LAB 1

C++ BASICS & LINUX

EEEC 10008

National Yang Ming Chiao Tung University

Grading policy

- 2:
  - Hand in ***on time***
  - Program can be *compiled*, and the answer is *correct*
- 1:
  - Hand in ***within a week***
  - Program can be *compiled*, and the answer is *correct*
- 0.6:
  - Hand in ***more than one week***
  - Program can be *compiled*, and the answer is *correct*
- 0:
  - No submission

LAB

- Lab time : 18:30 – 21:20 every Monday
- Classroom : computer PC02, PC04
- Brief introduction before every lab
- Homework deadline is next Monday.

Outline

- C++ basics
- Linux basics

- Lab1 exercise
- HW1

C++ Identifiers

- An Identifier is a name of variables constant, …
- A C++ identifier
  - Consists of a sequence of letters, digits, and the underscore character (_)
  - Must start with either a letter or an underscore character // avoid doing so in general
  - Is case-sensitive

- Keywords are special identifiers
  - E.g., if, for, char, …
  - Cannot be used for user-defined entities

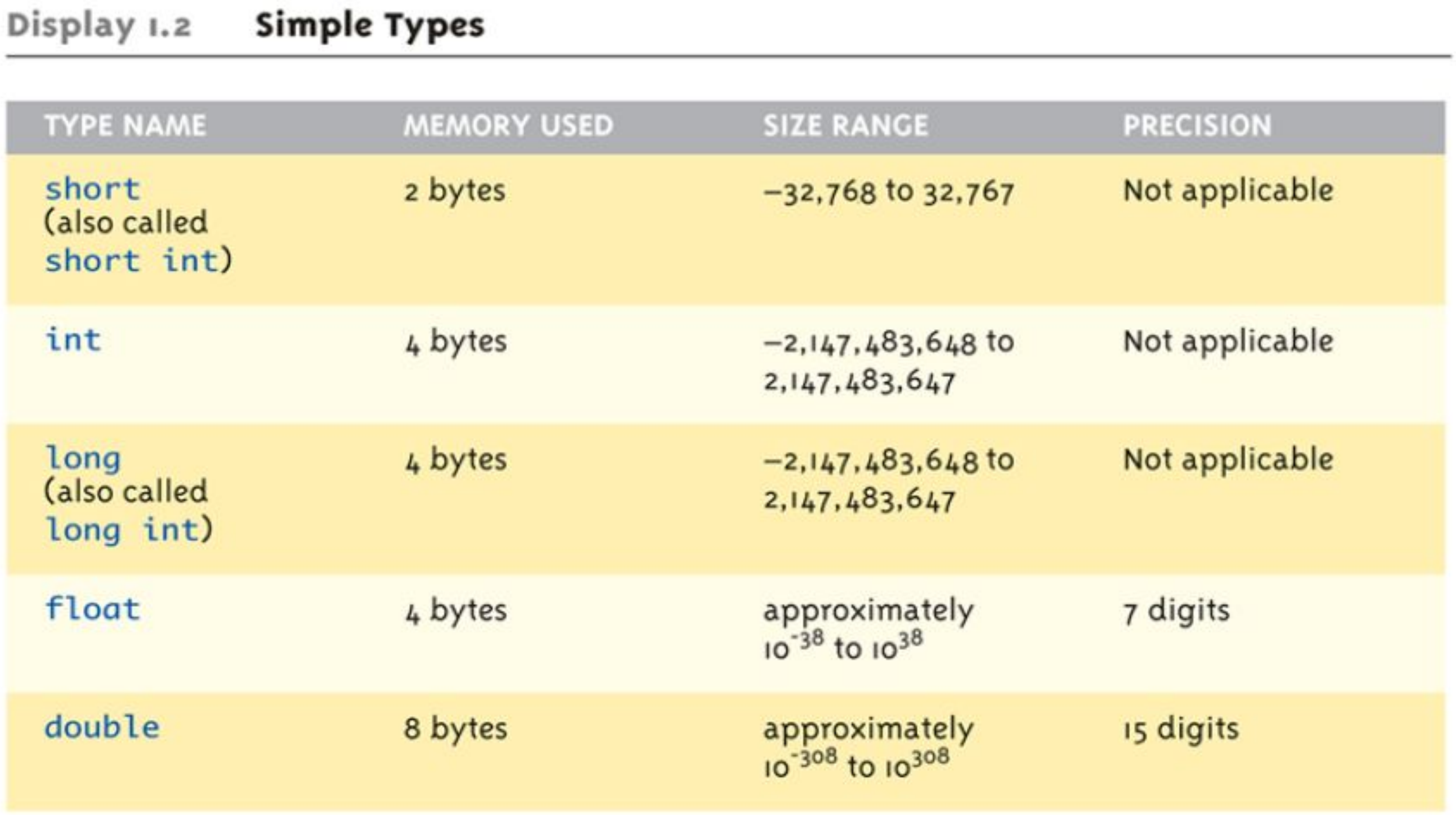
C++ Variables

- Variables
  - Its name is an identifier
  - is a memory location to store data
  - Must be declared before its use

```
  int number;              // declaration & definition
double width, length;  // declaration & definition
extern int count;        // declaration ONLY, discuss later
```

  - Meaningful names!
  - Naming convention: starting with a lowercase letter
    - E.g., weight, total_weight, …

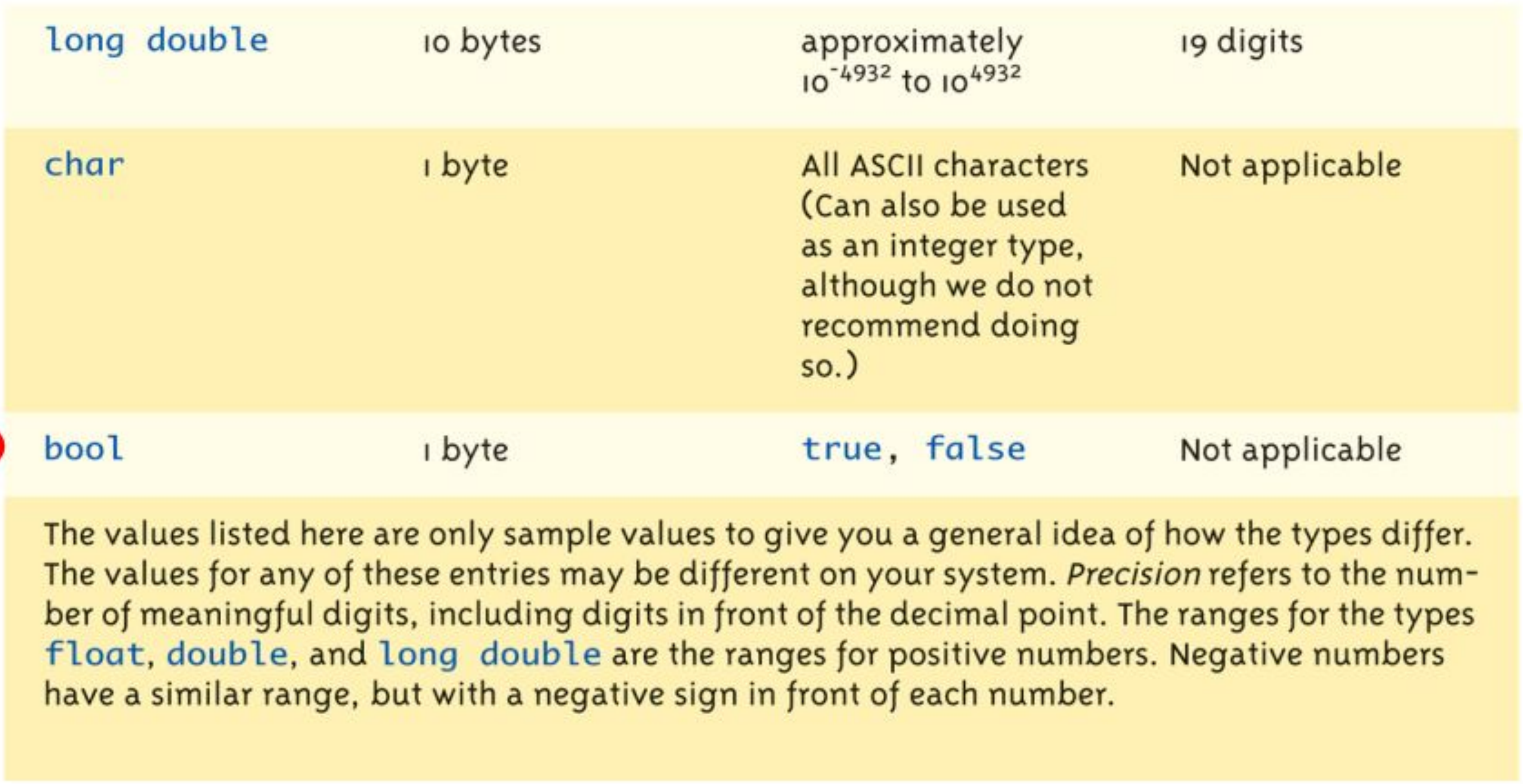
Fundamental Data Types (1/2)



Display 1.2 **Simple Types**

TYPE NAME	MEMORY USED	SIZE RANGE	PRECISION
`short` (also called `short int`)	2 bytes	−32,768 to 32,767	Not applicable
`int`	4 bytes	−2,147,483,648 to 2,147,483,647	Not applicable
`long` (also called `long int`)	4 bytes	−2,147,483,648 to 2,147,483,647	Not applicable
`float`	4 bytes	approximately 10^{-38} to 10^{38}	7 digits
`double`	8 bytes	approximately 10^{-308} to 10^{308}	15 digits

Fundamental Data Types (2/2)



`long double`	10 bytes	approximately 10^{-4932} to 10^{4932}	19 digits
`char`	1 byte	All ASCII characters (Can also be used as an integer type, although we do not recommend doing so.)	Not applicable
`bool`	1 byte	`true, false`	Not applicable

The values listed here are only sample values to give you a general idea of how the types differ. The values for any of these entries may be different on your system. *Precision* refers to the number of meaningful digits, including digits in front of the decimal point. The ranges for the types `float`, `double`, and `long double` are the ranges for positive numbers. Negative numbers have a similar range, but with a negative sign in front of each number.

