C PROGRAMMING

By Lungsom Lamnio

Control Statements

Definition: Control statements are used to dictate the flow of execution in a program based on specific conditions.

Types:

1. **Conditional Statements**:
    - **if Statement**: Executes a block of code if a specified condition is true.
    - **if-else Statement**: Provides an alternative block of code to execute if the condition is false.
    - **else if Statement**: Used for checking multiple conditions in sequence.
    - **switch Statement**: Allows multi-way branching based on the value of a variable.

2. **Looping Statements:**
    - **for Loop**: Executes a block of code a specific number of times.
    - **while Loop**: Continues executing a block of code as long as a condition is true.
    - **do-while Loop**: Executes a block of code at least once before checking the condition.

Jump Statements

Definition: Jump statements allow the control flow to be transferred to a different part of the program, making it possible to bypass or repeat certain sections of code.

Types:

1. **goto Statement:**
   - Transfers control to a labeled statement within the same function, allowing for non-linear code execution.
2. **break Statement:**
   - Exits from the nearest enclosing loop or switch statement, terminating its execution prematurely.

3. **continue Statement:**
    - Skips the current iteration of a loop and proceeds to the next iteration, allowing the loop to continue running.
4. **return Statement:**
    - Exits from a function and optionally returns a value to the caller, terminating the function's execution.

1. Goto Statement

Notes:

- The `goto` statement is used to jump to a labeled statement in the same function.
- It can be useful in certain scenarios but can lead to less readable code.

Syntax:

c Copy code

```c
goto label;
label:
    // code to execute
```

Copy code

```c
#include <stdio.h>

int main() {
    int number;

    input:  // Label for input
    printf("Enter a positive number (0 to exit): ");
    scanf("%d", &number);

    if (number < 0) {
        goto input;  // Jump back to input label
    }
    if (number == 0) {
        goto end;  // Jump to end label
    }

    printf("You entered: %d\n", number);
    goto input;  // Repeat input

    end:
    printf("Exiting the program.\n");
    return 0;
}
```

2. Continue Statement

Notes:

- The `continue` statement skips the current iteration of a loop and proceeds to the next iteration.

Syntax:

c Copy code

```c
continue;  // Skips to the next iteration of the loop
```

Example:

c

Copy code

```c
#include <stdio.h>

int main() {
    for (int i = 1; i <= 10; i++) {
        if (i % 2 == 0) {
            continue;  // Skip even numbers
        }
        printf("%d\n", i);  // Print odd numbers
    }
    return 0;
}
```

3. Break Statement

Notes:

- The `break` statement is used to exit a loop or switch statement prematurely.

Syntax:

c Copy code

```c
break;  // Exits the nearest enclosing loop or switch
```

Example:

c

Copy code

```c
#include <stdio.h>

int main() {
    for (int i = 1; i <= 10; i++) {
        if (i == 5) {
            break;  // Exit loop when i is 5
        }
        printf("%d\n", i);  // Prints numbers 1 to 4
    }
    return 0;
}
```

4. Switch Statement

Notes:

- The `switch` statement allows multi-way branching based on the value of a variable.

Syntax:

c Copy code

```c
switch (variable) {
    case value1:
        // code for value1
        break;
    case value2:
        // code for value2
        break;
    default:
        // code if no case matches
}
```

Copy code

```c
#include <stdio.h>

int main() {
    int day;
    printf("Enter day number (1-7): ");
    scanf("%d", &day);

    switch (day) {
        case 1: printf("Monday\n"); break;
        case 2: printf("Tuesday\n"); break;
        case 3: printf("Wednesday\n"); break;
        case 4: printf("Thursday\n"); break;
        case 5: printf("Friday\n"); break;
        case 6: printf("Saturday\n"); break;
        case 7: printf("Sunday\n"); break;
        default: printf("Invalid day\n");
    }
    return 0;
}
```

6. Do-While Loop

Notes:

- A `do-while` loop executes a block of code at least once and then checks the condition.

Syntax:

c Copy code

```c
do {
    // code to execute
} while (condition);
```

Example:

c

Copy code

```c
#include <stdio.h>

int main() {
    int i = 1;
    do {
        printf("%d\n", i);  // Prints the value of i
        i++;
    } while (i <= 5);  // Loop continues until i is greater than 5
    return 0;
}
```

```c
#include <stdio.h>

int main() {
    int choice;

    do {
        printf("Menu:\n");
        printf("1. Option 1\n");
        printf("2. Option 2\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("You selected Option 1\n");
                break;
            case 2:
                printf("You selected Option 2\n");
                break;
            case 3:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice, try again.\n");
        }

        if (choice == 3) {
            break;  // Exit the do-while loop
        }

    } while (1);  // Infinite loop until break

    return 0;
}
```

Assignment Questions

1. Write a program that uses goto to skip negative numbers and print only non-negative numbers entered by the user
2. Write a program that prints numbers from 1 to 20, skipping multiples of 3.
3. Write a program that uses break to exit a loop when a user enters a specific number.
4. Write a program using switch to create a simple calculator that performs addition, subtraction, multiplication, and division based on user input.

Steps to submit the assignments

Step 1: Make a GitHub repo with the name as YourMentor.
Step 2: Inside the repo YourMentor, make a folder with the name as C_BootCamp.
Step 3: Submit the assignment with the day number (eg: Day1_Assignment.pdf).
Step 4: DM me the link of repo.

NOTE:

1. Active members will get the chance to make a real-life project with me.
2. Advantages with Upcoming BootCamps.

THANK YOU