

Lecture03: Control Flow

3/4/2013

Slides modified from Yin Lou, Cornell CS2022: Introduction to C

1

Administrative things

- Any problem with submitting assignment #1?
- Assignment #2 is on the course webpage due next week.

2

Review from Last Week

- Variable declaration
 - Type, variable name
- C Punctuations
 - ; , “ {} (% #
- printf() : write to the screen
 - printf(**format string**, arg1, arg2, ...)
 - What is a format string? Placeholder tags whose values are substitute by args.
 - What are placeholder tags?
- scanf() : read from the screen (user)
 - scanf(format string, &arg1, &arg2, ...)
 - Store values into locations pointed by args

3

main.c: Variables & Punctuations

```
#include <stdio.h>

int main()
{
    int a, b, c;
    a = 10;
    b = 20;
    c = a * b;
    printf("a = %d b = %d c = %d\n", a, b, c);
    return 0;
}
```

4

More on printf()

- `printf(format string, val1, val2);`
 - format string can include placeholders that specify how the arguments `val1`, `val2`, etc. should be formatted
 - `%c` : format as a character
 - `%d` : format as an integer
 - `%f` : format as a floating-point number
 - `%%` : print a `%` character
- Example

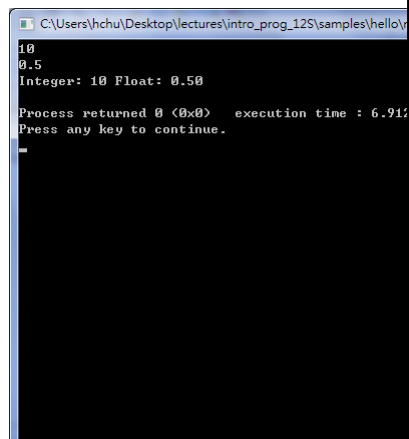
```
double f = 0.95;
printf("f = %f%%\n", f * 100);
```

5

scanf()

```
#include <stdio.h>

int main()
{
    int i;
    double f;
    scanf("%d", &i);
    scanf("%lf", &f);
    printf("Integer: %d Float: %.2f\n", i, f);
    return 0;
}
```



```
C:\Users\hchu\Desktop\lectures\intro_prog_125\samples\hello>
10
0.5
Integer: 10 Float: 0.50
Process returned 0 (0x0)   execution time : 6.911
Press any key to continue.
```

New Stuff: Control Flow

7

Statement

<statement> := <expression>;

```
x = 0;  
++i;           // comments go here: i = i + 1;  
printf("%d", x); /* another way to write comments */
```

8

Blocks

<block> := { <statements> }

```
{  
  x = 0;  
  ++i;  
  printf("%d", x);  
}
```

- A block is syntactically equivalent to a single statement.
 - if, else, while, for
 - Variables can be declared inside any block.
 - There is no semicolon after the right brace that ends a block.

9

Example

```
int x = 0;  
{  
  int x = 5;  
  printf("Inside: x = %d\n", x);  
}  
printf("Outside: x = %d\n", x);
```

Inside: x = 5
Outside: x = 0

10

if Statement

if (<condition>) <statement>

```
// single statement
if (2 < 5)
    printf("2 is less than 5.\n");

// block
if (2 < 5)
{
    printf("I'll always print this line.\n");
    printf("because 2 is always less than 5!\n");
}
```

11

if-else Statement

if (<condition>) <statement1> else <statement2>

```
if (x < 0)
{
    printf("%d is negative.\n", x);
}
else
{
    printf("%d is non-negative.\n", x);
}
```

12

else-if Statement

```

if (a < 5)
    printf("a < 5\n");
else
{
    if (a < 8)
        printf("5 <= a < 8\n");
    else
        printf("a >= 8\n");
}

```

```

if (a < 5)
    printf("a < 5\n");
else if (a < 8)
    printf("5 <= a < 8\n");
else
    printf("a >= 8\n");