Constants

```
double money;
money *= (1 + 0.05); //What is 0.05?
```

```
const double RATE = 0.05; //all uppercase letters
money *= (1 + RATE);  // better readability
RATE = 0.1;  // compilation error
```

- Named constants or declared constants (e.g., RATE)
  - Better readability and maintainability
  - Change attempts result in compilation errors!
- Named constants MUST be initialized

```
const int myWeight;  // compilation error!
```

Arithmetic Precision

- Examples:
  - 17 / 5 evaluates to 3 in C++!
    - Both operands are integers (Integer division)
  - 17.0 / 5 equals 3.4 in C++!
    - Highest-order operand is "double type“ (Double "precision" division)
  - int intVar1 =1, intVar2=2; intVar1 / intVar2;
    - Result: 0! (Integer division)
- Calculations done "one-by-one“
  - 1 / 2 / 3.0 / 4 performs 3 separate divisions.
    - First 1 / 2 equals 0
    - Then 0 / 3.0 equals 0.0
    - Then 0.0 / 4 equals 0.0!
- So not necessarily sufficient to change just "one operand" in a large expression

Type Casting

- Casting for Variables
  - C style
    - double dvar = (double) ivar;
  - C++ style
    - double dvar = static_cast<double>(ivar) ;
    - static_cast<type>(expression)
- Two kinds
  - implicit — also called "automatic"
    - done for you automatically
      17 / 5.5
      casting the 17 17.0
  - explicit type conversion
    - programmer specifies conversion with static_cast operator
      int m;
      static_cast<double>(m) / 5.5

Libraries

- C++ standard libraries
  - Input/output, math, strings, …
- #include <Library_Name>
  - directive to "add" contents of the specified library file to your program
  - called "preprocessor directive"
    - Executes before compilation, and simply “copies” library file into your program file

Namespaces

- Namespaces defined:
  - collection of name definitions

```
#include <iostream>
1. using namespace std;       // avoid this
 cout<<“Hello world!”;
2. using std::cout;
 cout<<“Hello world!”;
3. std::cout<<“Hello world!”;
```

  - includes entire standard library of name definitions

Console Input/Output

- I/O objects cin for input, cout for output, cerr for error output
- Defined in the C++ library called <iostream>
- Must have these lines (called pre-processor directives) near start of file:

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

  - Tells C++ compiler to use appropriate library so we can use the I/O objects cin, cout, cerr

Console Output

- Any data can be outputted to display screen
  - Variables
  - Constants
  - Literals
  - Expressions (which can include all of above)
- cout << numberOfGames << " games played.";
  - "value" of variable numberOfGames and literal string "games played." are outputted
- Cascading: multiple values in one cout
- New lines in output
  - cout << "Hello World\n";
  - cout << "Hello World" << endl;

Console Input

- cin >> num;
  - waits on-screen for keyboard entry
  - value entered at keyboard is "assigned" to num
- ">>" (extraction operator) points opposite
  - Think of it as "pointing toward where the data goes“
  - no literals allowed for cin
    - Must input to a variable
    - cin >> 23; // compilation error!

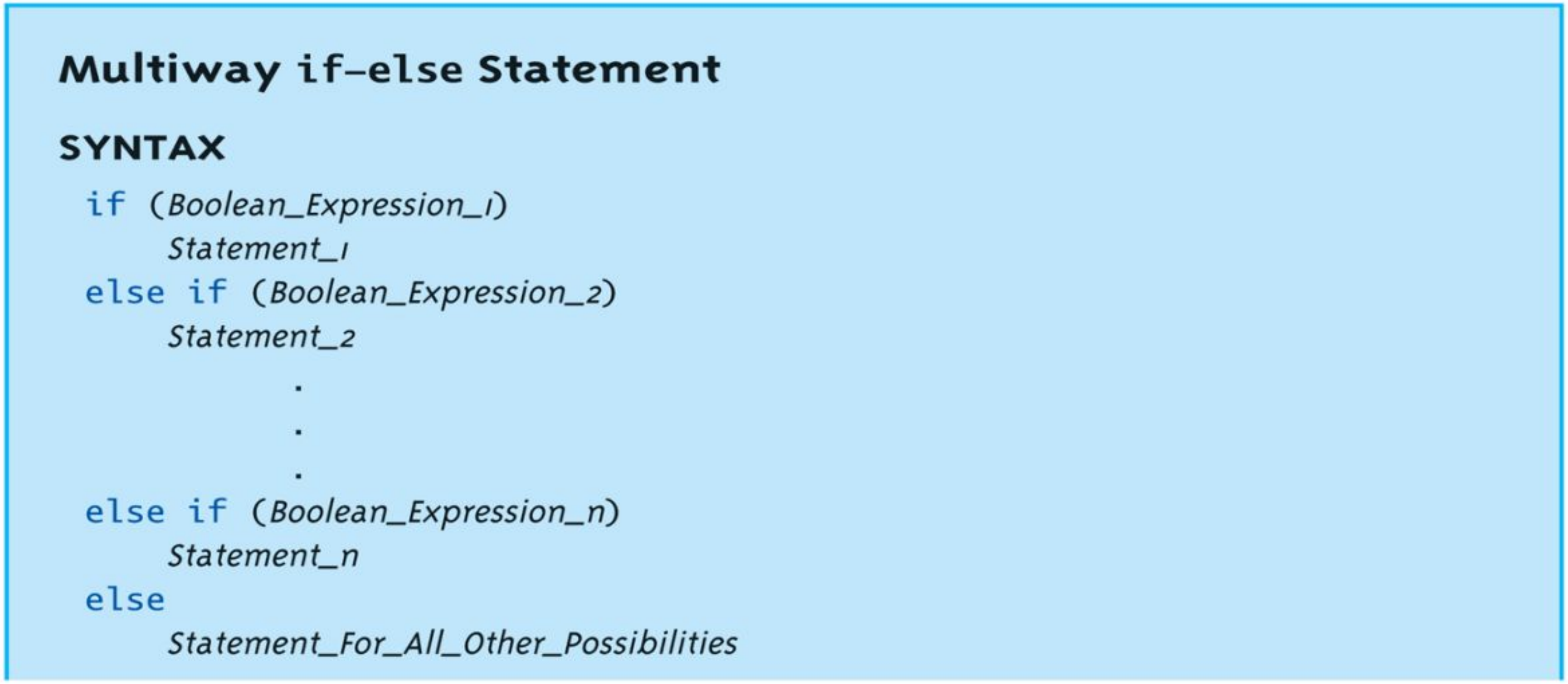
Branch Mechanisms

- if-else statements
  - Choice of two mutually exclusive statements based on condition expression
  - Syntax:

```
if(<Boolean_expression>){
  <true_statement>
}else{
  <false_statement>
}
```

Multiway if-else (1/2)

- Avoid “excessive” indenting
- Syntax :



