COMP26020 Programming Languages and Paradigms -- Part 1

Conditionals and Loops

Control Flow

- Statements end with `;` and are **executed sequentially**

```c
#include <stdio.h>

int main() {
    printf("hello!\n");     // print hello first
    printf("goodbye!\n");   // then print goodbye
    return 0;               // then return
}
```

Conditionals

Conditionals

```c
if (x == 50) {
    printf("The value of x is 50\n");
}
```

```c
if (x == 50) {
    printf("The value of x is 50\n");
} else {
    printf("The value of x is not 50\n");
}
```

```c
if(x < 50) {
    printf("The value of x is strictly less than 50\n");
} else if (x == 50) {
    printf("The value of x is exactly 50\n");
} else {
    printf("The value of x is strictly more than 50\n");
}
```

06-conditionals-loops/conditional.c

- Condition: `<`, `<=`, `>`, `>=`, `==` (equality), `!=` (difference)
- **Do not use `=` for equality!**

Conditionals

- "false" is `0`, "true" is everything but `0`
- Boolean operations: `&&` (logical AND), `||` (logical OR), `!` (negation)

```c
int one = 1;
int zero = 0;

if (one) { /* will take this branch */}
if (zero) { /* won't be taken */}
if (!one) { /* won't take */}
if (one || zero) { /* will take */ }
if (one && zero) { /* won't take */ }
if (one && zero || one && zero) { /* won't take, && evaluated before || */ }
if (one && (one || zero)) { /* will take */}
```

06-conditionals-loops/bool-in-conditional.c

- Beware of operators precedence:
  - `!` first, then `&&`, then `||`
  - Use parentheses to modify/clarify evaluation order

See complete list: https://en.cppreference.com/w/c/language/operator_precedence

Switch Statement

- Test the equality of an integer against a list of values

```c
switch(x) {
    case 1:
        printf("x is 1\n");
        break;

    case 2:
        printf("x is 2\n");
        break;

    case 3:
        printf("x is 3\n");
        break;

    default:
        printf("x is neither 1, nor 2, nor 3\n");
}
```

06-conditionals-loops/switch.c

- Use `break` to avoid fall through in cases below
- `default` path taken when none of the other cases matches

Loops

While Loop

Keep executing loop body while condition is true

```c
int x = 10;
printf("------ while loop:\n");
while (x > 0) {
    printf("x is %d\n", x);
    x = x - 1;
}

x = 10;
printf("------ do ... while loop:\n");
do {
    printf("x is %d\n", x);
    x = x - 1;
} while (x > 0);
```

06-conditionals-loops/while.c

`while ...` checks the condition before entering the loop body and `do ... while` checks after a first iteration

While Loops

```c
int x = -1;

while (x > 0) { /* won't enter the loop body, condition checked before */
    printf("x is %d\n", x);
    x = x - 1;
}

do {
    printf("x is %d\n", x);
    x = x - 1;
} while (x > 0); // still does one iteration, condition checked after the
                 // loop body
```

06-conditionals-loops/do-while.c

While Loops

```c
int x = -1;

while (x > 0) { /* won't enter the loop body, condition checked before */
    printf("x is %d\n", x);
    x = x - 1;
}

do {
    printf("x is %d\n", x);
    x = x - 1;
} while (x > 0); // still does one iteration, condition checked after the
                 // loop body
```

06-conditionals-loops/do-while.c

What about this:

```c
while(1) { printf("hello\n"); }
```

06-conditionals-loops/infinite-loop.c

Use `ctrl` + `C` to kill a stuck program

For Loops

- `for (<init stmt> ; <cond> ; <update stmt>) {<body>}`
  - `init stmt` executed once before the first iteration
  - `cond` checked before each iteration
  - `update stmt` executed after each iteration

```c
int i;

for(i = 0; i<10; i = i + 1) {
    printf("i is %d\n", i);
}

/* we can declare the iterator as part of the loop header */
for(int j = 0; j<10; j = j + 1) {
    printf("j is %d\n", j);
}
```

06-conditionals-loops/for.c

For and While Loops: Break and Continue

- `break` within a loop body simply exits the loop

```c
for(int i = 0; i < 10; i = i + 1) {
    printf("iteration %d\n", i);

    if(i == 5) {
        break;
    }
}
```

06-conditionals-loops/break.c

For and While Loops: Break and Continue

- `break` within a loop body simply exits the loop

```c
for(int i = 0; i < 10; i = i + 1) {
    printf("iteration %d\n", i);

    if(i == 5) {
        break;
    }
}
```

06-conditionals-loops/break.c

- `continue` within a loop body directly goes to the next iteration

```c
int i = 0;
while(i < 10) {
    i = i + 1;

    if(i == 5) {
        continue;
    }

    printf("iteration %d\n", i);
}
```

06-conditionals-loops/continue.c

Loops and Arrays

```c
int my_array[4];

for(int i=0; i<4; i = i +1) {
    my_array[i] = 100 + i;
    printf("index %d contains: %d\n", i, my_array[i]);
}

int my_2d_array[3][2];

for(int i=0; i<3; i++) {
    for(int j=0; j<2; j++) {
        my_2d_array[i][j] = i*j;
        printf("Index [%d][%d] = %d\n", i, j, my_2d_array[i][j]);
    }
}
```

06-conditionals-loops/loop-array.c

Back on Command Line Argument

```c
#include <stdio.h>

int main(int argc, char **argv) {

    printf("Number of command line arguments: %d\n", argc);

    for(int i=0; i<argc; i++)
        printf("argument %d: %s\n", i, argv[i]);

    return 0;
}
```

Back on Command Line Argument

```c
int print_help_and_exit(char **argv) {
    printf("Usage: %s <number 1> <number 2>\n", argv[0]);
    exit(-1);
}

/* Sum the two integers passed as command line integers */
int main(int argc, char **argv) {
    int a, b;

    if(argc != 3)
        print_help_and_exit(argv);

    a = atoi(argv[1]);
    b = atoi(argv[2]);

    printf("%d + %d = %d\n", a, b, a+b);

    return 0;
}
```

cmdline-args-conditional.c

Braces in Conditionals and Loops

- Conditional/loop body is a single statement? can omit the braces

```
if (x)
    printf("x is non null\n");
else
    printf("x is 0\n");

for (int i = 0; i<10; i = i +1)
    printf("i is %d\n", i);
```

- **Warning: if adding a second statement, put the braces back**
  - See the Apple bug: https://bit.ly/3ktvgF4
- **Also, ambiguity:**

```
if(one)
    if(two)
        foo();
  else            // whose else is this?
        bar();
```

Summary

- Conditionals: `if`, `switch`
- Loops: `while`, `for`

Feedback form: https://bit.ly/3jxgOxf