Data Structures in C

Prof. Georg Feil

Loops

Summer 2018

Acknowledgement

- These lecture slides are based on slides and other material by Professor Magdin Stoica
- Additional sources are cited separately

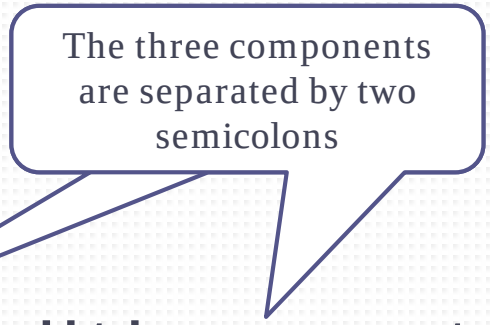
Reading Assignments

- C for Programmers (supplementary textbook)
  - Chapter 4: Control Statements Part II

Loop Types

- Definite (Counted) Loops
  - Repeat the statements in the loop block an exact, definite, number of times (e.g. 10, 20, 1000)
  - Useful when you know (or your program has calculated) how many times something needs to be repeated
- Indefinite (Conditional) Loops
  - Repeat the statements inside the loop block for as long as a condition is true
  - If the loop condition never changes from true to false the loop will execute forever (unless you use 'break') – infinite loop!
  - Useful when you don't know how many times something needs to be repeated, e.g. read input until end of file

The for Loop (pseudocode)

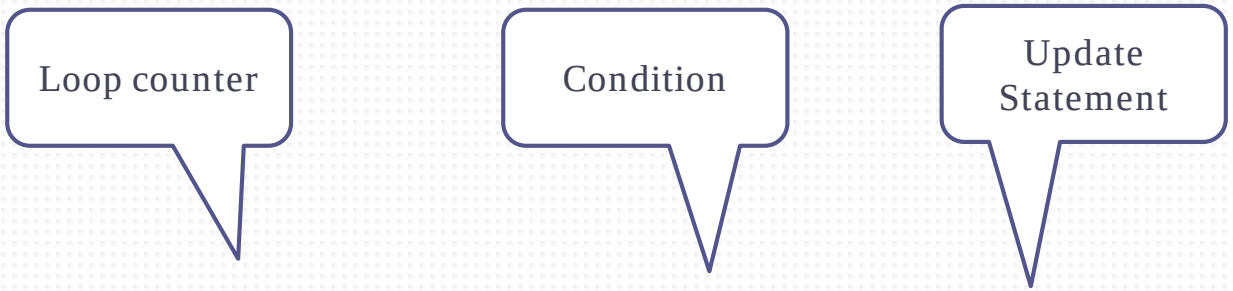



```
for (<initialization> ; <condition>; <next step>)
{
        <statement 1>;
        <statement 2>;
        <statement 3>;
        …
}
```

Loop Block

These statements will repeat for as long as the condition is true

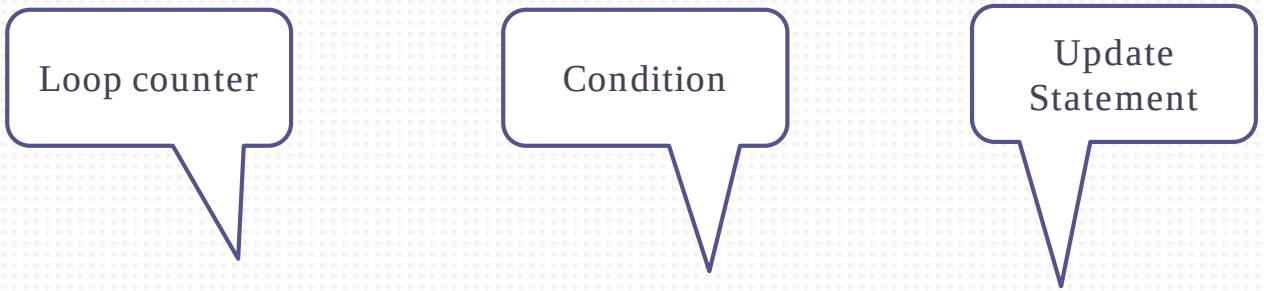
Example: for Loop




```
for (int count = 0; count < 5; count++) {
    printf("Hello loop %d\n", count);
}
```

- How many times does this loop execute?
- What is the last ‘count’ value printed?

Example: for Loop Backward




```
for (int count = 4; count >= 0; count--) {
    printf("Hello loop %d\n", count);
}
```

Naming your counter variable

- Use a representative name for your loop index (counter) variable
- Good names
  - lineNum: line number that goes from zero to 10 for every page
  - iPlayer: i is short for “index” so iPlayer is short for “index of player”
  - empNum: employee number
  - questionCounter
- Not so good names
  - i
  - j
  - k
  - l
  - These are good prefixes but do not identify what the counter is counting.

The while Loop (pseudocode)

The loop condition. Loop will executes as long as the expression is true.

```
while (<boolean expression>)
{
        <statement 1>;
        <statement 2>;
        <statement 3>;
        …
}
```

Loop Block

These statements will repeat for as long as the boolean expression is true

Example: Definite while Loop

```
int lineNo = 0;
while (lineNo < 5) {
      printf("Hello loop %d\n", lineNo);
      lineNo++;
}
```

The condition is evaluated at the top of the loop (before the block runs)

Example: Indefinite while Loop

```
int input = -1;
while (input != 0) {
    printf("\nPlease enter a number, 0 to quit: ");
    scanf("%d", &input);
    printf("\nYou entered %d\n", input);
}
```

Common properties of for and while loops

- Loop repeats as long as the loop condition is true
  - For loop condition usually tests the counter value
- The condition is evaluated BEFORE the loop block is executed
- The loop may run zero times if the condition is false at the start

Exercise 1

- Write a program to print two (or more) rectangles with different dimensions (width x height), made of different characters, e.g.
  - 14 x 7 rectangle made of Os
  - 8 x 2 rectangle made of Xs
- Use a separate function with appropriate parameters to avoid code duplication
  - Call it twice from the main function
- "Bonus": Allow the dimensions and character to be given by the user in an interactive manner

```
00000000000000
00000000000000
00000000000000
00000000000000
00000000000000
00000000000000
00000000000000

XXXXXXXX
XXXXXXXX
```

‘continue’ and ‘break’

Controlling loops: continue and break

- Do *partial* iterations using the continue keyword
  - It's possible to skip particular iterations
  - It's possible to partially execute an iteration
  - continue is used in conjunction with 'if' to decide that certain iterations could be entirely or partially skipped
- End a loop early with the break keyword
  - break ends the loop regardless of the result of the loop condition
- These keywords can be used will all types of loops

Example: continue

```
for (int num = 0; num < 100; num++) {
      if (num % 2 == 0) {
          // Skip the print statement for even numbers
          continue;
      }
       printf("%d is an odd number\n", num);
}
```

Example: break

```
int factor = 1;
while (true) {
      factor *= 2;
      if (factor > 1000) {
            break;     // Quit the loop
      }
     printf("%d\n", factor);
}
```

The do-while Loop

- Conditional loop that repeats the loop block while the boolean expression it defines is true

- The condition is evaluated AFTER the loop block is executed
  - The statements inside the loop block will always execute at least once

do-while Loop Example: Input Until 0

```
int num;
do {
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("\nYou entered %d\n", num);
} while (num != 0);
```

Need to declare this variable before the loop block

The only control structure that must end with a semicolon

Commenting a Loop

- Always comment loop statements
  - Regardless of type
- Your comment should explain why you need the loop
  - Example: “Go through all employees to calculate the total number of hours worked”