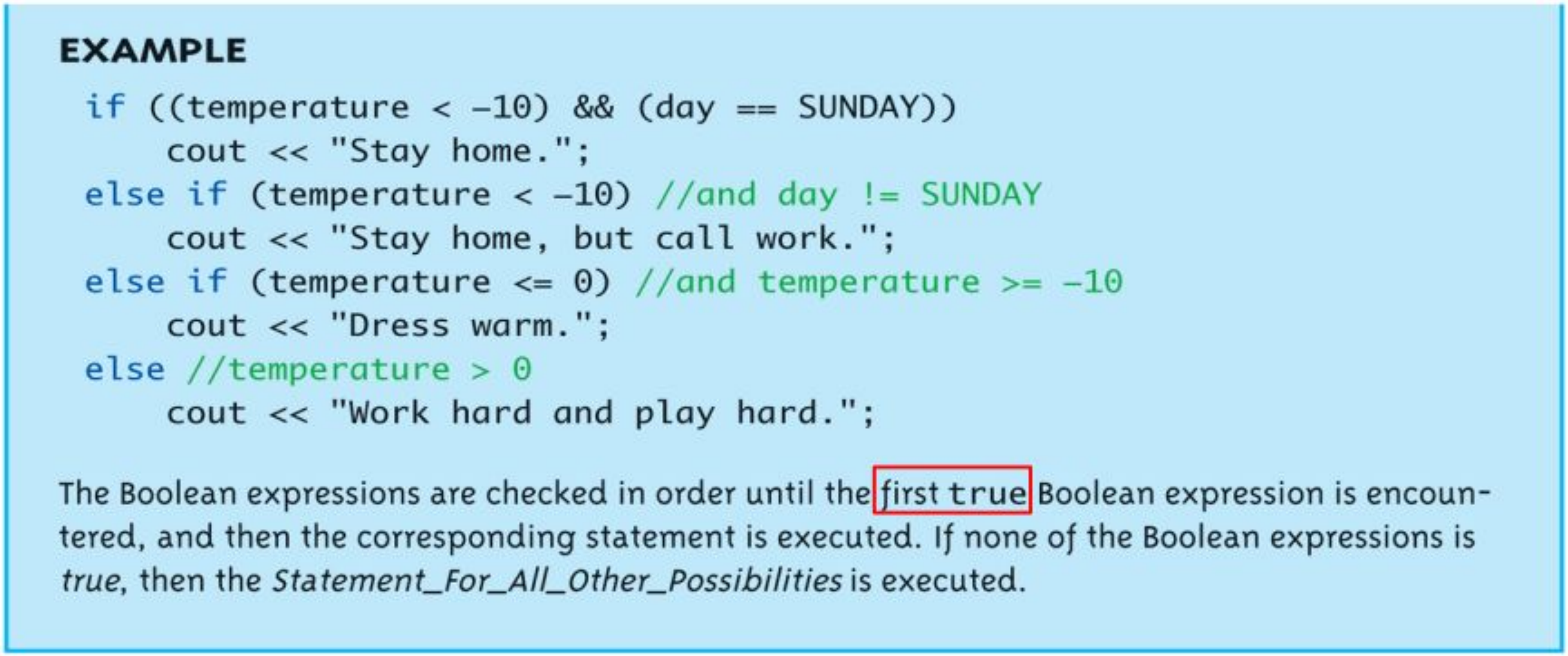
Multiway if-else Statement

SYNTAX

```
if (Boolean_Expression_1)
     Statement_1
else if (Boolean_Expression_2)
     Statement_2
              .
              .
              .
else if (Boolean_Expression_n)
     Statement_n
else
     Statement_For_All_Other_Possibilities
```

Multiway if-else (2/2)

- Example :



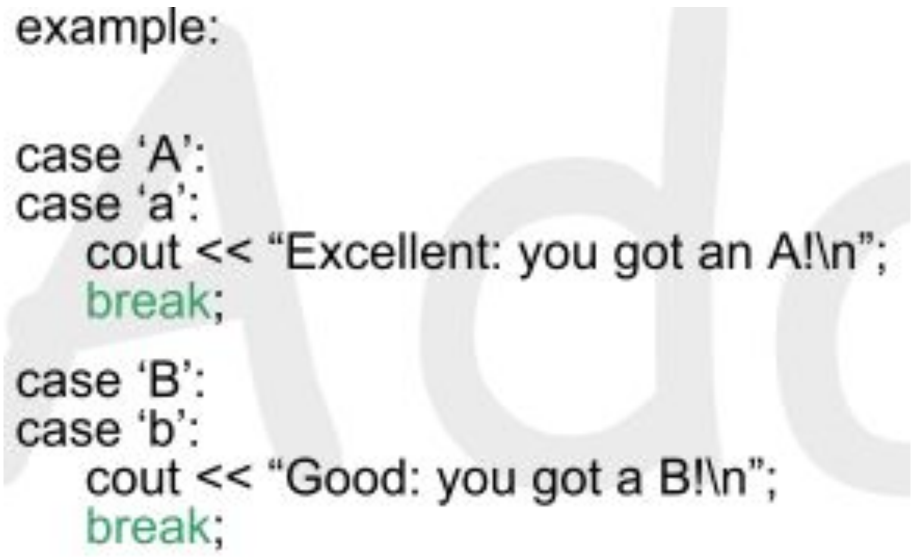
EXAMPLE

```
if ((temperature < -10) && (day == SUNDAY))
    cout << "Stay home.";
else if (temperature < -10) //and day != SUNDAY
    cout << "Stay home, but call work.";
else if (temperature <= 0) //and temperature >= -10
    cout << "Dress warm.";
else //temperature > 0
    cout << "Work hard and play hard.";
```

The Boolean expressions are checked in order until the first `true` Boolean expression is encountered, and then the corresponding statement is executed. If none of the Boolean expressions is *true*, then the *Statement_For_All_Other_Possibilities* is executed.

Switch Statement (1/3)

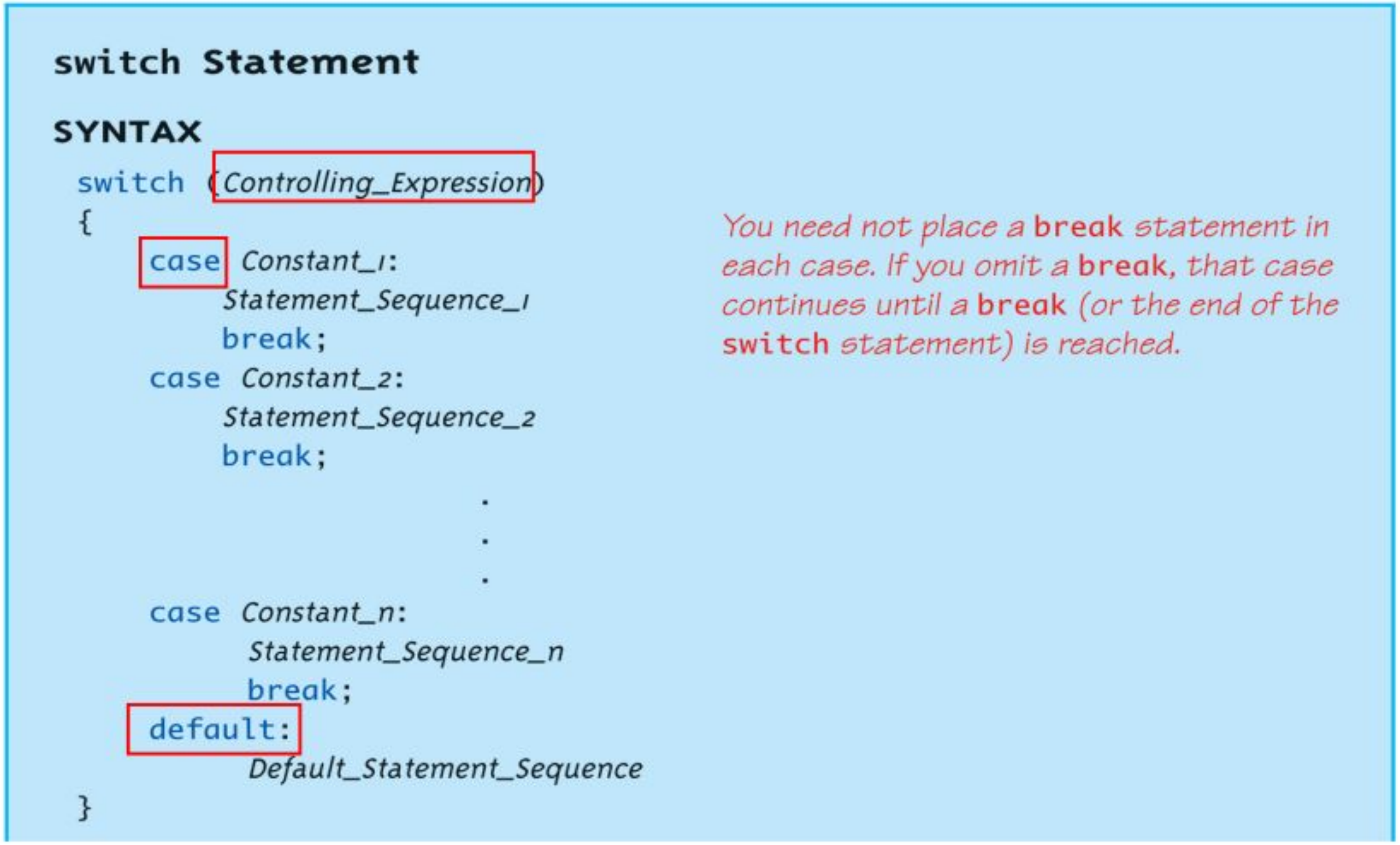
- Controlling expression MUST return an integral value
  - OK: char, int, bool, enum
  - not OK: float, double, …
- Case labels must also be integral values
- break and default are optional
- Execution "falls thru" until break



```
example:

case 'A':
case 'a':
    cout << "Excellent: you got an A!\n";
    break;
case 'B':
case 'b':
    cout << "Good: you got a B!\n";
    break;
```

Switch Statement (2/3)

