APS105
Winter 2012

Jonathan Deber
jdeber -at- cs -dot- toronto -dot- edu

Lecture 7
January 30, 2012

Today

- More Conditionals
- Loops

- Office Hours delayed to 1:30 pm today

Dangling `else`

```
result = 1;
if (x != 0)
  if (y != 0)
    result = x / y;
else
  printf("x must be non-zero. \n");
```

x:	8
y:	4
result:	2

x:	0
y:	4
result:	1

x:	2
y:	0
result:	1

```
x must be non-zero.
```

Dangling `else`

```
result = 1;
if (x != 0)
  if (y != 0)
    result = x / y;
  else
    printf("x must be non-zero. \n");
```

x:	8
y:	4
result:	2

x:	0
y:	4
result:	1

x:	2
y:	0
result:	1

```
x must be non-zero.
```

Dangling `else`

```
result = 1;
if (x != 0)
{
  if (y != 0)
  {
    result = x / y;
  }
}
else
{
  printf("x must be non-zero. \n");
}
```

Short-Circuiting

```
if (x / y > 2 )
{
   // do something
}
```

Short-Circuiting

```
if (y != 0)
{
    if (x / y > 2 )
    {
       // do something
    }
}
```

Short-Circuiting

```
if ((y != 0) && x / y > 2 )
{
   // do something
}
```

```
bool b = true || (x / y > 2);
```

Style

```
bool result;

if ( (x >= MAX) == true )
{
    result = true;
}
else
{
    result = false;
}
```

Style

```
bool result;

if ( (x >= MAX) )
{
    result = true;
}
else
{
    result = false;
}
```

Style

```
bool result = (x >= MAX);
```

Watch Your ;s!

```
if (expression)
   statement
```

; is a statement!

```
if (x > y);
{
    max = x;
}
```

```
if (x > y)
    ;
{
    max = x;
}
```

Watch Your ;s!

```
if (expression)
   statement
```

; is a statement!

```
if (x > y);
{
   max = x;
}
```

```
if (x > y)
    ;
max = x;
```

Other Conditionals

- Ternary conditional operator
- `switch` statement

Ternary ?:

expression ? *expression* : *expression*

condition ? *value-if-true* : *value-if-false*

```
int max = (x > y) ? x : y;
```

```
int max;
if (x > y)
{
   max = x;
}
else
{
   max = y;
}
```

Ternary ?:

```
char status = (grade >= 50) ? 'P' : 'F';
```

```
double d = (total >= 0) ? total : 0.0;
```

```
if (position == 1)
{
    printf("You won gold!\n");
}
else if (position == 2)
{
    printf("You won silver!\n");
}
else if (position == 3)
{
    printf("You won bronze!\n");
}
else
{
    printf("You finished.\n");
}
```

switch

```
switch (position)
{
    case 1:  printf("You won gold!\n");
             break;
    case 2:  printf("You won silver!\n");
             break;
    case 3:  printf("You won bronze!\n");
             break;
    default: printf("You finished.\n");
             break;
}
```

Equality Operators

- Also used to compare two values

Warning!
= vs ==

Symbol	*Meaning*
==	equal to
! =	not equal to

`bool` is faked

- Why do we need `#include <stdbool.h>`?
- C doesn't actually have Booleans
- Instead, uses `int`s
  - `0` is `false`
  - everything else is `true`

The file `stdbool.h` contains

```
#define true  1
#define false 0
```

int conditionals

```
if (1)
{
   printf("This would be printed.");
}

if (0)
{
   printf("This would not.");
}
```

`int` conditionals

```
bool b = false;
b++;
b = 1;
b = true;
```

`true`

Don't do this!

int conditionals

```
bool a = 1;
bool b = 1;

if ( (a + b) == 2 )
{
  printf("Hello\n");
}
```

Bad Style

```
bool a = true;
bool b = true;

if ( a && b)
{
  printf("Hello\n");
}
```

int conditionals

```
bool a = 1;
bool b = 2;

if ( (a + b) == 2 )
{
  printf("Hello\n");
}
```

Bad Style

```
bool a = true;
bool b = true;

if ( a && b)
{
  printf("Hello\n");
}
```

Conversion Specifier

- `printf` doesn't have %b
- You can print them as `int`, will get `1` or `0`
- Don't read them as `1` or `0` into an `int` with %d

```
bool b = true;
printf("b is %d", b);
scanf("%d", &b);
```

