CS2100 COMPUTER ORGANISATION

http://www.comp.nus.edu.sg/~cs2100/

Lecture #2c

Overview of C Programming

NUS National University of Singapore | School of Computing

Questions?

Ask at https://app.sli.do/event/qVCWNryB45Bnh6p2HRfnFG

OR

Scan and ask your questions here!
(May be obscured in some slides)

6. Selection Structures (1/2)

- C provides two control structures that allow you to select a group of statements to be executed or skipped when certain conditions are met.

if … else …

```
if (condition) {
   /* Execute these statements if TRUE */
}
```

```
if (condition) {
   /* Execute these statements if TRUE  */
}
else {
   /* Execute these statements if FALSE */
}
```

```
if condition:
   # Statement
```

```
if condition:
   # Statement
elif condition:
   # Statement
else:
   # Statement
```

6. Selection Structures (2/2)

switch

Python
No counterpart

```
/* variable or expression must be of discrete type */
switch ( <variable or expression> ) {
   case value1:
       Code to execute if <variable or expr> == value1
       break;

   case value2:
       Code to execute if <variable or expr> == value2
       break;
   ...

   default:
       Code to execute if <variable or expr> does not
       equal to the value of any of the cases above
       break;
}
```

6.1 Condition and Relational Operators

- A condition is an expression evaluated to ***`true`*** or ***`false`***.
- It is composed of expressions combined with relational operators.
  - Examples: `(a <= 10)`, `(count > max)`, `(value != -9)`

Relational Operator	Interpretation
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

Python

Allows
`1 <= x <= 5`

6.2 Truth Values

- Boolean values: true or false.
- There is no Boolean type in ANSI C. Instead, we use **integers**:
  - 0 to represent false
  - Any other value to represent true (1 is used as the representative value for true in output)
- Example:

Python

NOTE: *only integers!*
In Python and JavaScript you have truthy and falsy values, but not in C

TruthValues.c

```c
int a = (2 > 3);
int b = (3 > 2);

printf("a = %d; b = %d\n", a, b);
```

```
a = 0; b = 1
```

6.3 Logical Operators

- Complex condition: combining two or more Boolean expressions.
- Examples:
  - If temperature is greater than 40C or blood pressure is greater than 200, go to A&E immediately.
  - If all the three subject scores (English, Maths and Science) are greater than 85 and mother tongue score is at least 80, recommend taking Higher Mother Tongue.
- Logical operators are needed: && (and), || (or), ! (not).

A	B	A && B	A || B	!A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Python

```
A || B → A or B
A && B → A and B
!A → not A
```

6.4 Evaluation of Boolean Expressions (1/2)

- The evaluation of a Boolean expression is done according to the precedence and associativity of the operators.

Operator Type	Operator	Associativity
Primary expression operators	( ) [ ] . -> expr++ expr--	Left to Right
Unary operators	* & + - ! ~ ++expr --expr (typecast) sizeof	Right to Left
Binary operators	* / %	Left to Right
	+ -	
	< > <= >=	
	== !=	
	&&	
	||	
Ternary operator	?:	Right to Left
Assignment operators	= += -= *= /= %=	Right to Left

Python

```
cond ? expr1 : expr2 →
expr1 if cond else cond2
```

6.4 Evaluation of Boolean Expressions (2/2)

- What is the value of x?

```
int x, y, z,
    a = 4, b = -2, c = 0;
x = (a > b || b > c && a == b);
```

x is true (1)

gcc issues warning (why?)

- Always good to add parentheses for readability.

```
y = ((a > b || b > c) && a == b);
```

y is false (0)

- What is the value of z?

```
z = ((a > b) && !(b > c));
```

z is true (1)

Try out EvalBoolean.c

6.5 Short-Circuit Evaluation

- Does the following code give an error if variable a is zero?

```
if ((a != 0) && (b/a > 3)) {
    printf(. . .);
}
```

- Short-circuit evaluation
  - expr1 || expr2: If expr1 is true, skip evaluating expr2 and return true immediately, as the result will always be true.
  - expr1 && expr2: If expr1 is false, skip evaluating expr2 and return false immediately, as the result will always be false.