- Syntax :



switch Statement

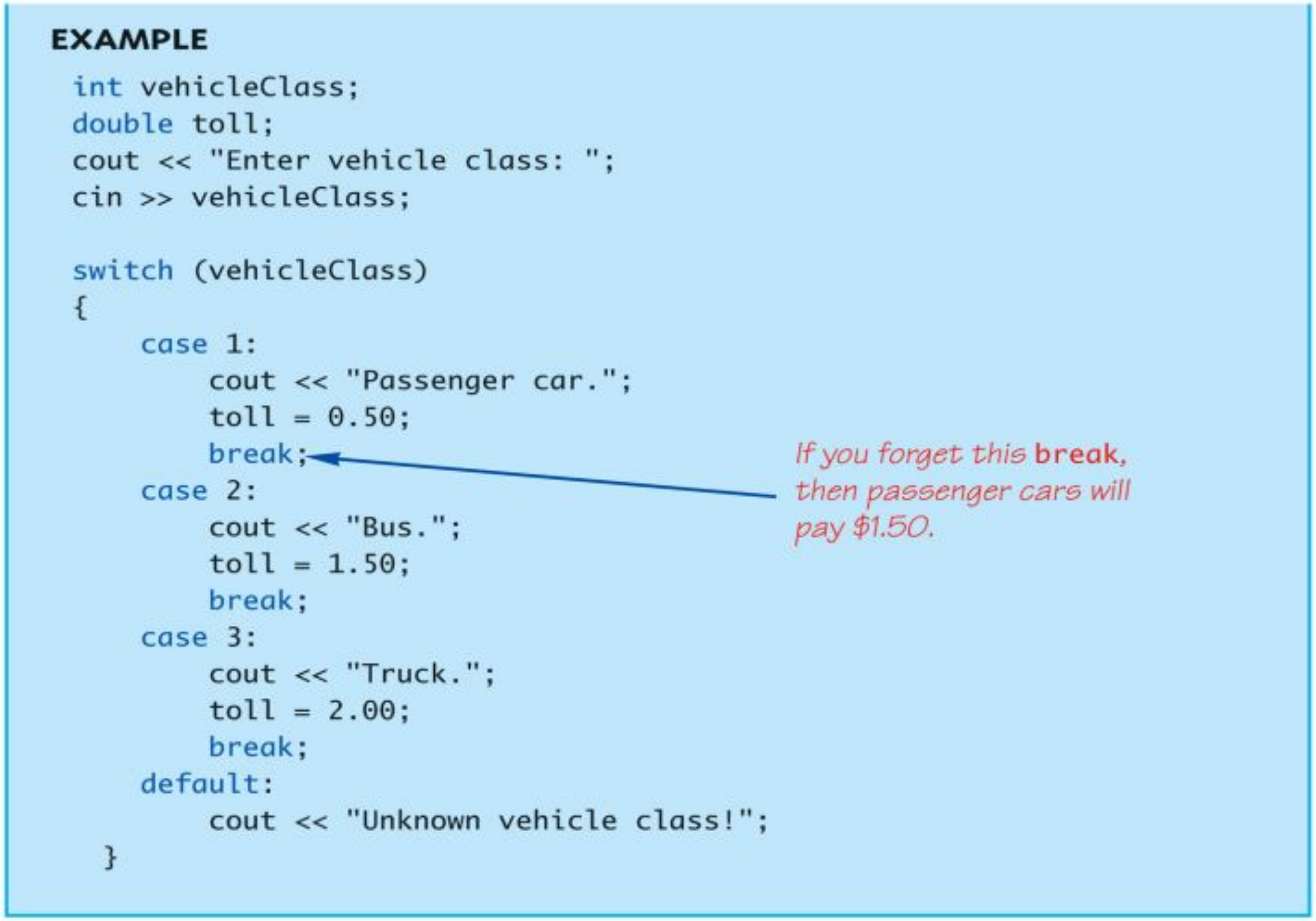
SYNTAX

```
switch (Controlling_Expression)
{
      case Constant_1:
            Statement_Sequence_1
            break;
      case Constant_2:
            Statement_Sequence_2
            break;
                                    .
                                    .
                                    .
      case Constant_n:
              Statement_Sequence_n
              break;
      default:
              Default_Statement_Sequence
}
```

You need not place a **break** *statement in each case. If you omit a* **break**, *that case continues until a* **break** *(or the end of the* **switch** *statement) is reached.*

Switch Statement (3/3)

- Example :



EXAMPLE

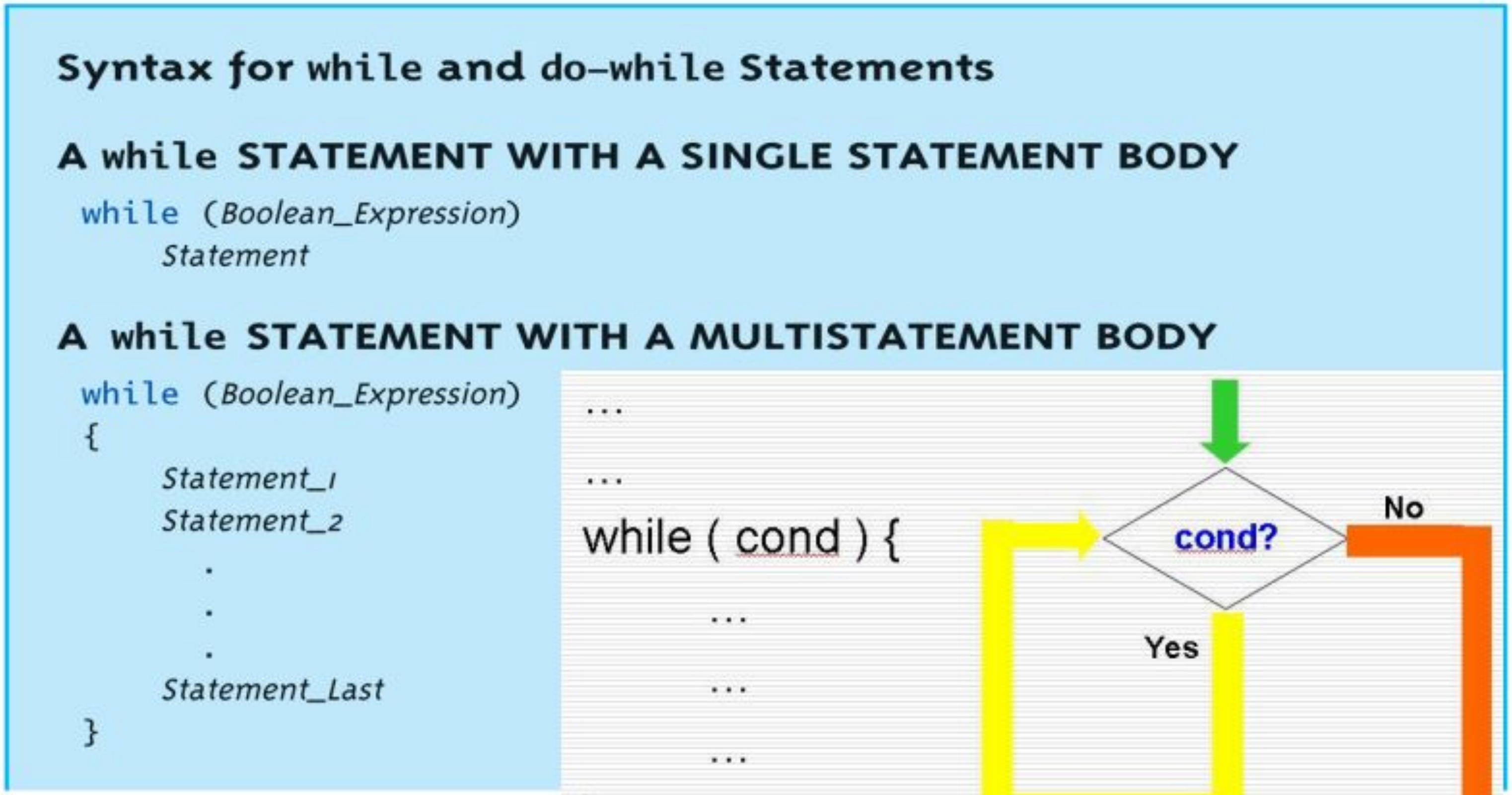
```
int vehicleClass;
double toll;
cout << "Enter vehicle class: ";
cin >> vehicleClass;

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!";
  }
```

If you forget this **break**, *then passenger cars will pay $1.50.*

Loops

- 3 Types of loops in C++
  - while
  - do-while
    - always enters the loop body at least once
  - for
    - appropriate for “counting” loops

while Loop Syntax



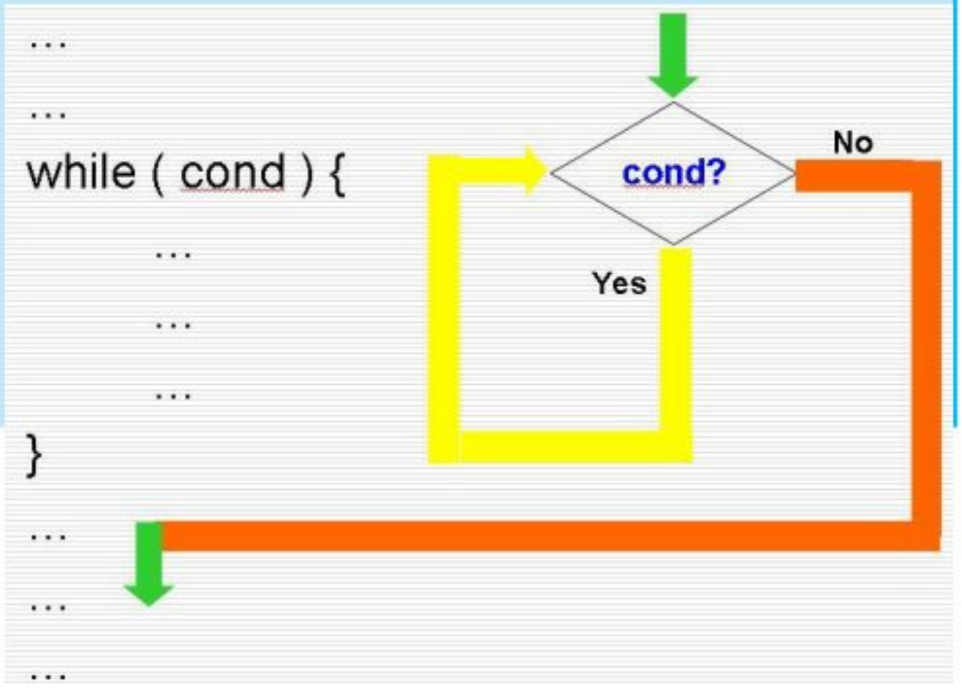
Syntax for while and do-while Statements

