanisms

- Switch statement
  - Also implements **multiway branches.**
  - Determined by a **controlling expression.**
  - Ends when either a **break** or the **end** is reached.

Ex.5

```
#include <iostream>
using namespace std;

int main(){

    int x;
    cout << "Please enter an integer :" << endl;
    cin >> x;

    switch(x){
        case 1:
            cout << "You entered 1." << endl;
            //break;
        case 2:
            cout << "You entered 2." << endl;
            break;
        default :
            cout << "Please enter either 1 or 2." << endl;
    }
    return 0;
}
```

What if we remove the break and enter an integer 1 ?

```
Please enter an integer :
1
You entered 1.
You entered 2.
```

Branching Mechanisms

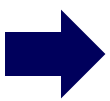
- Switch statement
  - Can be used with enumeration types

```cpp
#include <iostream>
using namespace std;

enum Animal{
    dog=6, cat=-2, rabbit
};

int main() {
    Animal animal = rabbit; //열거형 값 할당
    cout << "animal: " << animal << endl;

    switch(animal) {
        case dog:
            cout << "dog 입니다" << endl;
            break;
        case cat:
            cout << "cat 입니다" << endl;
            break;
        case rabbit:
            cout << "rabbit 입니다" << endl;
            break;
        default:
            cout << "Not an animal!" << endl;
    }
    return 0;
}
```

Branching Mechanisms

- Switch statement
  - **Can be used with enumeration types**

```cpp
#include <iostream>
using namespace std;

enum Animal{
    dog=6, cat=-2, rabbit
};

int main() {
    Animal animal = rabbit; //열거형 값 할당
    cout << "animal: " << animal << endl;

    switch(animal) {
        case dog:
            cout << "dog 입니다" << endl;
            break;
        case cat:
            cout << "cat 입니다" << endl;
            break;
        case rabbit:
            cout << "rabbit 입니다" << endl;
            break;
        default:
            cout << "Not an animal!" << endl;
    }
    return 0;
}
```



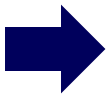
```
animal: -1
rabbit 입니다
```

Loops

- While statement
  - Loops through a block of code as long as a condition is **true.**
  - Boolean expression is checked before the body is executed.
  - Pay attention to the unintended introduction of **infinite loop.**

Ex.6

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5       int countDown;
6       cout << "Enter an integer : " << endl;
7       cin >> countDown; //Initialization
8
9       while (countDown > 0) //Loop condition
10      {
11          cout << "countDown is : " << countDown << endl;
12          countDown -= 1; //Update expression
13      }
14      cout << "That's all !" << endl;
15      return 0;
16  }
```



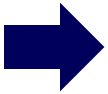
?

Loops

- While statement
  - Loops through a block of code as long as a condition is **true.**
  - Boolean expression is checked before the body is executed.
  - Pay attention to the unintended introduction of **infinite loop.**

Ex.6

```cpp
#include <iostream>
using namespace std;

int main(){
    int countDown;
    cout << "Enter an integer : " << endl;
    cin >> countDown; //Initialization

    while (countDown > 0) //Loop condition
    {
        cout << "countDown is : " << countDown << endl;
        countDown -= 1; //Update expression
    }
    cout << "That's all !" << endl;
    return 0;
}
```



```
Enter an integer :
3
countDown is : 3
countDown is : 2
countDown is : 1
That's all !

터미널이 작업에서 다시 사용됩니다.
 닫으려면 아무 키나 누르세요.
```

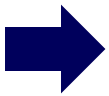
```
Enter an integer :
0
That's all !

터미널이 작업에서 다시 사용됩니다.
 닫으려면 아무 키나 누르세요.
```

Loops

- Do-while statement
  - Very similar to the **while statement.**

Ex.6

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5       int countDown;
6       cout << "Enter an integer : " << endl;
7       cin >> countDown; //Initialization
8
9       ?
10      {
11          cout << "countDown is : " << countDown << endl;
12          countDown -= 1; //Update expression
13      }                   ?
14      cout << "That's all !" << endl;
15      return 0;
16  }
```

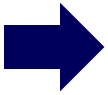


Loops

- Do-while statement
  - Very similar to the **while statement.**
  - Boolean expression is checked after the body is executed.
  - Do not forget the **final semicolon.**

Ex.7

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5       int countDown;
6       cout << "Enter an integer : " << endl;
7       cin >> countDown; //Initialization
8
9       do
10      {
11          cout << "countDown is : " << countDown << endl;
12          countDown -= 1; //Update expression
13      } while (countDown > 0); //Loop condition
14      cout << "That's all !" << endl;
15      return 0;
16  }
```



```
Enter an integer :
3
countDown is : 3
countDown is : 2
countDown is : 1
That's all !

