Please take a minute to fill out the form for last week if you haven't already (or do attendance for today) 🙂

# Week 3!

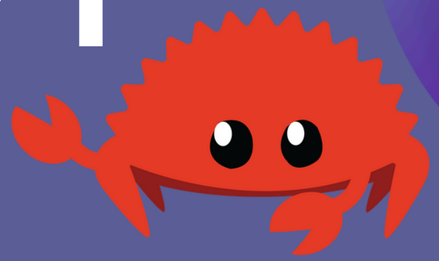
COMP1511 24T3

## Solving Modern Programming Problems with Rust

COMP6991 | 6 Units of Credit

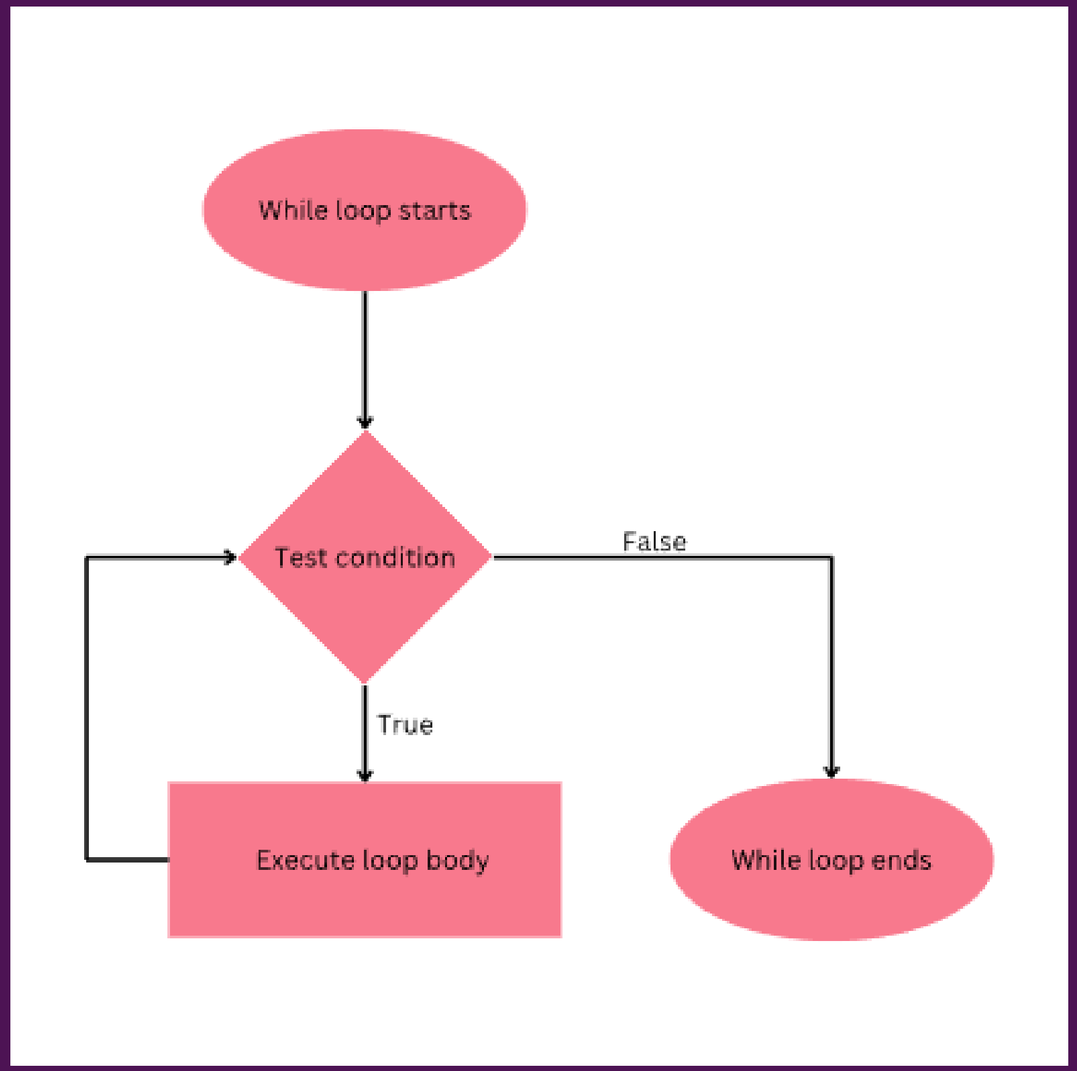
You are viewing the 2024 version | 2024