A while STATEMENT WITH A SINGLE STATEMENT BODY

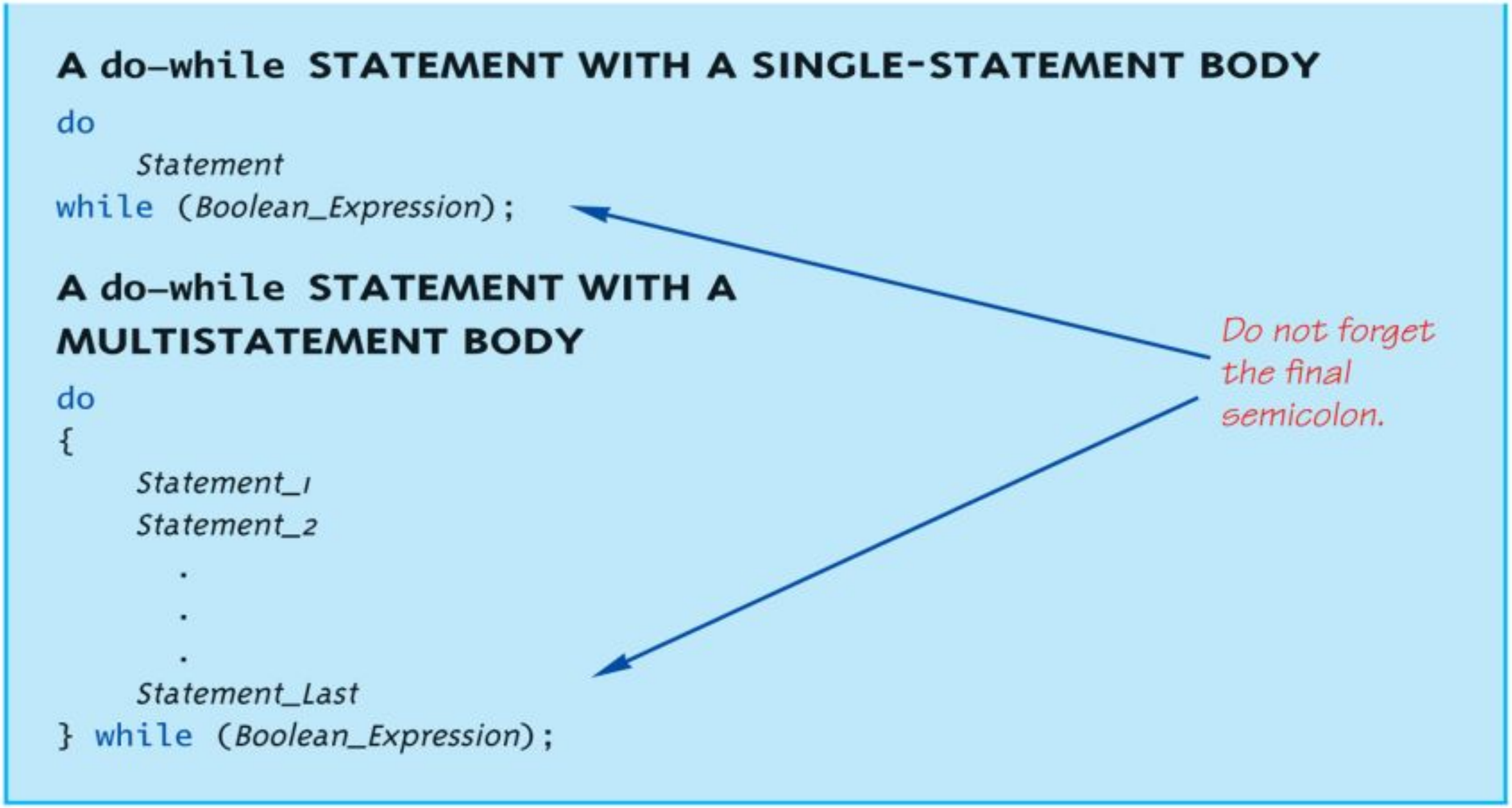
```
while (Boolean_Expression)
    Statement
```

A while STATEMENT WITH A MULTISTATEMENT BODY

```
while (Boolean_Expression)
{
    Statement_1
    Statement_2
       .
       .
       .
    Statement_Last
}
```



do-while Loop Syntax



A do-while STATEMENT WITH A SINGLE-STATEMENT BODY

```
do
    Statement
while (Boolean_Expression);
```

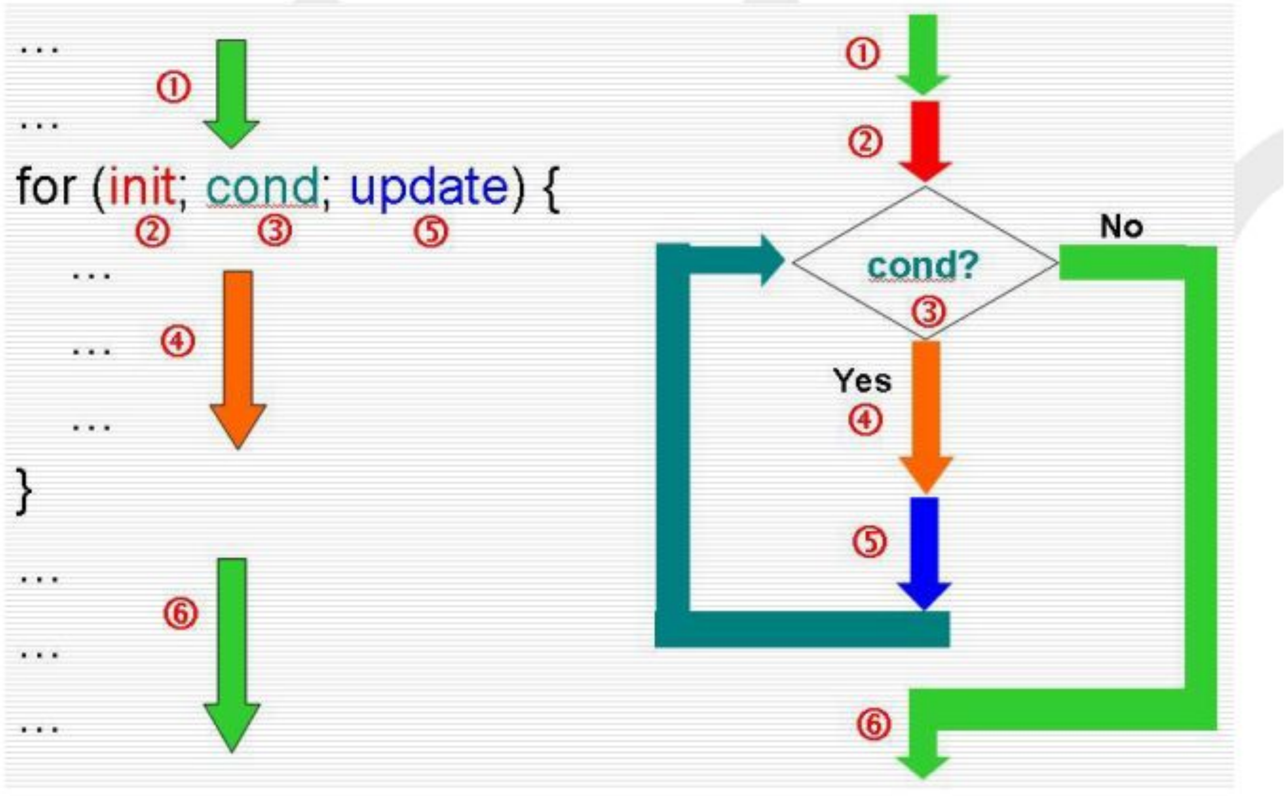
A do-while STATEMENT WITH A MULTISTATEMENT BODY

```
do
{
    Statement_1
    Statement_2
      .
      .
      .
    Statement_Last
} while (Boolean_Expression);
```

Do not forget the final semicolon.

