Control Structures in C

Hyung-Sin Kim

Computing for Data Science

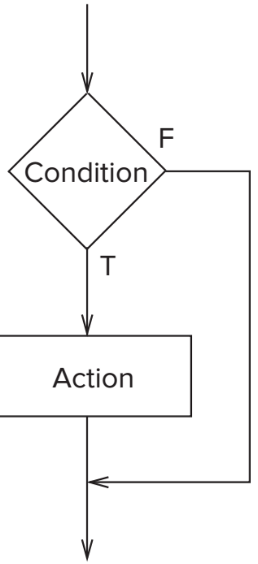
Graduate School of Data Science, Seoul National University

Conditional Constructs

If Statement

- For a block of one statement
    - **if** (<condition>)
    -     <one statement block>

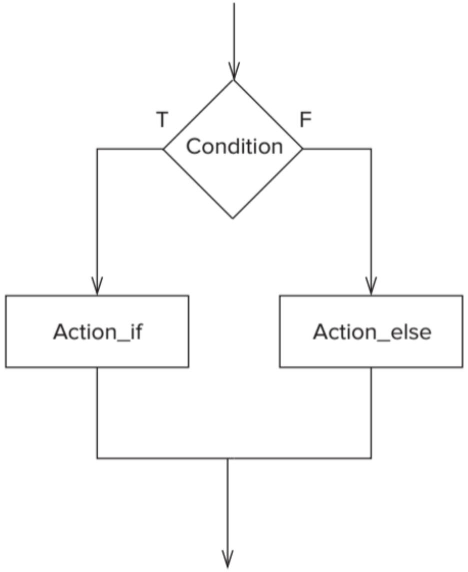
- For a block of multiple statements
    - **if** (<condition>) {
    -   <statement 1>
    -   <statement 2>
    -   ….
    - }



If-Else Statement

- For a block of one statement
  - if (<condition>)
  -     <one statement block>
  - else
  -     <one statement block>

- For a block of multiple statements
  - **if** (<condition>) {
  -     <multi-statement block>
  - }
  - **else** {
  -     <multi-statement block>
  - }



If-Else Statement

- Example

```
#include <stdio.h>

int main(void){
    int month;
    printf(“Enter the number of the month: “);
    scanf(“%d”, &month);

    if (month == 4 || mont == 6 || month == 9 || month == 11)
        printf(“The month has 30 days\n”);
    else if (month == 1 || month == 3 || month == 5 ||
                month ==7 ||month == 8 || month == 10 || month == 12)
        printf(“The month has 31 days\n”);
    else
        printf(“Don’t know that month\n”);
}
```

If-Else Statement

- An **else** is associated with the closest unassociated **if**

```
if (x != 10)
    if (y > 3)
        z = z / 2;
    else
        z * 2;
```

- To not be confused, it is better to **clarify** the associativity by using **parentheses**

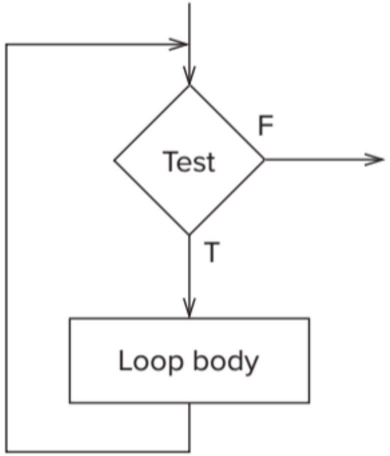
```
if (x != 10) {
    if (y > 3)
        z = z / 2;
}
else {
    z = z * 2;
}
```

Iteration Constructs

While Statement

- **while** (**<test>**) {
-     **<loop body>**
- }

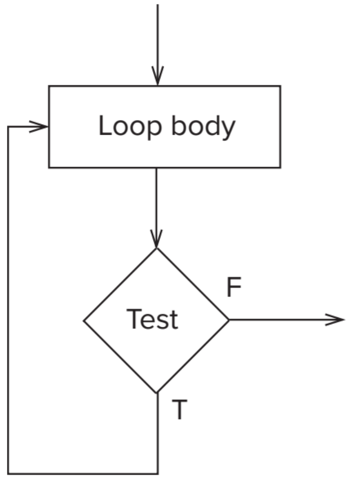
- Example
    - #include <stdio.h>
    - int main(void) {
    -     int x = 0;
    -     **while** (x < 10) {
    -         printf(“%d “, x);
    -         x += 1;
    -     }
    - }



