

# CSA2090: Systems Programming Introduction to C

## Lecture 3: Control Flow

Dr. Christopher Staff  
Department of Computer Science & AI  
University of Malta

# Aims and Objectives

- if and switch: branching
- for and while: looping
- Love: Ch. 5

# IF

- General form:

```
if <expression1> {  
    <statement1;*>  
} else if <expression2> {  
    <statement2;*>  
} else {  
    <statement3;*>  
}
```



# IF examples

```
int a = 5;  
if (a == 5) {  
    printf("Yes\n");  
} else {  
    printf("No\n");  
}
```

conditional expression  
in brackets

ALL C statements return a value,  
so any C statement can be an  
expression

# IF examples

```
int a = 5;  
if (a == 5) {  
    printf("Yes\n");  
} else {  
    printf("No\n");  
}
```

Beware == and !=!!

ALL C statements return a value,  
so any C statement can be an  
expression, e.g., if (a == 5) { ... }



# IF examples

```
int a = 5, b = 7;
if (a > 5) {
    a = a + b;           and
} else if (b == 7 && a != 5) {           not equal to
    b = b * a;
} else if (b != 5 || a < 8) {           or
    a = b;
} else {
    printf("Confused! \n");
}
```



# IF examples

```
int a = 5, b = 7;  
if (a == b)  
    printf("Yes\n");  
else {  
    a = b;  
    printf("No\n");  
}
```

# IF examples

```
int a = 5, b = 7;  
if (a == b)  
    printf("Yes\n");  
else {  
    a = b;  
    printf("No\n");  
}
```

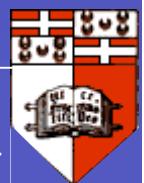
braces needed for two or  
more statements in a block



# SWITCH

- Like case in Pascal...

```
switch(i) {  
    case value1:  
        <statement1; >  
        break;  
    case value2:  
        <statement2; >  
    default:  
        <statement3; >  
}
```



# SWITCH

- Like case in Pascal...  

```
switch(i) {  
    case value1:  
        <statement1; >  
        break; ← break prevents control  
    case value2:  
        <statement2; >  
    default: ← default case for cases  
        <statement3; >  
}
```

*value* must be an integer value

*break* prevents control falling into next case

*default* case for cases that match no *value*



# SWITCH example

```
switch(i) {  
    case 1:  
        printf("i is one\n");  
        break;  
    case 2:  
        printf("i is two\n");  
        break;  
    default:  
        printf("i is another value\n");  
}
```



# Loops: WHILE

```
while (i < 30) {  
    something();  
    ...  
}
```

# Loops: WHILE

```
do  {  
    something();  
} while (i < 30);
```

# Loops: FOR

```
for ([expr1]; [expr2]; [expr3]) {  
    something();  
}  
all expressions are optional
```

```
for (i = 0; i < 10; i++)  
{  
    something();  
}
```

# Loops: FOR

```
for ([expr1]; [expr2]; [expr3]) {  
    something();  
}
```

```
for (i = 0; i < 10; i++)  
{  
    something();  
}
```

abbreviated form  
of  $i = i + 1$



# Break and Continue

- `break` breaks out of a loop (forced termination)
- `continue` forces early execution of the next loop

# Break and continue

```
while (a <= 100) {  
    if (a / 2 == a / 2.0)  
        continue;  
    else if (a + b > 200)  
        break;  
    a = (a * b * b) / (a + b);  
}
```

see break.c



# Loops

- How would you loop for infinity, ending only when a break; is encountered?
  - using while?
  - using for?

# Debugging

- C is not particularly friendly
- Compiler error messages can be mysterious
  - see errors.c
  - then try gcc -Wall errors.c
- Poor layout of programs can be misleading
  - see mislead.c

# Useful tools

- Lint, or *gcc -Wall*
- cb: adjusts indentation
- cflow: shows function calls
- gdb: graphical debugger
  - must first compile prog with -g flag
- cdecl: explains complicated declarations

# Common mistakes

- Using = instead of ==
- Nested comments
  - can use cpp #if 0 ... #endif surrounding code instead

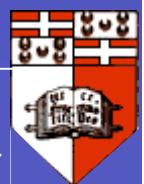
```
#if 0  
/* I want this commented out */  
#endif
```



# Common mistakes

```
while (a<100) ;  
    a += j;  
    printf("%d\n", a);  
...  
for (a = 0; a < 100; a+=j) ;  
    printf("%d\n", a);
```

A yellow arrow points from the text "NULL statement" to the semicolon at the end of the first line of code.



# Common mistakes

- Always use braces if you have nested if-statements
- Get into the habit of using braces everywhere!

# Common mistakes

$a[i++] = b[i++]$

- What's the value of  $i$  in each case?

# Common mistakes

$a[i++] = b[i++]$

- What's the value of  $i$  in each case?
  - Depends on whether execution is left-to-right or right-to-left!

# Next week...

- Lab sessions
  - Work through Love, Exercises 1
  - Attempt questions 1 and 2 at home
  - Lab session will address any problems you have, and tackle questions 3 and 4.