for Loop Syntax

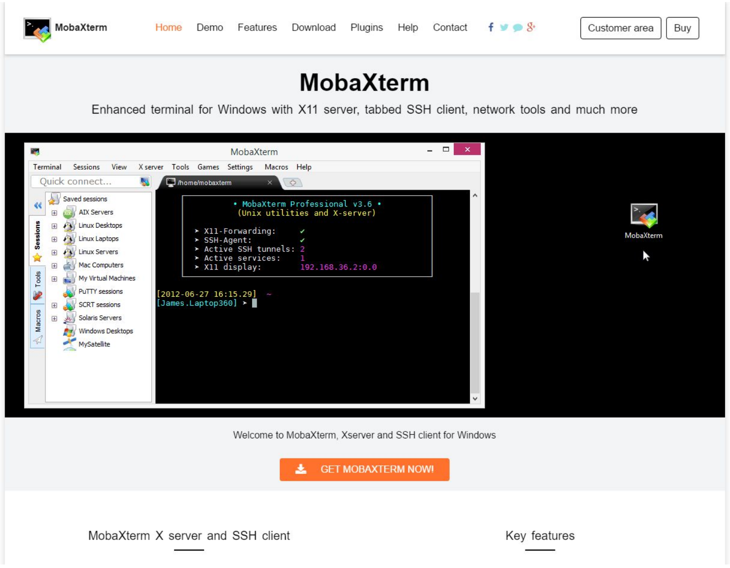
Syntax:
for (Init_Action; Bool_Cond; Update_Action)
Body_Statement



SPECIAL TOPIC - LINUX

Remote Access - MobaXTerm

- http://mobaxterm.mobatek.net
  - ->Download
  - ->Home Edition
  - ->(Portable/Installer Edition)



MobaXterm

Enhanced terminal for Windows with X11 server, tabbed SSH client, network tools and much more

```
        • MobaXterm Professional v3.6 •
         (Unix utilities and X-server)

  ➤ X11-Forwarding:     ✓
  ➤ SSH-Agent:          ✓
  ➤ Active SSH tunnels: 2
  ➤ Active services:    1
  ➤ X11 display:        192.168.36.2:0.0

[2012-06-27 16:15.29]  ~
[James.Laptop360] ➤
```

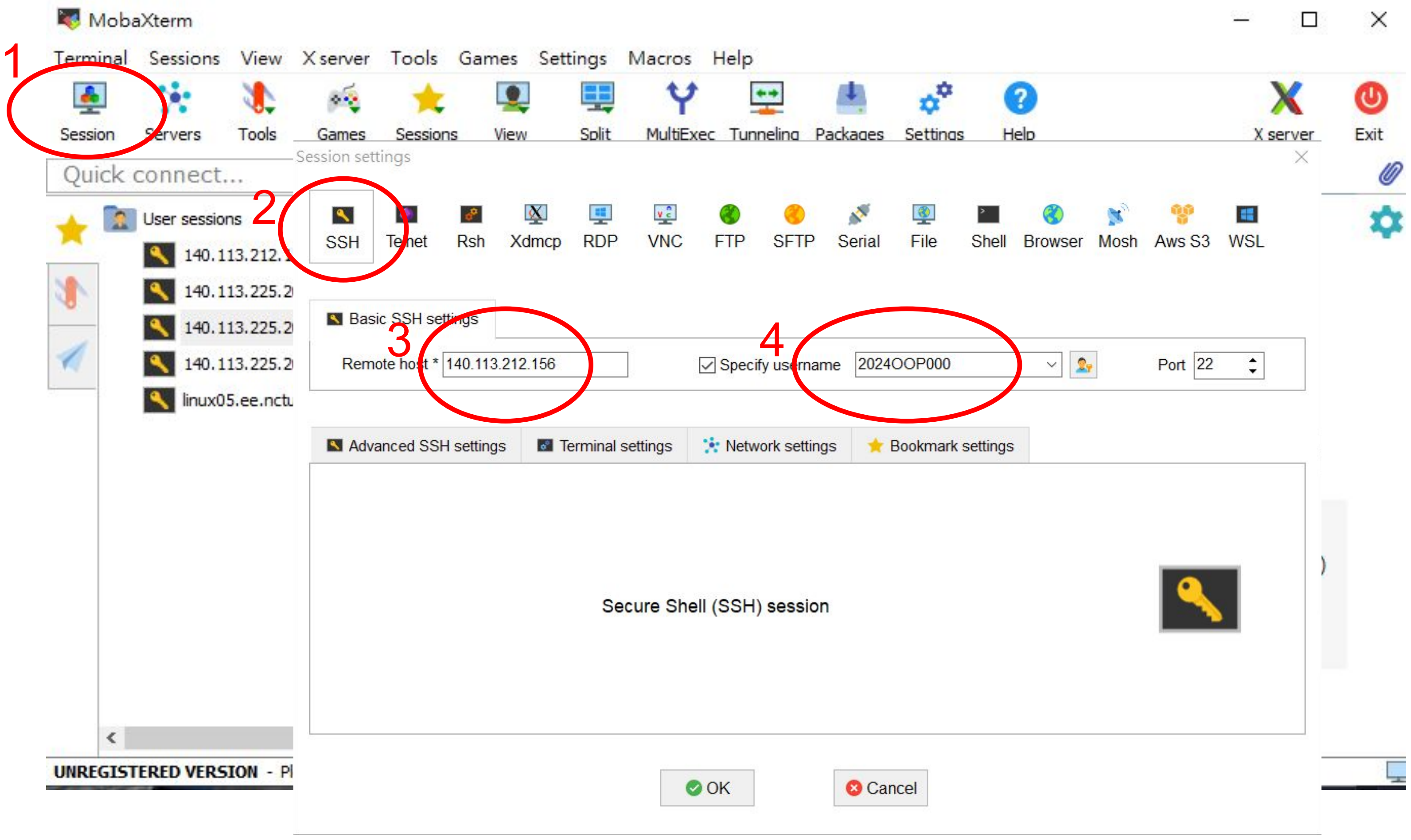
Welcome to MobaXterm, Xserver and SSH client for Windows

MobaXterm X server and SSH client

Key features

Workstation IP

- 140.113.212.156


MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
User sessions
SSH Telnet Rsh Xdmcp RDP VNC FTP SFTP Serial File Shell Browser Mosh Aws S3 WSL
Advanced SSH settings
Terminal settings
Network settings
Bookmark settings
OK
Cancel
UNREGISTERED VERSION
1
2
3
4

Session settings

Basic SSH settings

Remote host * 140.113.212.156

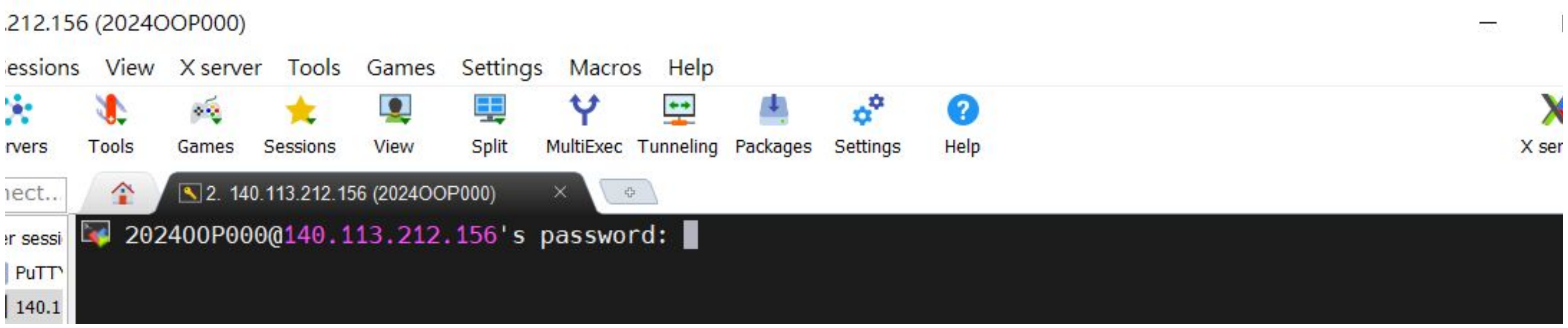
☑ Specify username 2024OOP000

Port 22

Secure Shell (SSH) session

Login

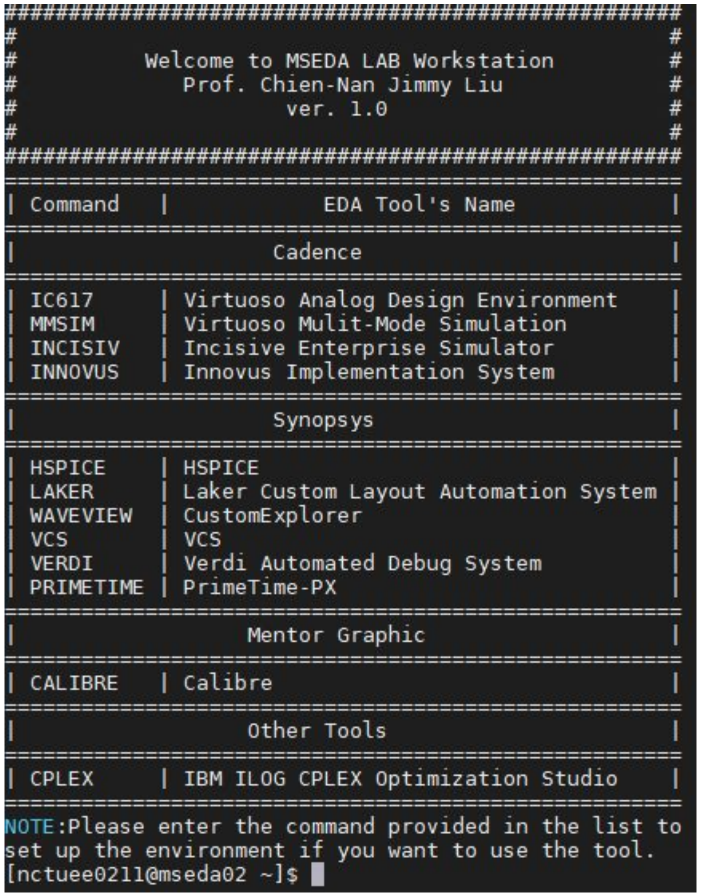
- Type your password (default: OOP2024)
- Remember to change your password by using this command : passwd