7. Repetition Structures (1/2)

- C provides three control structures that allow you to select a group of statements to be executed repeatedly.

```
while ( condition )
{
   // loop body
}
```

```
do
{
   // loop body
}  while ( condition );
```

```
for ( initialization; condition; update )
{
    // loop body
}
```

Initialization: initialize the **loop variable**

Condition: repeat loop while the condition on **loop variable** is true

Update: change value of **loop variable**

7. Repetition Structures (2/2)

- Example: Summing from 1 through 10.

Sum1To10_While.c

```
int sum = 0, i = 1;
while (i <= 10) {
   sum = sum + i;
   i++;
}
```

Sum1To10_DoWhile.c

```
int sum = 0, i = 1;
do {
   sum = sum + i;
   i++;
}
while (i <= 10);
```

Sum1To10_For.c

```
int sum, i;
for (sum = 0, i = 1; i <= 10; i++) {
   sum = sum + i;
}
```

7.1 Using 'break' in a loop (1/2)

BreakInLoop.c

```
// without 'break'
printf ("Without 'break':\n");
for (i=1; i<=5; i++) {
   printf("%d\n", i);
   printf("Ya\n");
}
```

```
// with 'break'
printf ("With 'break':\n");
for (i=1; i<=5; i++) {
   printf("%d\n", i);
   if (i==3)
      break;
   printf("Ya\n");
}
```

```
Without 'break':
1
Ya
2
Ya
3
Ya
4
Ya
5
Ya
```

```
With 'break':
1
Ya
2
Ya
3
```

7.1 Using ‘break’ in a loop (2/2)

BreakInLoop.c

```c
// with 'break' in a nested loop
printf("With 'break' in a nested loop:\n");
for (i=1; i<=3; i++) {
   for (j=1; j<=5; j++) {
      printf("%d, %d\n", i, j);
      if (j==3)
         break;
      printf("Ya\n");
   }
}
```

```
With 'break' in …
1, 1
Ya
1, 2
Ya
1, 3
2, 1
Ya
2, 2
Ya
2, 3
3, 1
Ya
3, 2
Ya
3, 3
```

- In a nested loop, `break` only breaks out of the inner-most loop that contains the `break` statement.

7.2 Using ‘continue’ in a loop (1/2)

ContinueInLoop.c

```
// without 'continue'
printf ("Without 'continue':\n");
for (i=1; i<=5; i++) {
   printf("%d\n", i);
   printf("Ya\n");
}
```

```
Without 'continue':
1
Ya
2
Ya
3
Ya
4
Ya
5
Ya
```

```
// with 'continue'
printf ("With 'continue':\n");
for (i=1; i<=5; i++) {
   printf("%d\n", i);
   if (i==3)
      continue;
   printf("Ya\n");
}
```

```
With 'continue':
1
Ya
2
Ya
3
4
Ya
5
Ya
```

7.2 Using ‘continue’ in a loop (2/2)

ContinueInLoop.c

```c
// with 'continue' in a nested loop
printf("With 'continue' in a nested loop:\n");
for (i=1; i<=3; i++) {
   for (j=1; j<=5; j++) {
      printf("%d, %d\n", i, j);
      if (j==3)
         continue;
      printf("Ya\n");
   }
}
```

