Engineering Computing I

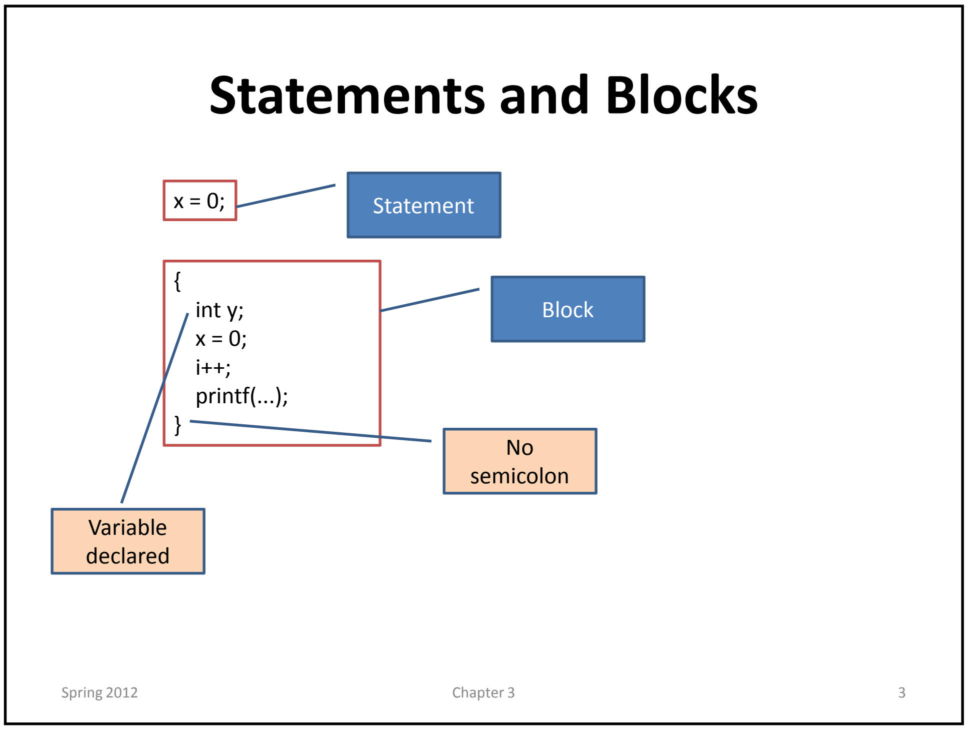
The C programming Language

Chapter 3

Control Flow

Chapter 3 - Control Flow

The control-flow of a language specify the order in which computations are performed



Statements and Blocks

```
x = 0;
```

Statement

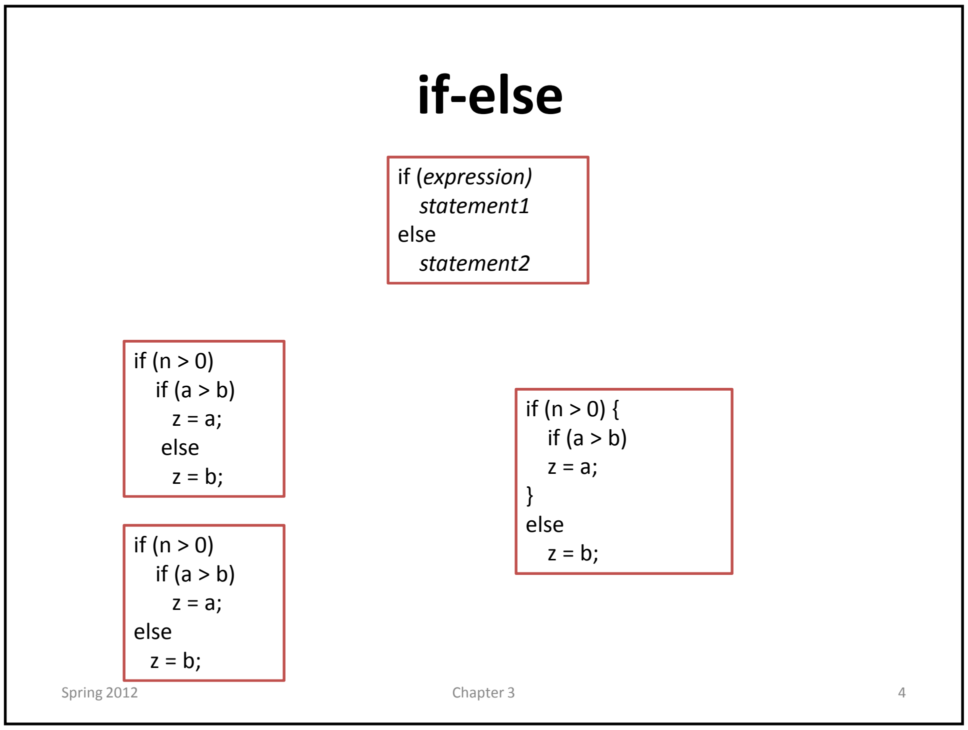
```
{
   int y;
   x = 0;
   i++;
   printf(...);
}
```