.212.156 (2024OOP000)
essions View X server Tools Games Settings Macros Help
rvers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X ser
2. 140.113.212.156 (2024OOP000)

```
2024OOP000@140.113.212.156's password: 
```

Command Line Interface

- Username
- Machine Name
- Path
  - [~] = Home directory



```
####################################################################
#                                                                  #
#               Welcome to MSEDA LAB Workstation                   #
#                  Prof. Chien-Nan Jimmy Liu                       #
#                          ver. 1.0                                #
#                                                                  #
####################################################################
====================================================================
| Command     |             EDA Tool's Name                        |
====================================================================
|                          Cadence                                 |
====================================================================
| IC617       | Virtuoso Analog Design Environment                 |
| MMSIM       | Virtuoso Mulit-Mode Simulation                     |
| INCISIV     | Incisive Enterprise Simulator                      |
| INNOVUS     | Innovus Implementation System                      |
====================================================================
|                          Synopsys                                |
====================================================================
| HSPICE      | HSPICE                                             |
| LAKER       | Laker Custom Layout Automation System              |
| WAVEVIEW    | CustomExplorer                                     |
| VCS         | VCS                                                |
| VERDI       | Verdi Automated Debug System                       |
| PRIMETIME   | PrimeTime-PX                                       |
====================================================================
|                        Mentor Graphic                            |
====================================================================
| CALIBRE     | Calibre                                            |
====================================================================
|                        Other Tools                               |
====================================================================
| CPLEX       | IBM ILOG CPLEX Optimization Studio                 |
====================================================================
NOTE:Please enter the command provided in the list to
set up the environment if you want to use the tool.
[nctuee0211@mseda02 ~]$
```

Linux Basics – commands

- ls : see files in the current directory
  - ls –n : see files in detail
- cd <directory_name> : enter the directory
  - cd .. : enter the upper directory
- mkdir <new_directory_name> : create new directory
- rm <file_name> : remove file
  - rm –r <directory_name> : remove directory
- cp <file_name> <directory_name/file_name>: copy file to the directory
- mv <file_name> <directory_name/file_name>: move file to the directory

Editor - VIM (II)

- You can now enter commands to save your file (Do not forget to press Enter)
  - :w   -> save your file
  - :wq -> save and quit
  - :q -> quit
  - :q! -> quit without saving

Editor – VIM (Advanced)

- In command mode:
  - /<string> -> find string
  - dd -> delete current line
  - yy -> copy current line
  - pp -> paste copied lines
  - u -> undo
  - Ctrl + r -> redo
  - Ctrl + v -> Block Select Mode (3rd mode, Esc to quit, you can d, y, p your selected block)

The other option: Notepad++

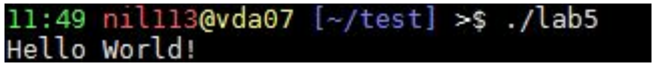
- https://notepad-plus-plus.org/downloads/
- Choose the edition you desired
- Plugins->Plugin Manager->NppFTP->Download
- NppFTP->Show NppFTP Window
- Settings->Profile Settings
- Hostname:140.113.212.156
- Connection type: SFTP

Compiler – g++

- Usage: g++ <option> <file_name>
- Options:
  - -o <file_name> : Name the binary
  - -std=<standard> : Choose the language standard, for example -std=c++11
  - -c : Create object file instead of binary
  - -g : Debug Mode
  - Ex : g++ main.cpp -o lab1
  - Ex : g++ -std=c++11 main.cpp -o lab1

Linux Basics - Execute

- Run the binary code in your current machine
- ./<executable binary> <parameters>
  - Example: ./lab1

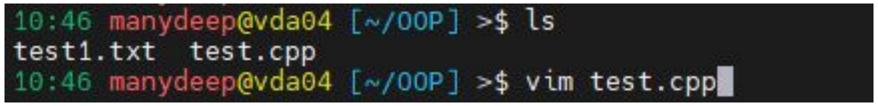


```
11:49 nil113@vda07 [~/test] >$ ./lab5
Hello World!
```

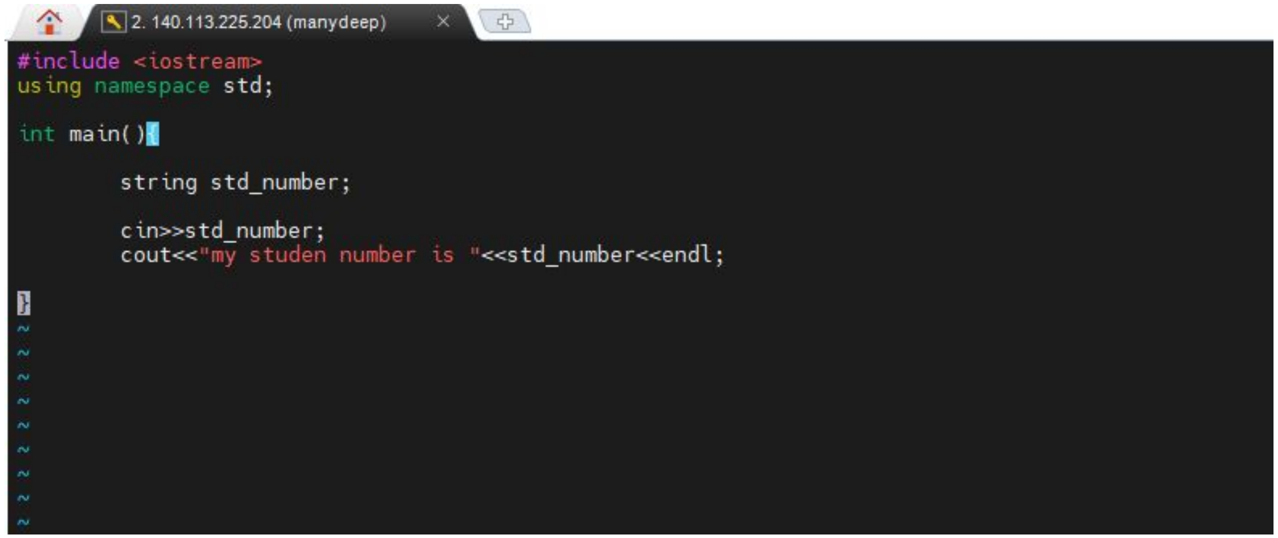
- Ctrl+C : Terminate the program (infinite loop)

Example

- Edit a file named "test.cpp"
  - vim test.cpp
  - Remember to use ":w" to save your file



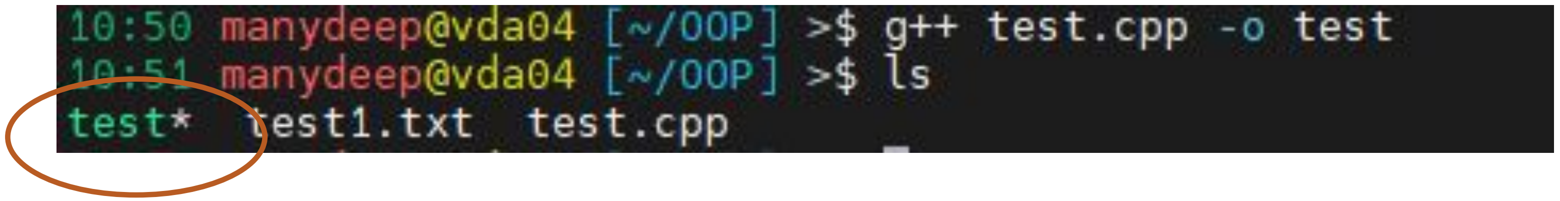
```
10:46 manydeep@vda04 [~/OOP] >$ ls
test1.txt  test.cpp
10:46 manydeep@vda04 [~/OOP] >$ vim test.cpp
```



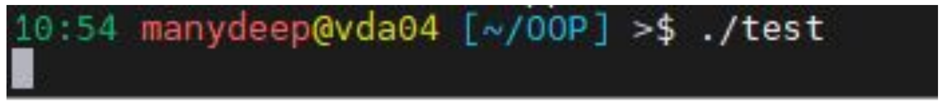
```
#include <iostream>
using namespace std;

int main(){

        string std_number;

        cin>>std_number;
        cout<<"my studen number is "<<std_number<<endl;

}
~
~
~
~
~
~
~
~
~
```

Example

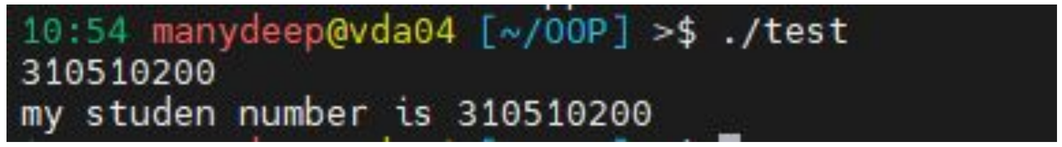
- Compile your file
  - g++ test.cpp –o test



```
10:50 manydeep@vda04 [~/OOP] >$ g++ test.cpp -o test
10:51 manydeep@vda04 [~/OOP] >$ ls
test*  test1.txt  test.cpp
```