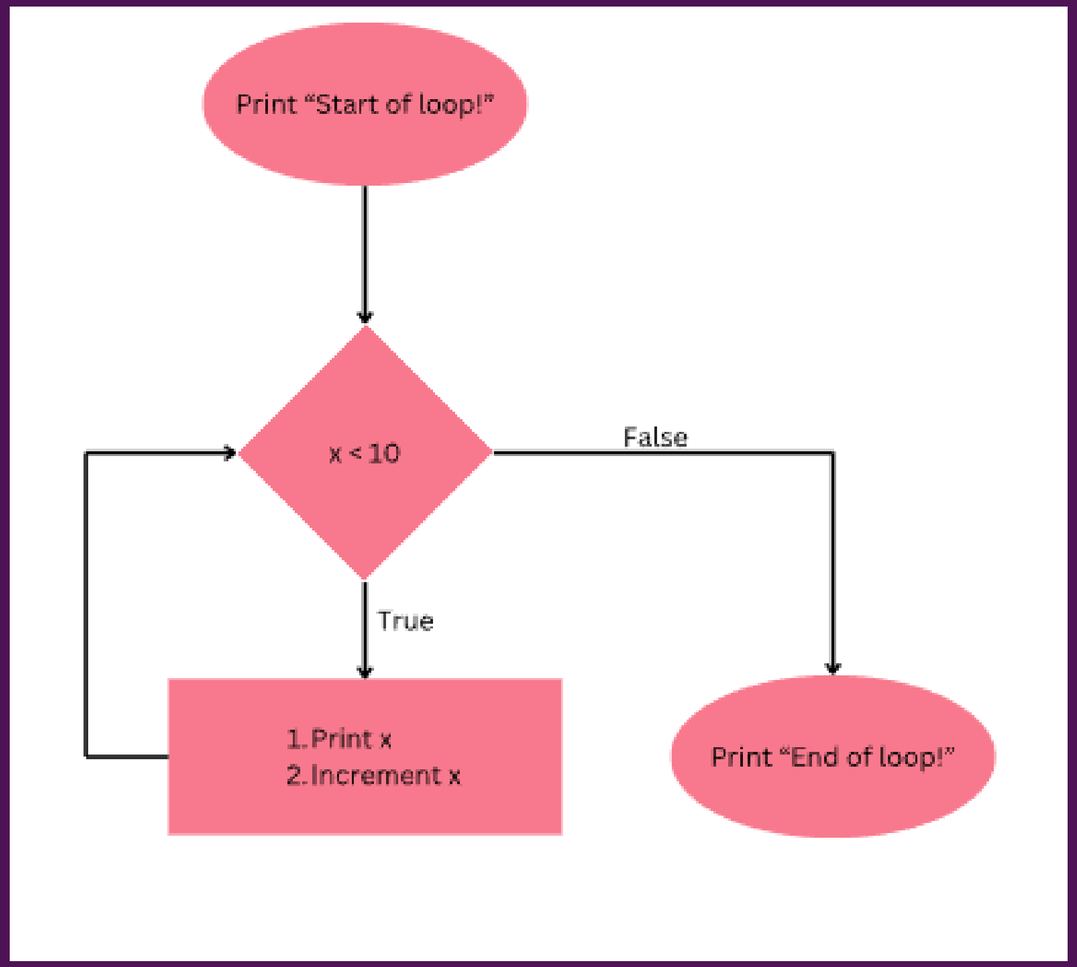
# COMP6991



# Overview

While Loops
2D While Loops
Scanning in Loops
Structs and Enums


While loop starts
Test condition
False
True
Execute loop body
While loop ends


Print “Start of loop!”
x < 10
False
True
1. Print x
2. Increment x
Print “End of loop!”