Block

No semicolon

Variable declared

Spring 2012 Chapter 3 3



if-else

```
if (expression)
   statement1
else
   statement2
```

```
if (n > 0)
   if (a > b)
      z = a;
    else
      z = b;
```

```
if (n > 0)
   if (a > b)
      z = a;
else
   z = b;
```

```
if (n > 0) {
   if (a > b)
   z = a;
}
else
   z = b;
```

Spring 2012 Chapter 3 4

else If

```
if (expression)
    statement
else if (expression)
    statement
else if (expression)
    statement
else if (expression)
    statement
else
    statement
```

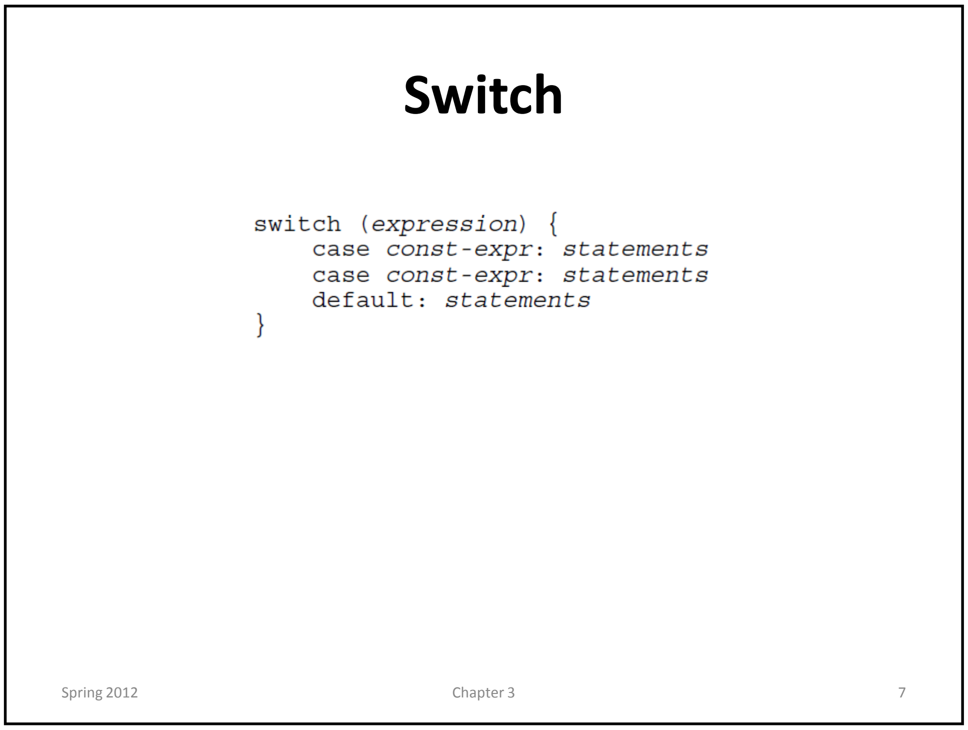
```
if (x < v[mid])
    high = mid + 1;
else if (x  > v[mid])
    low = mid + 1;
else
    return mid;
```

Exercise

Using ***else if*** statement, write a program that analyzes the variable ***SeaTemp*** (initialized to ***75***) and decides the sea-water condition based on the following table:

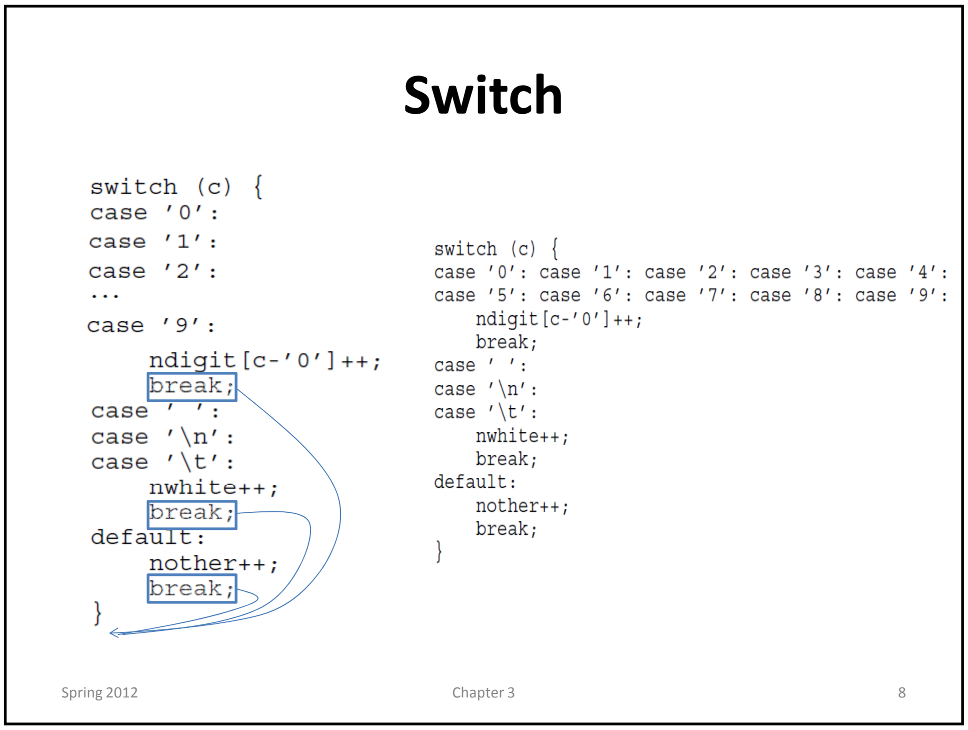
Sea Temperature	Condition
SeaTemp <= 60	“cold”
60 < SeaTemp <= 80	“pleasant”
80< SeaTemp <= 90	“warm”
90 < SeaTemp	“uncomfortably warm”

Then set ***SeaTemp*** to different values to test your program.



Switch

```
switch (expression) {
    case const-expr: statements
    case const-expr: statements
    default: statements
}
```



Switch

```
switch (c) {
case '0':
case '1':
case '2':
...
case '9':
     ndigit[c-'0']++;
     break;
case ' ':
case '\n':
case '\t':
     nwhite++;
     break;
default:
     nother++;
     break;
}
```

```
switch (c) {
case '0': case '1': case '2': case '3': case '4':
case '5': case '6': case '7': case '8': case '9':
    ndigit[c-'0']++;
    break;
case ' ':
case '\n':
case '\t':
    nwhite++;
    break;
default:
    nother++;
    break;
}
```

Exercise

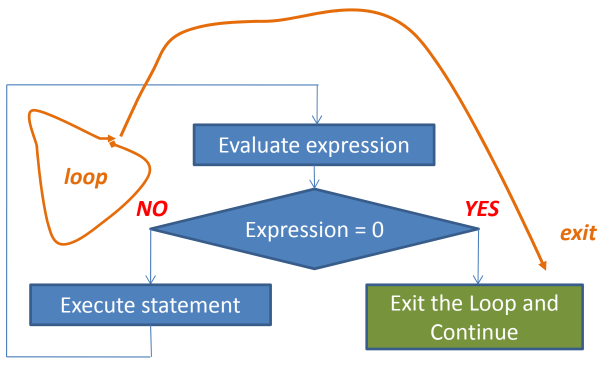
Using a ***switch*** statement, write a program that analyzes the variable ***SeaTemp*** (initialized to **75**) and decides the sea-water condition based on the following table:

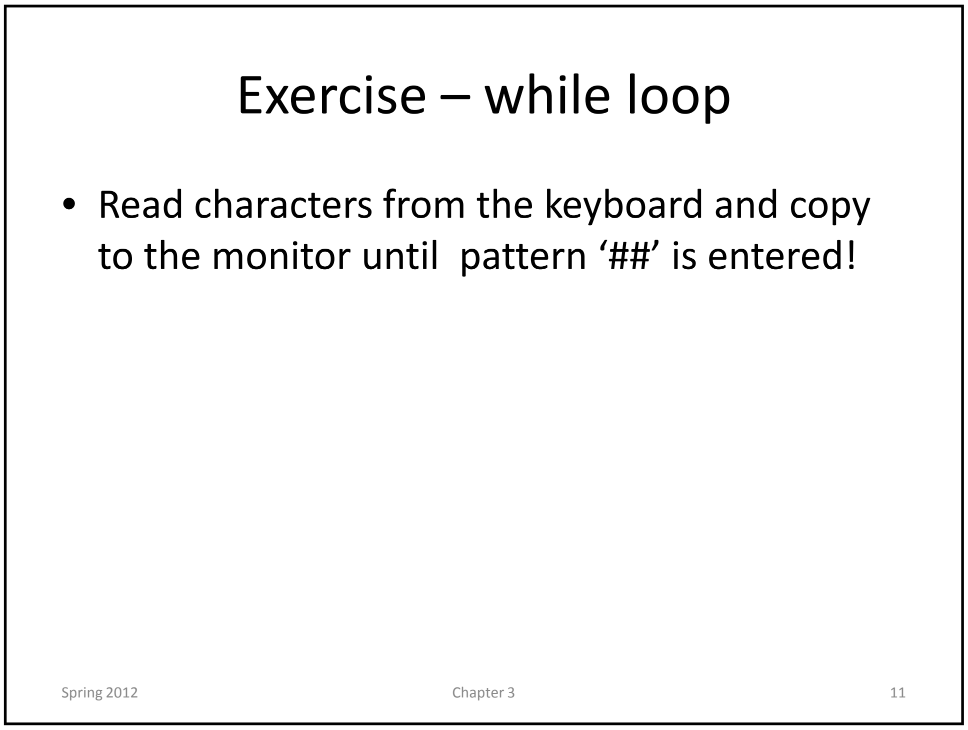
Sea Temperature	Condition
SeaTemp <= 60	"cold"
60 < SeaTemp <= 80	"pleasant"
80< SeaTemp <= 90	"warm"
90 < SeaTemp	"uncomfortably warm"

Then set ***SeaTemp*** to different values to test your program.

Loops - While

```
while (expression)
    statement
```

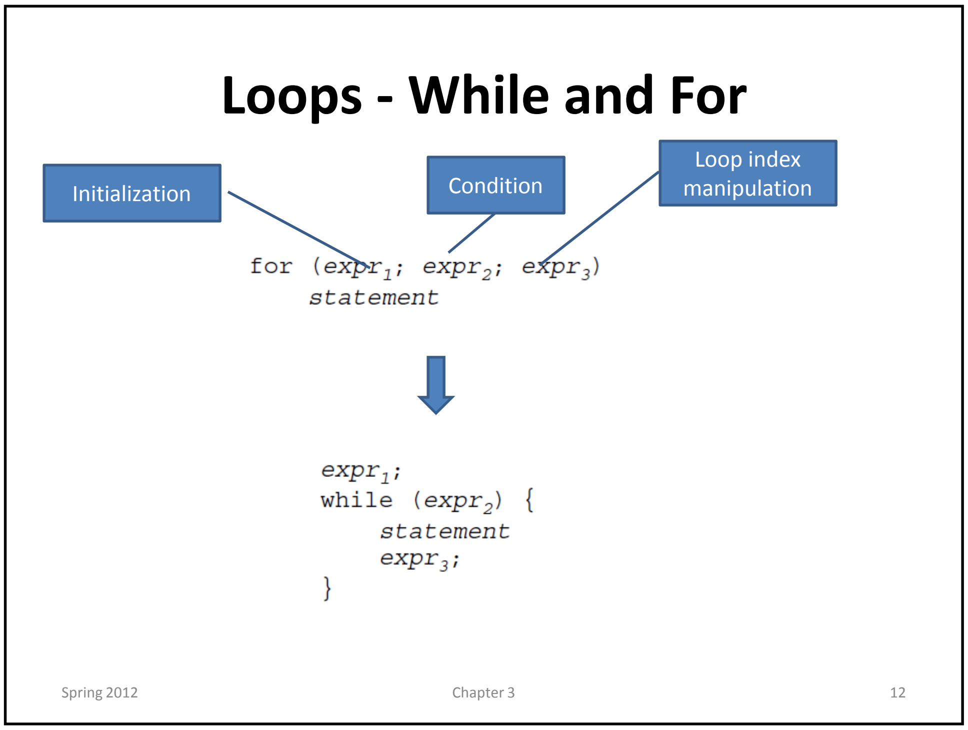




Exercise – while loop

- Read characters from the keyboard and copy to the monitor until pattern '##' is entered!

Spring 2012 Chapter 3 11



Loops - While and For

```
for (expr1; expr2; expr3)
     statement
```

```
expr1;
while (expr2) {
     statement
     expr3;
}
```

Spring 2012 Chapter 3 12

While Vs. For

- Matter of personal preference which one to use

while ((c = getchar()) == ' ' || c == '\n' || c = '\t');

- No initialization, so while is most natural
- For is preferable with simple initialization and increment

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

Exercise – for loop

- Write a program to print out the squares and cubes of the first 100 positive integers. Display a proper heading for the generated table.

