COMP1511 Tutorial 3

while loops| structs | enums

What are the 4 main components of a while loop?

- Initialisation - initialising the variable that controls the loop
- Condition - stating the condition that must be true for the while loop to execute its code
- Incrementation - updating the variable that controls the loop
- Body - what happens each time the while loop runs?

A

```
void a(void) {
    int i = 5;
    while (i > 0) {
        printf("%d\n", i);
        i--;
    }
}
```

B

```
void b(void) {
    int i = 1;
    while (i < 32) {
        printf("%d\n", i);
        i = i + i;
    }
}
```

C

```
void c(void) {
    int i = 0;
    while (i < 32) {
        printf("%d\n", i);
        i = i + 2;
    }
}
```

D

```
void d(void) {
    int i = 5;
    while (i >= 0) {
        printf("%d\n", i);
        i--;
    }
}
```

E

```
void e(void) {
    int i = 0;
    int keep_going = 1;
    while (keep_going == 1) {
        if (i > 3) {
            keep_going = 0;
        }
        i++;
    }
    printf("%d\n", i);
}
```

F

```
void f(void) {
    int i;
    while (i > 0) {
        printf("%d\n", i);
        i--;
    }

}
```

G

```
void g(void) {
    int i = 0;
    int max = 32;
    while (i < max) {
        printf("%d\n", i);
        max = max + 2;
    }

}
```

H

```
void h(void) {
    int i = 0;
    int keep_going = 0;
    while (keep_going == 1) {
        if (i > 3) {
            keep_going = 0;
        }
        i++;
    }
    printf("%d\n", i);
}
```

2D While Loops

- Nesting a while loop inside another while loop gives a 2D while loop
- Each time the outer while loop repeats, the inner while loop runs its entire cycle
- Useful of printing out rows and columns of characters, as well as looping through a 2D grid of objects

2D While Loops

```
void a(void) {
    int row = 0;
    while (row < SIZE) {
        int col = 0;
        while (col < SIZE) {
            if (row == col) {
                printf("O");
            } else {
                printf("X");

            }
            col++;
        }
        row++;
        printf("\n");
    }
}
```

```
void b(void) {
    int row = 0;
    while (row < SIZE) {
        int col = 0;
        while (col < SIZE) {
            if (col % 2 == 0) {
                printf("O");
            } else {
                printf("X");

            }
            col++;
        }
        row++;
        printf("\n");
    }
}
```

```
void c(void) {
    int row = 0;
    while (row < SIZE) {
        int col = 0;
        while (col < SIZE) {
            if (col != 1 && row != 1)
{
                printf("O");
            } else {
                printf("X");

            }
            col++;
        }
        row++;
        printf("\n");
    }
}
```

```
void d(void) {
    int row = 0;
    while (row < SIZE) {
        printf("X");
        int col = 1;
        while (col < 3) {
            if (row == 0 || row == 3)
{
                printf("X");
            } else {
                printf("O");
            }
            col++;
        }
        printf("X");
        row++;
        printf("\n");
    }
}
```

Enums and Structs

Enum: Custom data type that specifies a set of possible values for a variable to take (similar to having multiple #defines

Struct: Custom data type that groups together different (usually related) variables

Variable names

Legal C variable names:

- Can only contain letters, numbers and _
- Must not start with a number

Following the style guide, variable names must:

- Start with a lowercase letter
- Use snake_case (#defines should use SHOUTING_SNAKE_CASE)

Variable names should also be descriptive and relevant - for the humans who have to read them!