Do-While Statement

- **do** {
-     <**loop body**>
- } **while** (<**test**>);
    - Action is performed first and then the condition is evaluated whether to continue
- Example

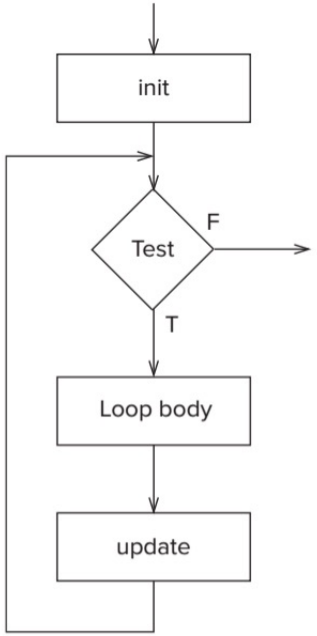
```
#include <stdio.h>
int main(void) {
    int x = 0;
    do {
        printf("%d ", x);
        x += 1;
    } while (x < 10);
}
```



For Statement

- For (**initialization**; **test**; **update**) {
-     **<loop body>**
- }

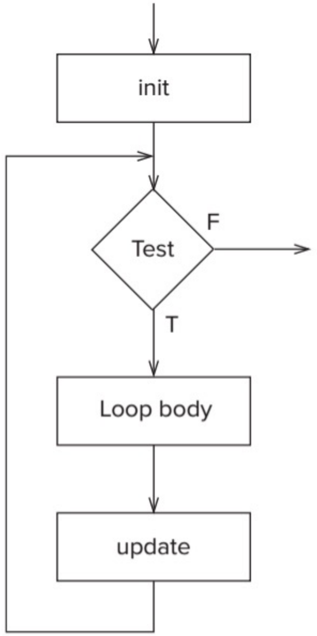
- Example
  - #include <stdio.h>
  - int main(void) {
  -     int x;
  -     **for** (x = 0; x < 10; x++) {
  -         printf(“%d “, x);
  -     }
  - }



For Statement

- Initialization part can be a **declaration**
  - Then, the declared variable's scope is the for statement itself (self contained!)

- Example
  - #include <stdio.h>
  - int main(void) {
  -     **for** (int x = 0; x < 10; x++) {
  -         printf("%d ", x);
  -     }
  - }



For Statement

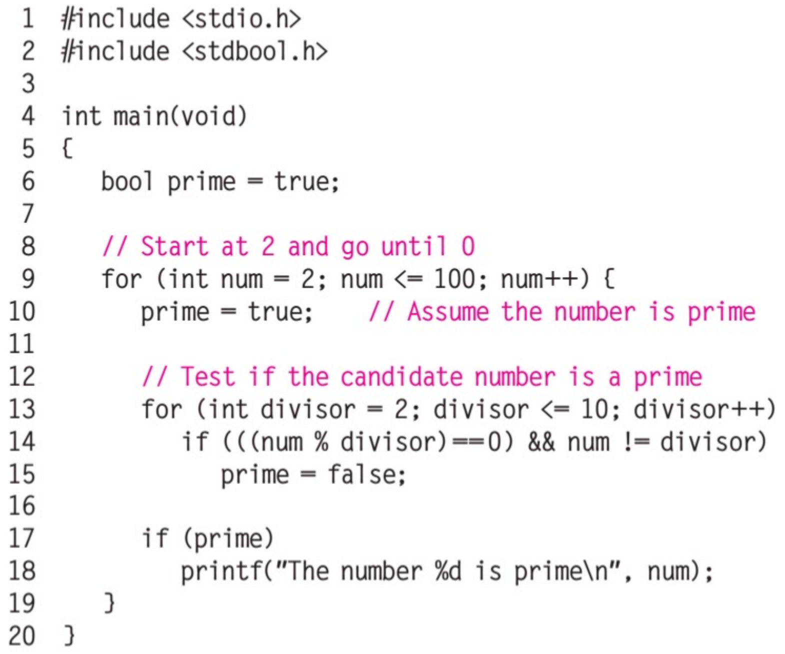
- **Nested Loops** are also provided in C
    - Then, the declared variable’s scope is the for statement itself (self contained!)