Wrong

```
b is 1
```

- Read them as `char`, then convert to `bool`

Equality Operators

- Also used to compare two values

Warning!
= vs ==

Symbol	*Meaning*
==	equal to
! =	not equal to

Value of an Expression

`2 + 4`	`6`
`8 * 2 - 1`	`15`
`3 <= 5`	`true`
`29 == 29`	`true`

Value of an Expression

`2 + 4`	`6`
`8 * 2 - 1`	`15`
`3 <= 5`	`true`
`29 == 29`	`true`
`x = 7`	

Value of an Expression

```
2 + 4
```

6

```
8 * 2 - 1
```

15

```
3 <= 5
```

true

```
29 == 29
```

true

```
x = 7
```

x: 7

Value of an Expression

```
2 + 4          6

8 * 2 - 1      15

3 <= 5         true

29 == 29       true

x = 7          7        x: 7
```

Value of an Expression

```
2 + 4
```
6

```
8 * 2 - 1
```
15

```
3 <= 5
```
true

```
29 == 29
```
true

```
x = 7
```
7

x: 7

```
y = (z = 9);
```

Value of an Expression

```
2 + 4
```
6

```
8 * 2 - 1
```
15

```
3 <= 5
```
true

```
29 == 29
```
true

```
x = 7
```
7

x: 7

```
y = (z = 9);
```

z: 9

Value of an Expression

```
2 + 4
```
6

```
8 * 2 - 1
```
15

```
3 <= 5
```
true

```
29 == 29
```
true

```
x = 7
```
7

x: 7

```
y = (z = 9);
```
9

z: 9

Value of an Expression

`2 + 4`	`6`
`8 * 2 - 1`	`15`
`3 <= 5`	`true`
`29 == 29`	`true`
`x = 7`	`7`
`y = (z = 9);`	`9`

`x:` `7`

`y:` `9`

`z:` `9`

Value of an Expression

```
2 + 4
```
6

```
8 * 2 - 1
```
15

```
3 <= 5
```
true

```
29 == 29
```
true

```
x = 7
```
7

x: 7

```
y = (z = 9);
```
9

y: 9
z: 9

```
a = b = 4;
```

Value of an Expression

```
2 + 4
```
6

```
8 * 2 - 1
```
15

```
3 <= 5
```
true

```
29 == 29
```
true

```
x = 7
```
7

x: 7

```
y = (z = 9);
```
9

y: 9

z: 9

```
a = b = 4;
```

b: 4

Value of an Expression

```
2 + 4
```
6

```
8 * 2 - 1
```
15

```
3 <= 5
```
true

```
29 == 29
```
true

```
x = 7
```
7

x: 7

```
y = (z = 9);
```
9

y: 9

z: 9

```
a = b = 4;
```
4

b: 4

Value of an Expression

`2 + 4`	`6`
`8 * 2 - 1`	`15`
`3 <= 5`	`true`
`29 == 29`	`true`
`x = 7`	`7`
`y = (z = 9);`	`9`
`a = b = 4;`	`4`

`x:` `7`

`y:` `9`
`z:` `9`

`a:` `4`
`b:` `4`

When = does not equal ==

```
#define MAX 10

...

if (num = MAX)
{
    printf("Full.\n")
}
else
{
    printf("Still room.\n")
}
```

num: 5

Full.

When = does not equal ==

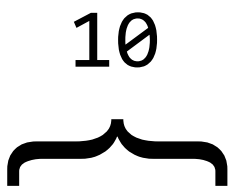
```
#define MAX 10

...

if (num = MAX)
{
    printf("Full.\n")
}
else
{
    printf("Still room.\n")
}
```

num: ~~5~~ 10

Full.

When = does not equal ==

```
#define MAX 10
...
if (num = MAX)
{
    printf("Full.\n")
}
else
{
    printf("Still room.\n")
}
```



num: ~~5~~ 10

Full.