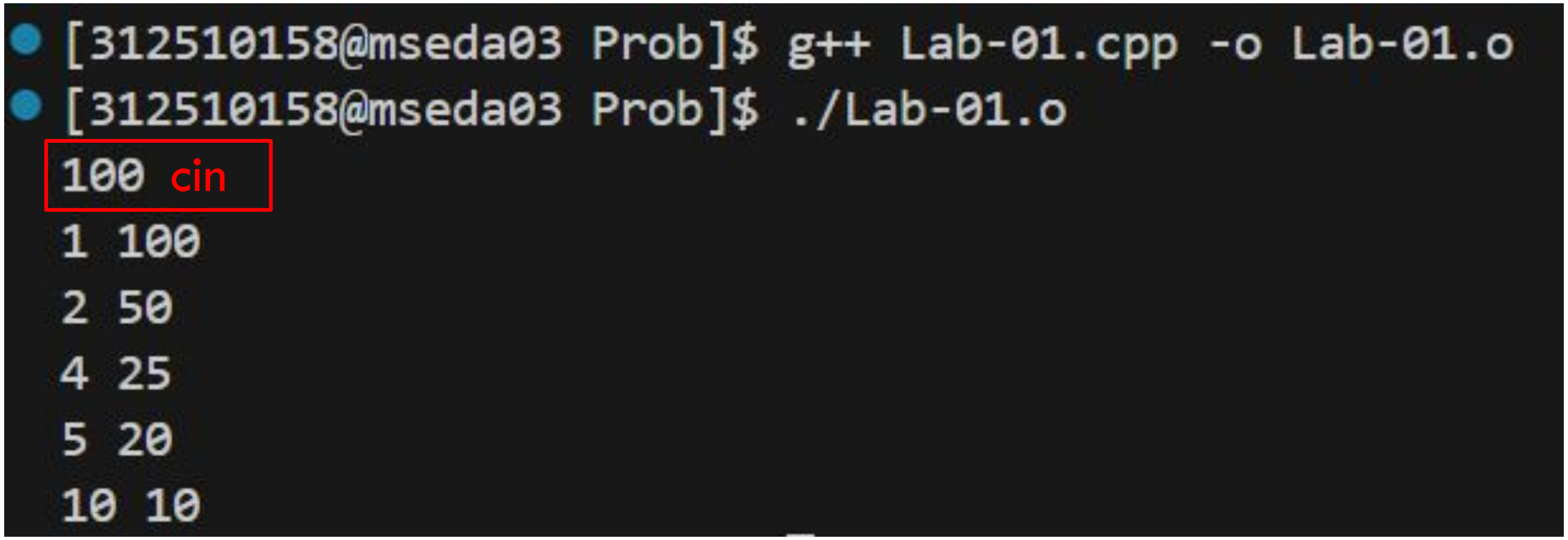
```
10:54 manydeep@vda04 [~/OOP] >$ ./test
```



```
10:54 manydeep@vda04 [~/OOP] >$ ./test
310510200
my studen number is 310510200
```

Lab Exercise (1/2)

- Input an integer N, print out all the combinations of multiplication in the format of **"A B"**, where A x B equals to N.
- N will not be larger than 10 digits
- A should be not greater than B (A ≦ B)

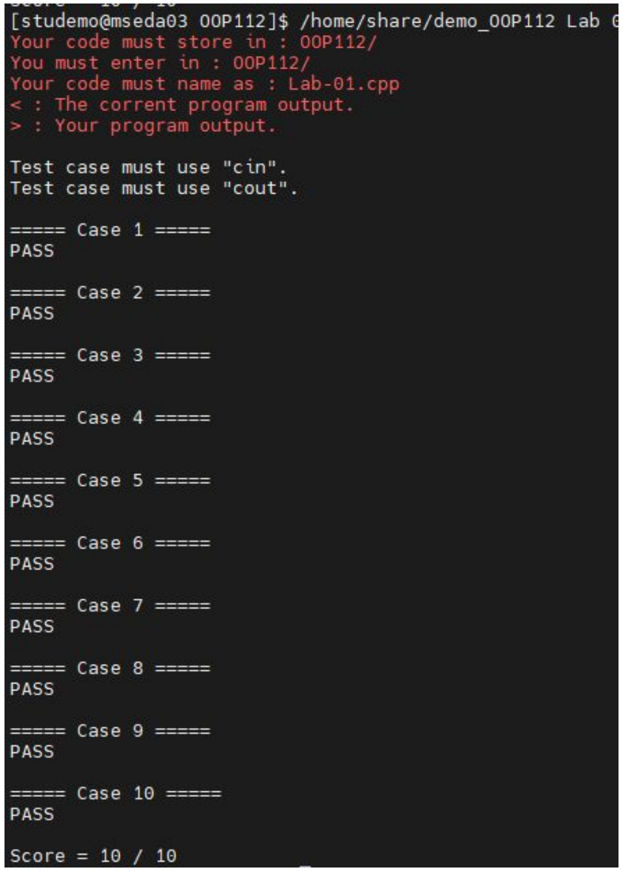


```
[312510158@mseda03 Prob]$ g++ Lab-01.cpp -o Lab-01.o
[312510158@mseda03 Prob]$ ./Lab-01.o
100  cin
1 100
2 50
4 25
5 20
10 10
```

Lab Exercise (2/2)

1. Create a directory “OOP112” (mkdir OOP112)
2. Change your working directory to “OOP112” (cd OOP112)
3. Create a cpp file “Lab-01.cpp” (touch Lab-01.cpp)
4. Write your code in Lab-01.cpp

- During demo, please type this command:
- /home/share/demo_OOP112 Lab 01
- TA will check your code with this command



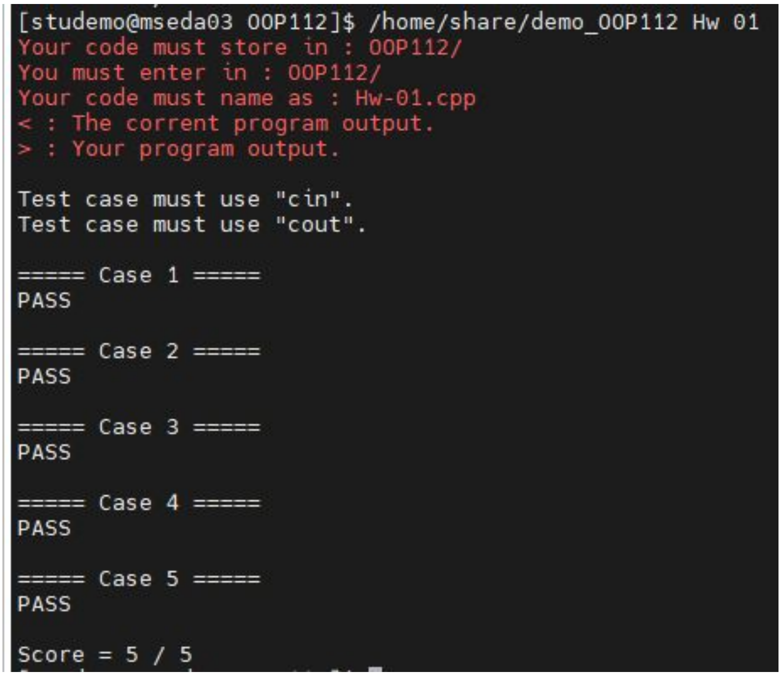
```
[studemo@mseda03 OOP112]$ /home/share/demo_OOP112 Lab 0
Your code must store in : OOP112/
You must enter in : OOP112/
Your code must name as : Lab-01.cpp
< : The corrent program output.
> : Your program output.

Test case must use "cin".
Test case must use "cout".

===== Case 1 =====
PASS

===== Case 2 =====
PASS

===== Case 3 =====
PASS

===== Case 4 =====
PASS

===== Case 5 =====
PASS

===== Case 6 =====
PASS

===== Case 7 =====
PASS

===== Case 8 =====
PASS

===== Case 9 =====
PASS

===== Case 10 =====
PASS

Score = 10 / 10
```

Submission

- Ask TAs for demo
- Try your best to debug your code by yourself
- Upload all your cpp to new E3
- Naming rule : Lab-01.cpp

HW1 (Only This Lab)

- In this semester, TA will also create our own OJ (Online Judge) to check your Homework.
- Every student needs to **complete HW1 in this lab**
- During demo, please type this command:
- /home/share/demo_OOP112 Hw 01



```
[studemo@mseda03 OOP112]$ /home/share/demo_OOP112 Hw 01
Your code must store in : OOP112/
You must enter in : OOP112/
Your code must name as : Hw-01.cpp
< : The corrent program output.
> : Your program output.

Test case must use "cin".
Test case must use "cout".

===== Case 1 =====
PASS

===== Case 2 =====
PASS

===== Case 3 =====
PASS

===== Case 4 =====
PASS

===== Case 5 =====
PASS

Score = 5 / 5
```