- Example

```
#include <stdio.h>
int main(void) {
    for (int multiplicand = 0; multiplicand < 10; multiplicand++) {
        for (int multiplier = 0; multiplier < 10; multiplier++) {
            printf(“%d\t “, multiplier * multiplicand);
        }
        printf(“\n”);
    }
}
```

Example – Prime Numbers

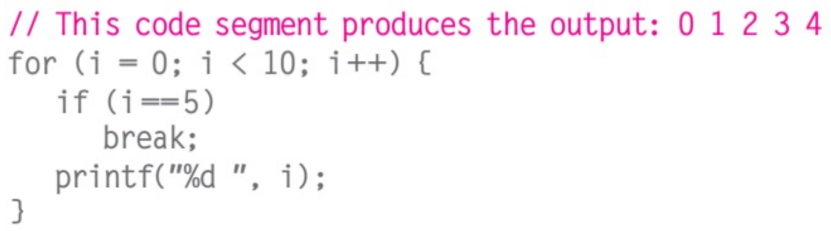
- Find all prime numbers less than 100!
  - Two for loops (not the best algorithms but works ☺)



```c
#include <stdio.h>
#include <stdbool.h>

int main(void)
{
   bool prime = true;

   // Start at 2 and go until 0
   for (int num = 2; num <= 100; num++) {
      prime = true;     // Assume the number is prime

      // Test if the candidate number is a prime
      for (int divisor = 2; divisor <= 10; divisor++)
         if (((num % divisor)==0) && num != divisor)
            prime = false;

      if (prime)
         printf("The number %d is prime\n", num);
   }
}
```

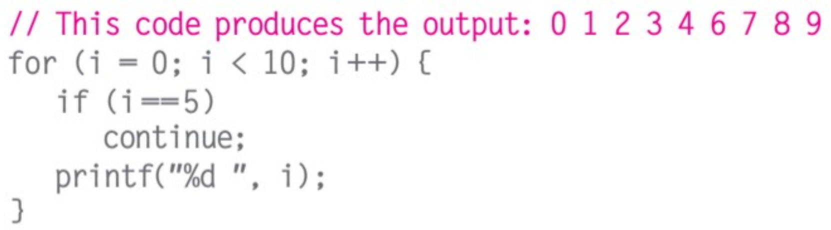
Break and Continue Statements

- **break;** exits a loop or a switch statement right away



```
// This code segment produces the output: 0 1 2 3 4
for (i = 0; i < 10; i++) {
   if (i==5)
      break;
   printf("%d ", i);
}
```

- **continue;** causes only the current iteration to end



```
// This code produces the output: 0 1 2 3 4 6 7 8 9
for (i = 0; i < 10; i++) {
   if (i==5)
      continue;
   printf("%d ", i);
}
```

Switch Statement

- Similar to multiple if-else statements but uses “**case**”
  - Important to include **break;** at the end of each case

- **default** case can be optionally included
  - Executed when no case matches the switch expression
  - If you miss a break statement at the end of a case ‘b’, and if the keypress is ’b’, both the case ‘b’ and the default case will be executed

```
switch (keyPress) {
case 'a':
    // Do statement A
    break;

case 'b':
    // Do statement B
    break;

case 'x':
    // Do statement C
    break;

case 'y':
    // Do statement D
    break;
}
```

Example – A Simple Calculator

```c
#include <stdio.h>

int main(void)
{
    int operand1, operand2; // Input values
    int result = 0;         // Result of the operation
    char operation;         // operation to perform

    // Get the input values
    printf("Enter first operand: ");
    scanf("%d", &operand1);
    printf("Enter operation to perform (+, -, *, /): ")
    scanf("\n%c", &operation);
    printf("Enter second operand: ");
    scanf("%d", &operand2);

    // Perform the calculation
    switch(operation) {
    case '+':
        result = operand1 + operand2;
        break;

    case '-':
        result = operand1 - operand2;
        break;

        case '*':
            result = operand1 * operand2;
            break;

        case '/':
            if (operand2 != 0)    // Error-checking code.
                result = operand1 / operand2;
            else
                printf("Divide by 0 error!\n");
            break;

        default:
            printf("Invalid operation!\n");
            break;
    }

    printf("The answer is %d\n", result);
}
```

Questions?

Thanks!