Loops

Repetition

```
printf("%d\n", 1);
printf("%d\n", 2);
printf("%d\n", 3);
printf("%d\n", 4);
printf("%d\n", 5);
printf("%d\n", 6);
printf("%d\n", 7);
printf("%d\n", 8);
printf("%d\n", 9);
printf("%d\n", 10);
```

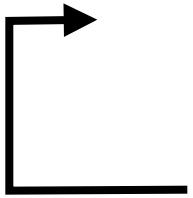
Repetition

```
printf("%d\n", 1);
printf("%d\n", 2);
printf("%d\n", 3);
printf("%d\n", 4);
printf("%d\n", 5);
printf("%d\n", 6);
printf("%d\n", 7);
printf("%d\n", 8);
printf("%d\n", 9);
printf("%d\n", 10);
```

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

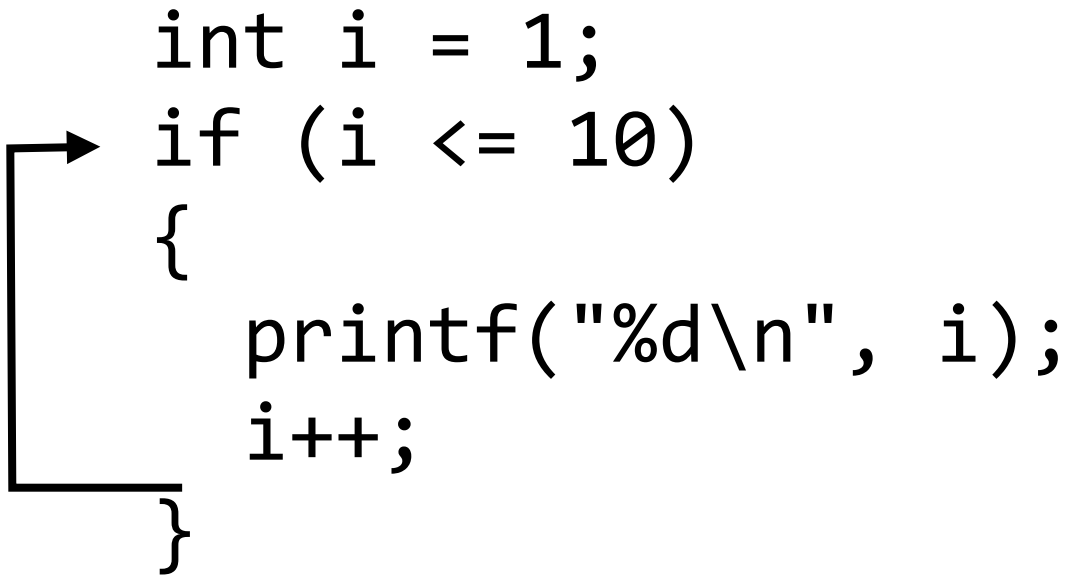
`while` statements

```
while (expression)
  statement
```



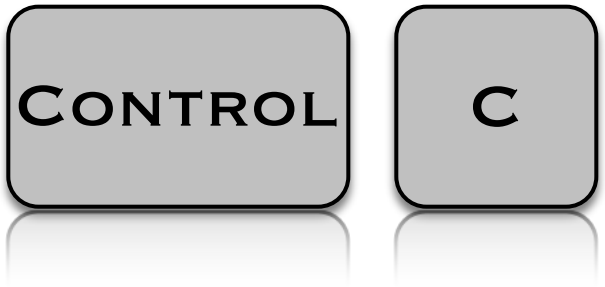
```
if (expression)
  statement
```

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



Infinite Loops

```
int i = 1;
while (true)
{
    printf("%d\n", i);
    i++;
}
```



Infinite Loops

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

```
10
10
10
10
10
10
10
10
10
10
10
10
10
10
10
10
10
```

Infinite Loops

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

```
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

Infinite Loops

```
int i = 0;

while (true)
{
  printf("Enter a number: ");
  scanf("%d", &i);
  printf("%d squared is %d \n", i, i * i);
}

printf("All done.\n");
```

⟵ Never executed

Zero Iterations

```
while (false)
{
  printf("Never printed... \n");
}

printf("But this is. \n");
```

```
int i = 0;
printf("Enter a number: ");
scanf("%d", &i);

printf("Beginning countdown...\n");

while (i >= 0)
{
   printf("%d\n", i);
   i--;
}

printf("Finished countdown.\n");
```

```
int num = 0;
printf("Enter a number: ");
scanf("%d", &num);

int sum = 0;

int i = 1;
while (i >= num)
{
    sum += i;
    i++;
}

printf("Sum of 1 to %d is %d.\n", num, sum);
```

do/while

```
int num;
printf("Enter a positive number: ");
scanf("%d", &num);

while (num < 0)
{
   printf("Enter a positive number: ");
   scanf("%d", &num);
}

// At this point, num must be positive
```

do/while

```
int num;



do
{
   printf("Enter a positive number: ");
   scanf("%d", &num);
} while (num < 0);

// At this point, num must be positive
```