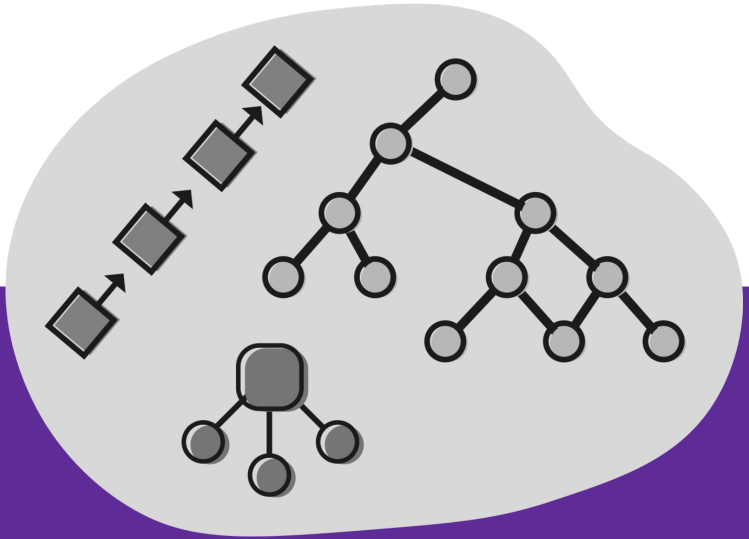
COMP2521

Data Structures and Algorithms



Blake Morris

Introductions ✨

- What did you learn in COMP1511?
- What are you still unsure of?

New C Syntax ⌨️

New C Syntax - the *for* loop

```c
int i = 10;                 // Where do we start?
while (i >= 0) {            // When do we stop?
    printf(“%d\n”, i);
    i++;                    // How do we iterate?
}
```

New C Syntax - the *for* loop

```
int i = 10;
while (i >= 0) {
    printf(“%d\n”, i);
    i++;
}
```

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

New C Syntax - the *for* loop

```c
int i = 10;
while (i >= 0) {
    printf(“%d\n”, i);
    i++;
}
```

```c
for (int i = 10; i >= 0; i++) {
    printf(“%d\n, i);
}
```

New C Syntax - the *for* loop

```c
int i = 10;
while (i >= 0) {
    printf(“%d\n”, i);
    i++;
}
```

```c
for (int i = 10; i >= 0; i++) {
    printf(“%d\n, i);
}
```

New C Syntax - the *for* loop

```c
int i = 10;
while (i >= 0) {
    printf(“%d\n”, i);
    i++;
}
```

```c
for (int i = 10; i >= 0; i++) {
    printf(“%d\n, i);
}
```

New C Syntax - early *`return`*, *`break`*, *`continue`*

- Sometimes we want to exit a loop or function early

```c
for (int i = 0; i < 100; i++) {
    if (i % 5 == 0) continue;
    printf(“%d\n”, i);
}

for (int i = 0; i < 100; i++) {
    if (i % 5 == 0) break;
    printf(“%d\n”, i);
}
```

New C Syntax - *switch* statement

- Simplifies a long list of if/else statements

```
switch (n) {
    case 1: number = "one"; break; // break exits the switch
    case 2: number = "two"; break;
    case 3: number = “three”; break;
    /* … */
    default: number = “???”; break;
}
```

New C Syntax - conditional expression

```
CONDITION ? VALUE IF TRUE : VALUE IF FALSE
```

What will this conditional expression evaluate to?

```
int x = 4;
char *parity = x % 2 == 0 ? “even” : “odd”;
```

New C Syntax - conditional expression

```c
int max(int a, int b) {
    int m;
    if (a >= b) {
        m = a;
    } else {
        m = b;
    }
    return m;
}
```

```c
int max(int a, int b) {
    return a >= b ? a : b;
}
```

New C Syntax - *typedef*

```
typedef int Integer;

struct point {
    int x;
    int y;
};
typedef struct point Point;

struct node {
    int value;
    struct node *next;
};
typedef struct node *Node;
```