Nested for loops

Show the output of the following program by manually going through the code.

```
#include <stdio.h>
/* Simple nested for loops example */
#define N   2
#define M   3

main()
{
    int i, j;

    for (i=N; i >= 0; i--)
        for (j=0; j <= M; j++)
                printf("%d\t%d\n", i, j);
}
```

Loops - Do-While

```
do
    statement
while (expression);
```

```
do {
    s[i++] = n % 10 + '0';
} while ((n /= 10) > 0);
```

Break and Continue

- The ***break*** statement provides an early exit from ***for***, ***while***, and ***do***, just as from ***switch***.
- A ***break*** causes the innermost enclosing ***loop*** or ***switch*** to be exited immediately

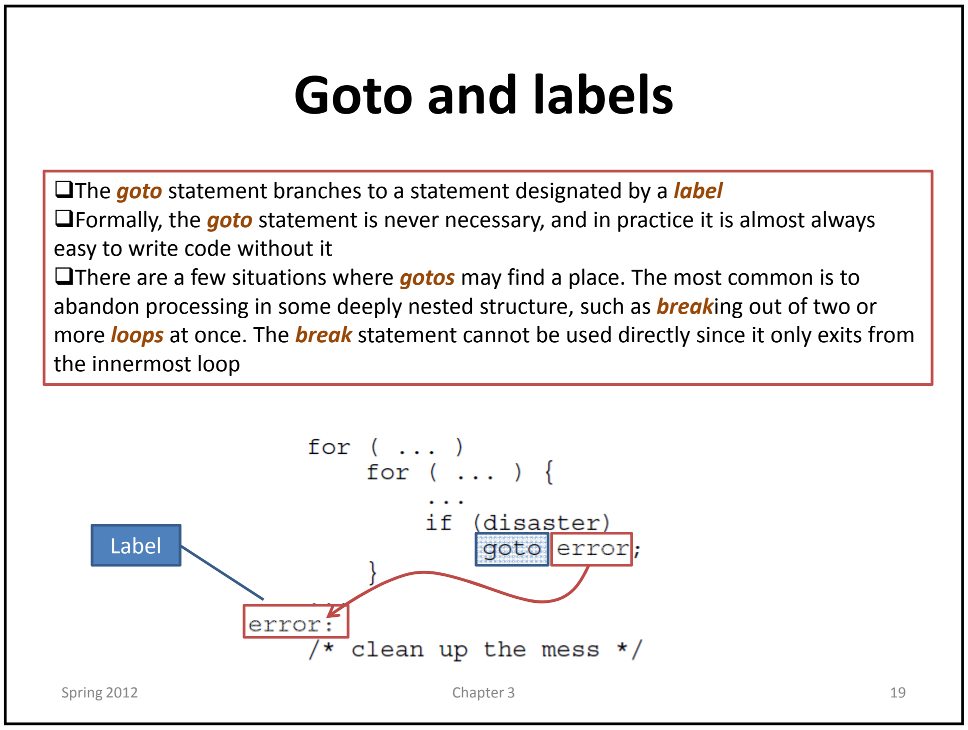
```
/* trim:  remove trailing blanks, tabs, newlines */
int trim(char s[])
{
    int n;

    for (n = strlen(s)-1; n >= 0; n--)
        if (s[n] != ' ' && s[n] != '\t' && s[n] != '\n')
            break;
    s[n+1] = '\0';
    return n;
}
```

Break and Continue

- The ***continue*** statement causes the next iteration of the enclosing ***for***, ***while***, or ***do*** loop to begin.
- In the ***while*** and ***do***, this means that the test part is executed immediately; in the ***for***, control passes to the increment step.
- The continue statement applies only to ***loops***, not to ***switch***.

```
for (i = 0; i < n; i++)
    if (a[i] < 0)   /* skip negative elements */
        continue;
    ... /* do positive elements */
```



Goto and labels

❑The ***goto*** statement branches to a statement designated by a ***label***
❑Formally, the ***goto*** statement is never necessary, and in practice it is almost always easy to write code without it
❑There are a few situations where ***gotos*** may find a place. The most common is to abandon processing in some deeply nested structure, such as ***break***ing out of two or more ***loops*** at once. The ***break*** statement cannot be used directly since it only exits from the innermost loop

```
        for ( ... )
            for ( ... ) {
                ...
                if (disaster)
                    goto error;
            }
        ...
    error:
        /* clean up the mess */
```