```

13

if-else Statement Pitfalls

```

if (a > 70)
    if (a > 80)
        printf("grade = B\n");
    else
        printf("grade < B\n");
else
    printf("Fail.\n");
printf("Done.\n");

```

```

if (a > 70)
{
    if (a > 80)
    {
        printf("grade = B\n");
    }
    else
    {
        printf("grade < B\n");
    }
}
else
{
    printf("Fail.\n");
}
printf("Done.\n");

```

14

Relational Operators

C has the following relational operators

<code>a == b</code>	true iff a equals b
<code>a != b</code>	true iff a does not equal b
<code>a < b</code>	true iff a is less than b
<code>a > b</code>	true iff a is greater than b
<code>a <= b</code>	true iff a is less than or equal to b
<code>a >= b</code>	true iff a is greater than or equal to b
<code>a && b</code>	true iff a is true and b is true
<code>a b</code>	true iff a is true or b is true
<code>!a</code>	true iff a is false

15

Booleans in C

- C DOES NOT have a boolean type.
 - boolean means true or false.
- Instead, conditional operators evaluate to integers (int)
 - 0 indicates false. Non-zero value is true.
 - if (<condition>) checks whether the condition is non-zero.
 - **Programmer must be very careful to this point!**

Example

```
if (3)
    printf("True.\n");
if (!3)
    // unreachable code
if (a = 5)
    // always true, potential bug (a == 5)
int a = (5 == 5); // a = 1
```

16

In-Class Exercise #1

Write a program that reads a number grade (0-100) and converts it into a letter grade (ABCF): A:80-100, B:70-79, C:60-69, and F<60.

Your program should have the following input & output:

Input a number grade> **80**
Letter grade: A

17

Conditional expressions

< condition> ? <expression1> : <expression2>

```
char grade;
grade = (score >= 70 ? 'S' : 'U');
printf("You have %d item%s.\n", n, n == 1 ? "" : "s");
```

Conditional expression often leads to concise code.

18

switch Statement

A common form of if statement

```
if (x == a)
    statement1;
else if (x == b)
    statement2;
...
else
    statement0;
```

switch statement

```
switch (x)
{
    case a: statement1; break;
    case b: statement2; break;
    ...
    default: statement0;
}
```

19

More on switch Statement

Fall-through property

```
int month = 2;
switch (month)
{
    case 1:
        printf("Jan.\n");
        break;
    case 2:
        printf("Feb.\n");
    case 3:
        printf("Mar.\n");
    default:
        printf("Another month.\n");
}
```

```
Feb.
Mar.
Another month.
```

20

More on switch Statement

Fall-through property

```
int month = 2;
int days;
switch (month)
{
    case 2:
        days = 28;
        break;
    case 9:
    case 4:
    case 6:
    case 11:
        days = 30;
        break;
    default:
        days = 31;
}
```

It's always recommended to have default, though it's optional.

```
if (month == 2)
{
    days = 28;
}
else if (month == 9)
|| (month == 6) || (month == 11))
{
    days = 30;
}
```

21

In-Class Exercise #2

Write a program that reads a month (1-12) and converts it into a season: Spring:3-5, Summer:6-8; Fall:9-11 Winter:12-2. **Use the switch statement.**

Your program should have the following input & output:

```
Input a month(1-12)> 1
Season: Winter
```

22

while Loop

while (<condition>) <statement>

- If the condition is initially false, the statement is never executed.

do <statement> while (<condition>);

- The statement is executed at least once.

