한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

Introduction to Computers & Lab

Lab 09

2021.05.06

Prof. Muhammad Bilal
TA. Sohee Jang

Index

1. Review
    - Preprocessor
        - Include
        - Macro
        - Directives
    - 2D Array Manipulation

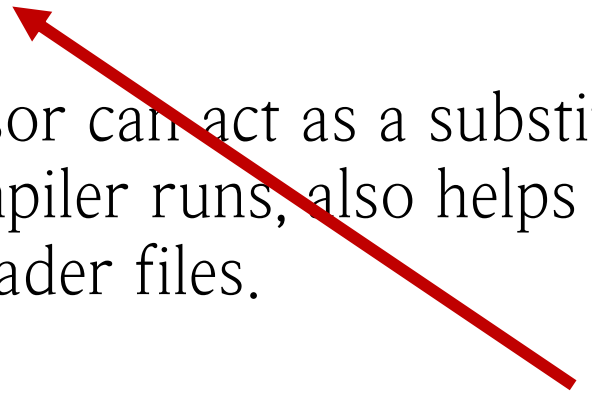
2. This week's Tasks + Hint

Preprocessor

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

- A preprocessor is a separate program that runs immediately before compilation. When the preprocessor is running, it looks for directives in each code file. Directives are code that starts with # and ends with a line break.

- The preprocessor can act as a substitute for simply manipulating text just before the compiler runs, also helps with debugging and prevents duplicate inclusion of header files.



Ex) #include

Include

We are familiar with the #include directive. When you do #include, the preprocessor copies the contents of the embedded file to the location of the indicator. (It was used for forward declaration.)

#include <filename>

#include "filename"

Macro

- Use #define
- Macros are rules that define how inputs are converted into outputs.
- Macros include object-like macros and function-like macros.
- A function-like macro works like a function.

```
#define identifier
#define identifier substitution_text
```

Example of using Macros


```cpp
#include <iostream>
#define FAVORITE_NUM 9
using namespace std;

int main() {
  cout << "My favorite number is : " << FAVORITE_NUM << endl;
  return 0;
}
```

```
My favorite number is : 9
```

* preprocessed

```cpp
cout << "My favorite number is : " << 9 << endl;
```

Preprocessor Directives

Conditional compilation preprocessing indicators allow you to specify conditions to compile or conditions not to compile.

#if
#else
#endif
#elif
#ifdef
#ifndef
#error

#ident
#import
#line
#machine
#system
#warning

Conditional compilation


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

int main() {
  #ifdef PRINT_JOE
  cout << "Joe" << endl;
  #endif

  #ifdef PRINT_BOB
  cout << "Bob" << endl;
  #endif

  return 0;
}
```

```
Joe
```

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

int main() {
  #ifdef PRINT_JOE
  cout << "Joe" << endl;
  #endif

  #ifndef PRINT_BOB
  cout << "Bob" << endl;
  #endif

  return 0;
}
```

```
Joe
Bob
```

Matrix – array[row][column]

- To prevent the function from modifying passed array elements, you can make the array const.

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

int main() {

  int array[3][5]=
  {
    {1,2,3,4,5},
    {6,7,8,9,10},
    {11,12,13,14,15}
  };

  return 0;
}
```

Column

ROW

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
	?			
			?	

Task 1 : Preprocessor Directive

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

Complete the code using preprocessor directive.
Try writing it using preprocessor direct without touching the main.

★ #if, #elif, #else, #endif, #define, #error

Task 2 : Trace

If the matrix you receive is a square matrix, → N X N
write a program to obtain the diagonal sum of the matrices.

Trace is not defined unless it is a square matrix.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

Task 3 : Inverse Matrix

Write a program that outputs an inverse matrix for the matrix you received input.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} A^{-1} = \begin{bmatrix} x & y \\ z & k \end{bmatrix}$$

$$AA^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x & y \\ z & k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \rightarrow A^{-1} = \begin{bmatrix} x & y \\ z & k \end{bmatrix} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Task 4 : Transpose matrix

Write a function that finds the transpose of the input matrix and returns the transpose matrix to the calling function.

★ transpose $a_{ij}^{\mathsf{T}} = a_{ji}$ example ->

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow A^{\mathsf{T}} = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

★ Be sure to declare a function.

$$B = \begin{bmatrix} x & y \\ z & w \end{bmatrix} \Rightarrow B^{\mathsf{T}} = \begin{bmatrix} x & z \\ y & w \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & 5 & -2 & 7 \end{bmatrix} \Rightarrow C^{\mathsf{T}} = \begin{bmatrix} 1 & -3 \\ 1 & 5 \\ 1 & -2 \\ 1 & 7 \end{bmatrix}$$