COMP0204: Introduction to Programming for Robotics and AI

Control Flow

Course lead: Dr Sophia Bano

Lecture by: Dr Yunda Yan

MEng Robotics and AI

UCL Computer Science

Blocks

- A block (also known as a compound statement) zero or more statements enclosed in curly braces {}.
- **Syntax:**

```
{
    // Statement 1
    // Statement 2
    // ...
    // Statement N
}
```

```c
int number = 10;

// Single control flow block with an 'if' statement
if (number > 5) {
    printf("The number is greater than 5.\n");
}

// Rest of the program
printf("This is outside the 'if' block.\n");
```

Blocks

- Block can be empty {}
- Compiled as a single unit
- Variables can be declared inside
- Blocks can be nested

```c
// An empty block
{
    // This block does nothing
}


// A single-statement block
{
    x = x * 2;
}
```

```c
#include <stdio.h>

int main() {
    int x = 5;

    // Outer block
    {
        int y = 10;
        printf("Inside the outer block:\n");
        printf("x = %d\ny = %d\n", x,y);

        // Inner block
        {
            int z = 15;
            printf("\nInside the inner block:\n");
            printf("x = %d\ny = %d\nz = %d\n", x,y,z);
        }

        // The 'z' variable is not accessible here
        printf("\nBack inside the outer block:\n");
        printf("x = %d\ny = %d\n", y);
        // 'z' is still not accessible here
    }

    // The 'y' and 'z' variables are not accessible here
    printf("\nOutside both blocks:\n");
    printf("x = %d\n", x);
    // 'y' and 'z' are not accessible here

    return 0;
}
```

Control conditions

- No Boolean in C (0 is False, 1 is True)
- Condition is an expression (or a series of expressions)

  x > 10 or x < y
- Expression is non-zero → condition is true

Control Instructions

- Used to determine flow of program
  a. Sequence control – instructions run in sequence
  b. Decision control – if- else conditional statements
  c. Case control – switch - do separate things given a case
  d. Loop control – for, while loops – to do a task repeatedly

if Conditional Statement

Syntax

```
if (condition)
x=<expression_if_true>;
else
x=<expression_if_false>;
```

```
condition ? expression_if_true :
expression_if_false;
```

C provides syntactic sugar to express the same using the

ternary operator ’?:’

The ternary operator makes the code shorter and

easier to understand (syntactic sugar).

if Conditional Statement

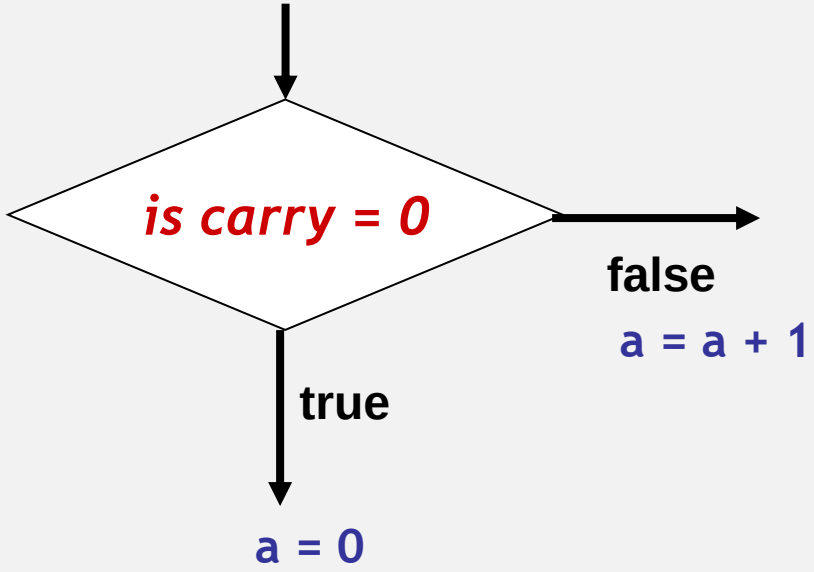
Example

if the value of carry is 0 then

set the value of a to 0

else

set the vale of a to a+1



if Conditional Statement - types

1. Basic ‘if’ Statement

```c
int x = 10;

if (x > 5) {
    printf("x is greater than 5.\n");
}
```

2. ‘if-else’ Statement

```c
int x = 3;

if (x > 5) {
    printf("x is greater than 5.\n");
} else {
    printf("x is not greater than 5.\n");
}
```

if Conditional Statement - types

3. ‘if-else if-else’ Statement (multiple conditions)

```c
int x = 7;

if (x > 10) {
    printf("x is greater than 10.\n");
} else if (x > 5) {
    printf("x is greater than 5 but not greater than 10.\n");
} else {
    printf("x is not greater than 5.\n");
}
```

if Conditional Statement - types

4. Nested ‘if’ statements:

```c
int x = 10;
int y = 5;

if (x > 5) {
    if (y > 2) {
        printf("Both x and y are greater than their respective thresholds.\n");
    } else {
        printf("x is greater than 5, but y is not greater than 2.\n");
    }
} else {
    printf("x is not greater than 5.\n");
}
```