# While Loops

In groups hand execute your two assigned while loops

If you are struggling, it might help to write down what values each variable is and change them as you iterate

A

```c
#include <stdio.h>

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

B

```c
#include <stdio.h>

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

C

```c
#include <stdio.h>

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

D

```c
#include <stdio.h>

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

E

```c
#include <stdio.h>

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

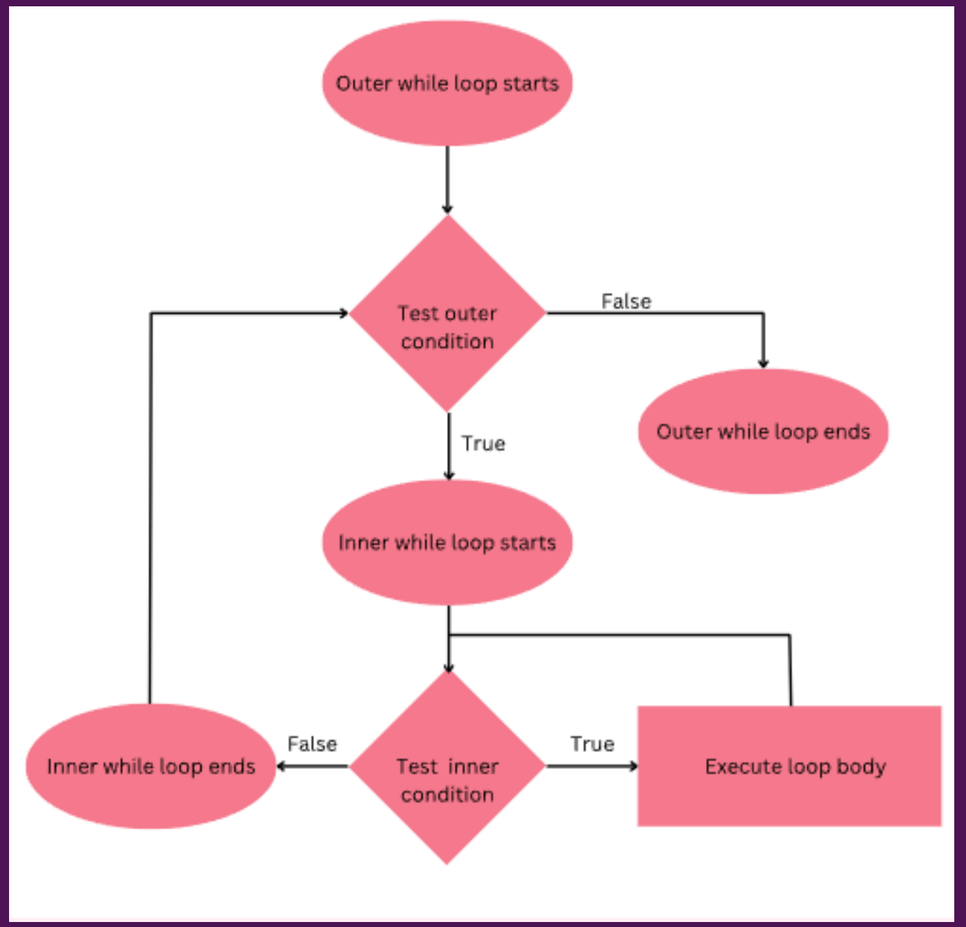
F

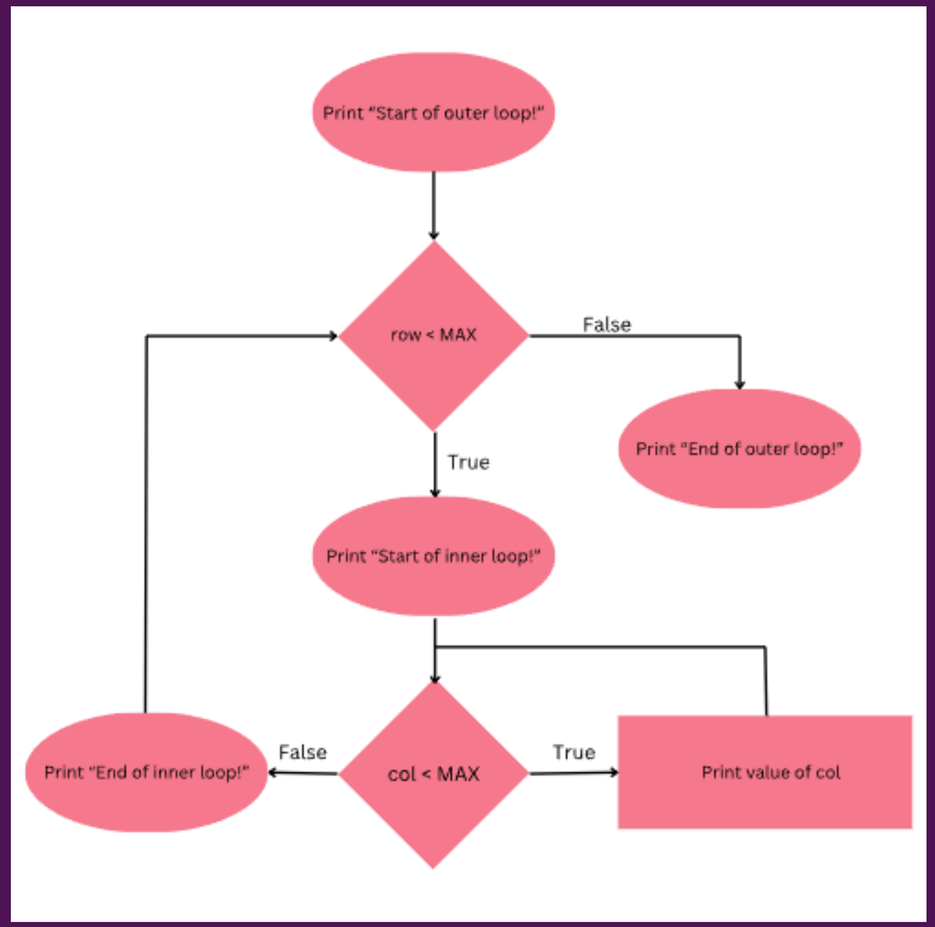
```c
#include <stdio.h>