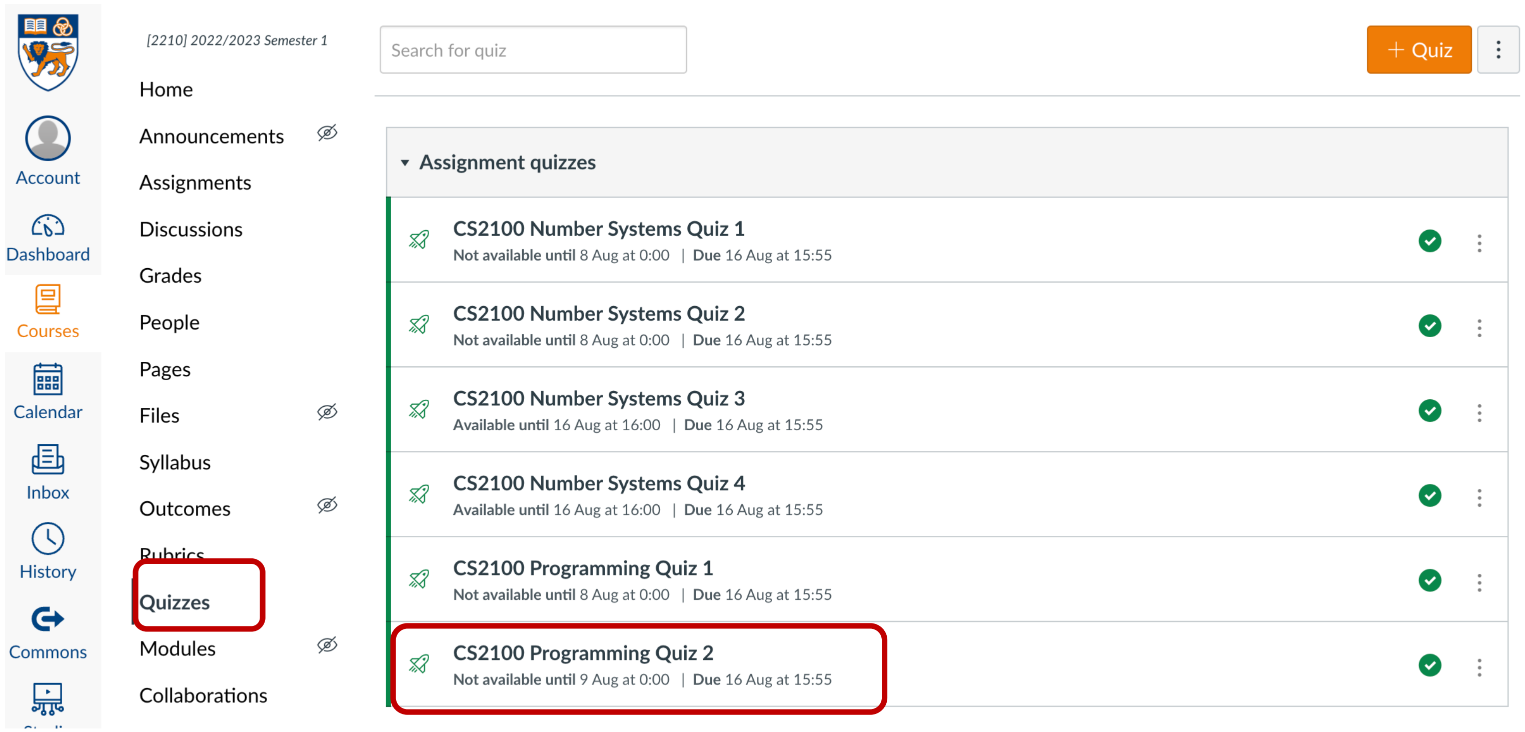
```
With ...
1, 1
Ya
1, 2
Ya
1, 3
1, 4
Ya
1, 5
Ya
2, 1
Ya
2, 2
Ya
2, 3
2, 4
Ya
2, 5
Ya
```

```
3, 1
Ya
3, 2
Ya
3, 3
3, 4
Ya
3, 5
Ya
```

- In a nested loop, `continue` only skips to the next iteration of the inner-most loop that contains the `continue` statement.

Quiz

- Please complete the “CS2100 C Programming Quiz 2” in Canvas.
  - Access via the “Quizzes” tool in the left toolbar and select the quiz on the right side of the screen.



CS2100 Number Systems Quiz 1
Not available until 8 Aug at 0:00 | Due 16 Aug at 15:55

CS2100 Number Systems Quiz 2
Not available until 8 Aug at 0:00 | Due 16 Aug at 15:55

CS2100 Number Systems Quiz 3
Available until 16 Aug at 16:00 | Due 16 Aug at 15:55

CS2100 Number Systems Quiz 4
Available until 16 Aug at 16:00 | Due 16 Aug at 15:55

CS2100 Programming Quiz 1
Not available until 8 Aug at 0:00 | Due 16 Aug at 15:55

CS2100 Programming Quiz 2
Not available until 9 Aug at 0:00 | Due 16 Aug at 15:55

End of File