Linked Lists (*Revision*) 🔗

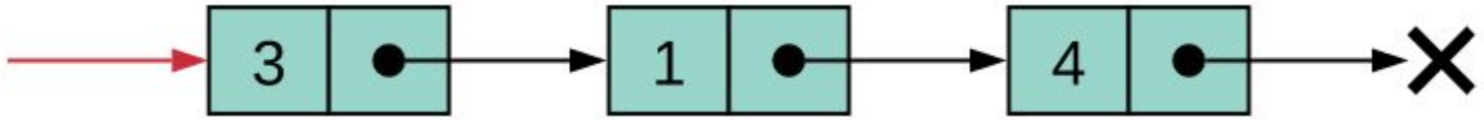
Linked Lists - Representations

```c
// Representation 1


struct node {
    int value;
    struct node *next;
};

typedef struct node *List;
```

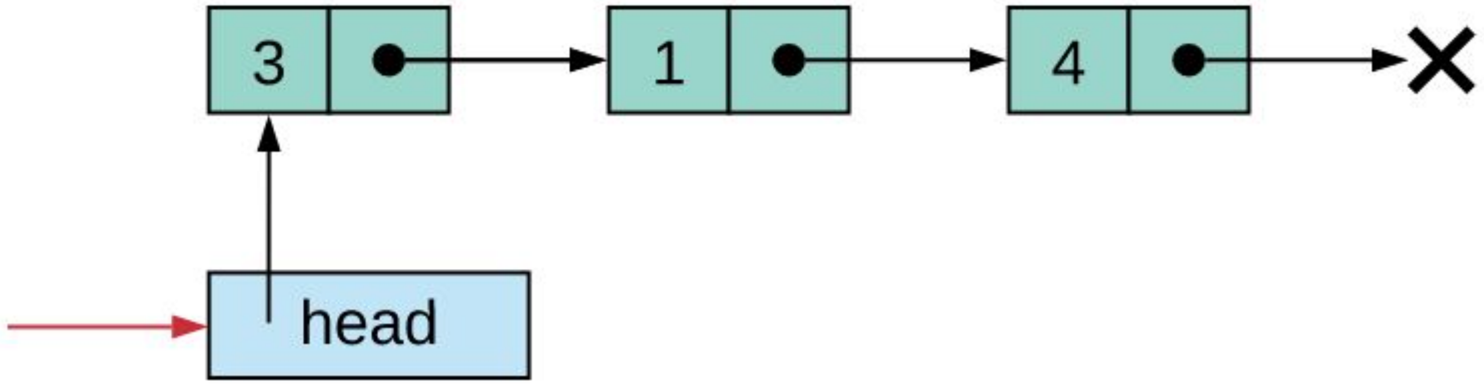
```c
// Representation 2


struct node {
    int value;
    struct node *next;
};

struct list {
    int value;
    struct node *next;
};

typedef struct list *List;
```

Linked Lists - Representations

1.



2.



C (*Revision*)

Command Line I/O

Consider a program called myprog which is invoked as:

```
$ ./myprog hello there, 'John Shepherd' > myFile
```

- What are the values of *argc* and *argv*?
- Where would the statement `printf("%s", argv[1]);` place its output?
- Where would the statement `ch = getchar();` get its input?

Pointers and Memory

```c
int x, y;
char *c, *d, *e, *f;

y = 2;
x = y;
d = "abc";
c = d;
e = "xyz";
f = "xyz";

x++;
*c = 'A';
```

- Show the state of the memory after the first set of assignments.
- Would anything be different if we had done `c = "abc"; d = c;`?
- Show the state of memory after `x++;`.
- What happens when `*c = 'A';` is executed? Why?

Admin ℹ️

- Slides and code will be emailed to you
- Please fill in the feedback form!
- Lab is in Flute (Ainsworth level 3)