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

# 2D While Loops

Exactly the same as regular while loops

Nesting means the inner loop happens many times for each iteration of the outer loop


Outer while loop starts
Test outer condition
False
Outer while loop ends
True
Inner while loop starts
Inner while loop ends
False
Test inner condition
True
Execute loop body


Print "Start of outer loop!"
row < MAX
False
Print "End of outer loop!"
True
Print "Start of inner loop!"
col < MAX
False
True
Print "End of inner loop!"
Print value of col

# Your turn!

| (A) | (B) | (C) | (D) |
|---|---|---|---|
| OXXX | OXOX | OXOO | XXXX |
| XOXX | OXOX | XXXX | XOOX |
| XXOX | OXOX | OXOO | XOOX |
| XXXO | OXOX | OXOO | XXXX |

**1**

```c
#include <stdio.h>

#define SIZE 4

int main(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");
    }
    return 0;
}
```

**2**

```c
#include <stdio.h>

#define SIZE 4

int main(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");
    }
    return 0;
}
```

**3**

```c
#include <stdio.h>

#define SIZE 4

int main(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");
    }
    return 0;
}
```

**4**

```c
#include <stdio.h>

#define SIZE 4

int main(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");
    }
    return 0;
}
```

# Scanning and Loops

| | |
|---|---|
| A: Enter a series of integers until you reach a negative number. Then, stop and calculate the sum. | B: Enter numbers until the user presses 'q'. Then, display the count of numbers entered. |
| C: Scan for even numbers within a given range until end of input and display them. | D: Scan for integers keeping a cumulative sum, until the sum of entered integers reaches or exceeds the targer sum provided by the user. |

# Structs and Enums (coffee shop)

The Program:

1. Creates a struct coffee that stores the coffee type (an enum), the number of sugars and the size of a coffee
2. Creates an enum coffee_type which defines the different types of drinks that con be ordered.
3. The program will then take in a coffee order and store it in the struct.
4. The program will then determine the price of the coffee based on the users order, outputting the price to the terminal (stdout).

The Rules:

- The base price is always 4.5.
- A LARGE coffee incurs an additional charge.
- A LATTE, CAPPUCCINO or MATCHA incurs an additional charge.
- Every sugar added incurs an additional charge.

# Lab Time!