터미널이 작업에서 다시 사용됩니다.
 닫으려면 아무 키나 누르세요.
```

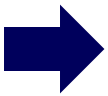
```
Enter an integer :
0
countDown is : 0
That's all !

터미널이 작업에서 다시 사용됩니다.
 닫으려면 아무 키나 누르세요.
```

Loops

- For statement
  - Can do anything that a **while loop** can do.

Ex.7

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5       //Declaratioin
6       int i, countDown;
7       cout << "Enter an integer : " << endl;
8       cin >> countDown;
9
10      //Initialization; Loop condition; Update expression
11      for (i=countDown; i > 0; i--)
12      {
13          cout << "countDown is : " << i << endl;
14      }
15      cout << "That's all !" << endl;
16
17      return 0;
18  }
```



```
Enter an integer :
3
countDown is : 3
countDown is : 2
countDown is : 1
That's all !

터미널이 작업에서 다시 사용됩니다.
 닫으려면 아무 키나 누르세요.
```

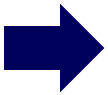
```
Enter an integer :
0
That's all !

터미널이 작업에서 다시 사용됩니다.
 닫으려면 아무 키나 누르세요.
```

Loops

- For statement
  - Can do anything that a **while loop** can do.
  - A variable can be **declared** and **initialized** in a for statement.
  - Can add **multiple actions** using the **comma operator.**

Ex.7

```
#include <iostream>
using namespace std;

int main(){
    int countDown;
    cout << "Enter an integer : " << endl;
    cin >> countDown;

    //(Multiple) Declaratioin & Initialization; Loop condition; Update expression
    for (int j=0, i=countDown; i > 0; i--, j++)
    {
        cout << "countDown is : " << i << endl;
        cout << "j : " << j << endl;
    }
    cout << "That's all !" << endl;

    return 0;
}
```



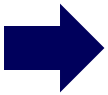
?

Loops

- For statement
  - Can do anything that a **while loop** can do.
  - A variable can be **declared** and **initialized** in a for statement.
  - Can add **multiple actions** using the **comma operator.**

Ex.7

```
#include <iostream>
using namespace std;

int main(){
    int countDown;
    cout << "Enter an integer : " << endl;
    cin >> countDown;

    //(Multiple) Declaratioin & Initialization; Loop condition; Update expression
    for (int j=0, i=countDown; i > 0; i--, ++j)
    {
        cout << "countDown is : " << i << endl;
        cout << "j : " << j << endl;
    }
    cout << "That's all !" << endl;

    return 0;
}
```



```
Enter an integer :
5
countDown is : 5
j : 0
countDown is : 4
j : 1
countDown is : 3
j : 2
countDown is : 2
j : 3
countDown is : 1
j : 4
That's all !
```

Outline

- Class Review
  - Boolean Expressions
  - Branching Mechanisms
  - Loops

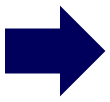
- Assignment

- Attendance Check

Assignment

- Problem
  - Calculate **positive divisors** of a given integer
  - Print out **which case** the number of divisors belongs to.
    - The number of positive divisors $\leq 3 \rightarrow$ Prints out "Case 1"
    - $4 \leq$ The number of positive divisors $\leq 10 \rightarrow$ Prints out "Case 2"
    - $11 \leq$ The number of positive divisors $\rightarrow$ Prints out "Case 3"
  - **Your code will be graded with sufficient testcases.**

User input
: 6 + Enter



```
Input number: 6
Positive divisors: 1 2 3 6
The number of positive divisors: 4
Which case the number of positive divisors belongs to? Case 2
```

Assignment

- Skeleton code

```cpp
#include <iostream>

using namespace std;

int main()
{

    int num, cnt=0;

    //----------Implement here(Read a number from user input)-----------//

    //-----------------------------------------------------//

    cout << "Input number: " << num << " " << endl;
    cout << "Positive divisors: " ;

    //-------------Implement here---------------//

    //-----------------------------------------//

    cout << "The number of positive divisors: " << cnt << endl;
    cout << "Which case the number of positive divisors belongs to? ";

    //-------------Implement here------------------//

    //-------------------------------------------//

    return 0;
}
```

Assignment

- Submission
  - Due date : 9/19 14:30
  - Push “제출” button on Elice

Outline

- Class Review
  - Boolean Expressions
  - Branching Mechanisms
  - Loops
  - Introduction to File Input
- Assignment
- Attendance Check

Attendance Check

- Problem
  1. Define enum 'ItemType'
     - SWORD, ARMOR, POTION
  2. Change offense, defense, health according to selected item.
  3. If invalid input : print "Invalid!"