5. Ternary Operator (‘? :’):

```c
int x = 7;
int result = (x > 5) ? 100 : 200;
printf("Result: %d\n", result);
```

if Conditional Statement

Example:

```c
#include <stdio.h>

int main() {
    int x = 10;
    int y = 20;
    int max;

    // standard syntax for condition expression
    if (x > y) {
        max = x;
    } else {
        max = y;
    }

    // syntatic sugar for condition expression
    //max = (x > y) ? x : y;

    printf("The maximum is: %d\n", max);

    return 0;
}
```

if Conditional Statement

Exercise

Write a C program that takes a year as input and determines whether it is a leap year or not. A leap year is defined as follows:

- If the year is evenly divisible by 4, it is a leap year.
- However, if the year is evenly divisible by 100, it is not a leap year.
- But, if the year is evenly divisible by 400, it is still a leap year.

switch Conditional Statement

- Used when we have multiple possible execution paths based on the value of a single expression
- Select one of many code blocks to be executed based on the value of an expression

Syntax

```
switch (expression) {
    case constant1:
        // Code to execute if expression equals constant1
        break;
    case constant2:
        // Code to execute if expression equals constant2
        break;
    // ...
    default:
        // Code to execute if expression doesn't match any case
}
```

switch Conditional Statement

Execution Flow:

- The **switch** expression is evaluated.
- The program looks for a **case** label whose **constant matches the expression value**.
- The code block associated with the matching case label is executed.
- Execution continues until a **break** statement is encountered.
- Execution falls through if **break** is not included

default Case:

- If **no case** matches the expression, the code inside the **default block** is executed.

```c
#include <stdio.h>

int main()
{
    int day = 3;

    switch (day) {
        case 1:
            printf("Monday\n");
            break;
        case 2:
            printf("Tuesday\n");
            break;
        case 3:
            printf("Wednesday\n");
            break;
        default:
            printf("Invalid day\n");
    }
}
```

switch Conditional Statement

Exercise

Write a C program that takes a numerical grade as input (0-100) and calculates the corresponding letter grade based on the following grading scale:

- A: 90-100
- B: 80-89
- C: 70-79
- D: 60-69
- F: 0-59

Use a **switch** statement to determine and display the letter grade for the input grade.

Loop statements

- while loop
- for loop
- do while loop
- break and continue keywords

while Loop

Syntax

```
while (condition) {
    /* loop body */
}
```

```c
#include<stdio.h>
int main()
{
    int count = 1;
    while (count <= 5) {
        printf("Iteration %d\n", count);
        count++;
    }
}

```

- Pre-test loop
- Simplest loop structure – evaluate body as long as condition is true
- Condition evaluated first, so body may never be executed

Important:

- Ensure the loop condition eventually becomes false to avoid infinite loops, which can crash your program

for Loop

Syntax

```
for (initialization; condition; iteration) {
    /* loop body */
}
```

- Counter control loop

Initialization: Setting the initial value of the loop variable. i=1

Condition: Defining the condition for continuing the loop. i<=10

Iteration: Modifying the loop variable at the end of each iteration. i++

for Loop - Examples

Single for loop

```c
#include<stdio.h>
int main()
{
    for (int i = 1; i <= 5; i++)
    {
    printf("Iteration %d\n", i);
    }
}
```

Nested for loop: Multiplication table

```c
#include<stdio.h>

int main() {
    int rows = 5; // Number of rows for the multiplication table

    // Outer loop for the rows
    for (int i = 1; i <= rows; i++) {
        // Inner loop for the columns
        for (int j = 1; j <= rows; j++) {
            printf("%d x %d = %d\t", i, j, i * j);
        }
        // Move to the next line after each row
        printf("\n");
    }

    return 0;
}
```

do while Loop

Syntax

```
do {
    /* loop body */
    } while (condition);
```

```c
#include<stdio.h>

int main()
{
    int count = 1;
    do {
        printf("Iteration %d\n", count);
        count++;
    } while (count <= 5);
    return 0;
}
```

- Post-test loop
- Differs from while loop – condition evaluated after each iteration
- Body executed at least once
- Note semicolon at end

Important:

- Ensure the loop condition eventually becomes false to avoid infinite loops, which can crash your program

While(1) loop and break keyword

Syntax

```
while (1) {
    /* loop body continue indefinitely */
}
```

- break; used sometimes to terminate a loop early
- break; exits innermost loop or switch statement to exit early

```c
#include<stdio.h>

int main()
{
    int number;
    while (1) {
        printf("Enter a number (0 to exit): ");
        scanf("%d", &number);

        if (number == 0) {
            printf("Exiting the loop...\n");
            break; // Exit the loop if the number is 0
        }

        // Perform some processing with the number
        printf("You entered: %d\n", number);
    }

}
```

continue keyword

- Used to skip an iteration
- skips rest of innermost loop body, jumping to loop condition

```c
#include <stdio.h>

int main() {
    int i = 1;

    while (i <= 5) {
        // Check if i is even
        if (i % 2 == 0) {
            printf("Skipping even number: %d\n", i);
            i++;
            continue; // Skip the rest of the loop body and continue to the next iteration
        }

        printf("Processing odd number: %d\n", i);
        i++;
    }

    return 0;
}
```

Output

```
Processing odd number: 1
Skipping even number: 2
Processing odd number: 3
Skipping even number: 4
Processing odd number: 5
```

Loops - Practice Example

Exercise

Write a C program that takes a positive integer num as input, display the sum of all even numbers from 1 to num.

Use for loop to implement

Use while loop to implement

Use do while loop to implement

Reminder:

1st Assessment – 16th Oct during the lab session