23

while Loop example

```
#include <stdio.h>

int main()
{
    int i = 1;

    while (i < 10)
    {
        printf("%d ", i);
        i++;          // i = i + 1;
    }
    return 0;
}
```

```
i = 1;
Is i < 10?      Yes, continue

i ++;
Is i < 10?;     Yes, continue
i is now 2;

i ++;
Is i < 10?;     Yes, continue
i is now 3;

...

i ++;
Is i < 10?;     No, exit the loop
i is now 10;
```

24

do Loop example

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

```
int main()
```

```
{
```

```
    int i = 1;
```

```
    do
```

```
    {
```

```
        printf("%d ", i);
```

```
        i++;
```

```
    } while (i < 10);
```

```
}
```

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

```
int main()
```

```
{
```

```
    int i = 1;
```

```
    while (i < 10)
```

```
    {
```

```
        printf("%d ", i);
```

```
        i++;           // i = i + 1;
```

```
    }
```

```
}
```

25

for Loop

for (<exp1>; <exp2>; <exp3>) <statement>

```
exp1;
```

```
while (exp2)
```

```
{
```

```
    statement
```

```
    exp3;
```

```
}
```

```
for (i = 0; i < n; i++)
```

```
{
```

```
    // do something
```

```
}
```

26

for Loop example

```
#include <stdio.h>

int main()
{
    int i;

    for (i=1; i<10; i++)
    {
        printf("%d ", i);
    }
}
```

```
#include <stdio.h>

int main()
{
    int i = 1;

    while (i < 10)
    {
        printf("%d ", i);
        i++;
    }
}
```

```
#include <stdio.h>

int main()
{
    int i = 1;

    do
    {
        printf("%d ", i);
        i++;
    } while (i < 10);
}
```

27

Infinite Loop

```
while (1)
{
    // do something
}
```

```
for (;;)
{
    // do something
}
```

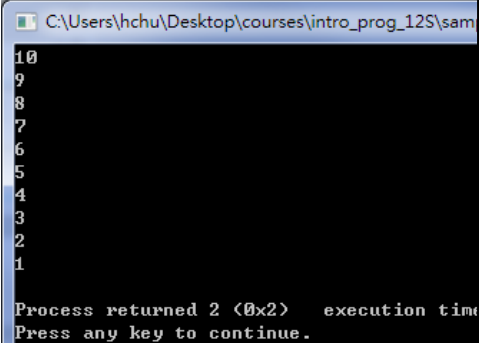
Both are okay, but **for** may lead to fewer machine code on some platform, which means it is slightly more efficient.

28

break exit the loop

```
#include <stdio.h>

int main()
{
    int n = 10;
    while (1)
    {
        if (!n)
        {
            break; // exit the loop
        }
        printf("%d\n", n);
        --n; // similar to n--;
    }
}
```



```
C:\Users\hchu\Desktop\courses\intro_prog_12S\sam
10
9
8
7
6
5
4
3
2
1
Process returned 2 (0x2) execution time
Press any key to continue.
```

continue jump to before the end of loop

```
#include <stdio.h>

int main()
{
    int i;
    for (i = 0; i < 10; ++i)
    {
        if (i == 5)
        {
            continue; /* jump to before the end of the loop */
        }
        printf("%d\n", i);
        /* continue jumps to here */
    }
}
```



```
C:\Users\hchu\Desktop
0
1
2
3
4
6
7
8
9
Process returned
Press any key to
```

Common Pitfalls

Always use {}

```
int i;
for (i = 0; i < 10; ++i);
    printf("%d\n", i);
```

Put literals on the left

```
int n = 10;
while (n = 1)
{
    printf("%d\n", n);
    --n;
}
```

31

How to Avoid Bugs?

Always use {}

```
int i;
for (i = 0; i < 10; ++i)
{
    printf("%d\n", i);
}
```

Put literals on the left

```
int n = 10;
while (1 == n) // 1 = n, likely a bug
{
    printf("%d\n", n);
    --n;
}
```

32

Review for Today

- if, else-if
- < condition> ? <expression1> : <expression2>
- switch, case, default, break
- for (<exp1>; <exp2>; <exp3>) <statement>
- while (<condition>) <statement>
- do <statement> while (<condition>);
- break, continue

33

In-Class Exercise #3

Write a program that print all Fibonacci numbers smaller than 100. Fibonacci numbers are numbers in the following integer sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Your program should output exactly the following line. Note that there should not be a “,” after the last number.

Fib: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

34