

# Conditional Execution and Loops in C

---

# Outline

- 1 Conditional Execution
- 2 Nested `if` Statements
- 3 Loop
- 4 `break` and `continue`
- 5 Nested Loops
- 6 The `switch` Statement
- 7 Exercise

# Conditional Execution

---

# Conditional Execution in C

- **Conditional execution** allows a program to take different actions based on certain conditions
- Conditions are expressed using `if`, `else`, and `else if` statements
- Condition expressions must evaluate to `true` (non-zero) or `false` (zero)

# if Statement

## Syntax:

```
if(condition){  
    // statements  
}
```

## Example:

```
1 #include <stdio.h>  
2 int main(){  
3     int x = 10;  
4     if(x > 0){  
5         printf("Positive number\n");  
6     }  
7 }
```

# if-else Statement

```
if(condition){  
    // commands to execute if true  
} else{  
    // commands to execute if false  
}
```

## Example: if-else

```
1  #include <stdio.h>
2
3  int main(){
4      int age = 18;
5
6      if(age >= 18){
7          printf("Eligible to vote\n");
8      } else{
9          printf("Not eligible to vote\n");
10     }
11 }
```

## else if Ladder

```
if(condition1){  
    ...  
} else if(condition2) {  
    ...  
} else{  
    ...  
}
```



## Example: else if Ladder

```
1  #include <stdio.h>
2
3  int main(){
4      int marks = 75;
5
6      if(marks >= 90){
7          printf("Grade A");
8      } else if(marks >= 75){
9          printf("Grade B");
10     } else{
11         printf("Grade C");
12     }
13 }
```

# Boolean Algebra in `if` Statements

- There can be multiple conditions
- Need to perform Boolean algebra on these conditions, because `if` statement expects only a single value
- Boolean operations on multiple conditions evaluate to a single value (true or false)
- Boolean operators:
  - AND (`&&`): Code runs only if all conditions are true
  - OR (`||`): Code runs if at least one condition is true
  - NOT (`!`): Negates a condition (flips true to false, and vice-versa)

## Example: Loan Eligibility (AND operator)

```
1  #include <stdio.h>
2
3  int main(){
4      if((age >= 18) && (income >= 20000)){
5          printf("Eligible for loan");
6      } else{
7          printf("Ineligible for loan");
8      }
9  }
```

## Example: Age Check (AND operator)

```
1  #include <stdio.h>
2
3  int main(){
4      if((age >= 13) && (age <= 19)){
5          printf("The user is a teenager");
6      } else{
7          printf("The user is not a teenager");
8      }
9  }
```

## Example: Sports Eligibility (OR operator)

```
1  #include <stdio.h>
2
3  int main(){
4      if((age >= 16) || (weight >= 60)){
5          printf("Eligible");
6      } else{
7          printf("Not eligible");
8      }
```

## Nested if Statements

---

## Nested if Statements

Sometimes, it necessary to put an if statement inside another. This is called nested statements. Can have as many levels of nesting as necessary.

```
if(cond1){  
    // code that gets executed if cond1 is true  
    if(cond2){  
        // executed if both cond1 and cond2 are true  
    } else{  
        // executed if both cond1 is true and cond2 is false  
    }  
    // code that gets executed if cond1 is true  
}
```

## Example: Loan Eligibility (revisited)

```
1  #include <stdio.h>
2
3  int main(){
4      if(age >= 18){
5          if(income >= 20000){
6              printf("Eligible for loan");
7          } else{
8              printf("Not eligible: income too low");
9          }
10     } else{
11         printf("Not eligible: under 18");
12     }
13 }
```



# Loop

---

# Loops in C

- Loops are used to execute a block of code repeatedly.
- Types of loops in C:
  - `for` loop: when number of iterations is known
  - `while` loop: when condition is checked before each iteration
  - `do-while` loop: condition checked *after* executing loop body

In `do-while` loop, the body of the loop is always executed at least once.

# for Loop

```
for(initialization; condition; update){  
    // statements  
}
```

The elements (initialization, condition and update) inside the for keyword, can be omitted. For example,

- Initialization can be performed before the for keyword
- Condition and update can moved inside the loop body
- `for(;;){...}` creates an infinite loop

# Example: for Loop

```
1  #include <stdio.h>
2
3  int main(){
4      for(int i = 1; i <= 5; i++){
5          printf("%d ", i);
6      }
7  }
```

## Example: Another Way to Construct for Loops

```
1  #include <stdio.h>
2
3  int main(){
4      int i = 1;
5      for(;;){
6          if(i>5){
7              break;    // exits the loop when i>5
8          }
9          printf("%d\n", i);
10         i++;
11     }
12 }
```

## Example: Sum Odd Integers (if inside for)

```
1  #include <stdio.h>
2
3  int main(){
4      int i, sum = 0;
5      for(i = 1; i <= 11; i++){
6          if (i%2 == 1){
7              sum += i;
8          }
9      }
10     printf("%d", sum);
11 }
```

## Example: Sum Odd Integers (No if statement)

```
1  #include <stdio.h>
2
3  int main(){
4      int i, sum = 0;
5      for(i = 1; i <= 11; i+=2){
6          sum += i;
7      }
8      printf("%d", sum);
9  }
```

# while Loop

## Syntax:

```
while(condition){  
    // statements  
}
```

## Example:

```
1 #include <stdio.h>  
2 int main(){  
3     int i = 1;  
4     while(i <= 5){  
5         printf("%d ", i);  
6         i++;  
7     }  
8 }
```



## Example: Greatest Common Divisor (GCD)

- The GCD of two integers  $a$  and  $b$  is  $c$  if both  $a$  and  $b$  are divisible by  $c$
- First, assume that the smaller number is the GCD
- Then check if both  $a$  and  $b$  are divisible by the assumed GCD. If not, then decrement the assumed value by 1
- Keep repeating this process until both  $a$  and  $b$  are found to be divisible

## Example: GCD (cont.)

```
1  #include <stdio.h>
2
3  int main(){
4      int a, b, gcd;
5      scanf("%d %d", &a, &b);
6
7      if(a < b){
8          gcd = a;
9      } else{
10         gcd = b;
11     }
```

*Continued in the next page*

## Example: GCD (cont.)

```
12  
13     while((a%gcd!=0) || (b%gcd!=0)){  
14         gcd--;  
15     }  
16  
17     printf("%d", gcd);  
18 }
```

# do-while Loop

## Syntax:

```
do{  
    // statements  
} while(condition);    // don't forget this semicolon
```

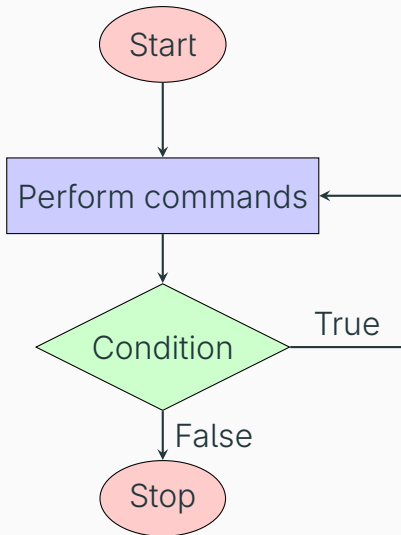
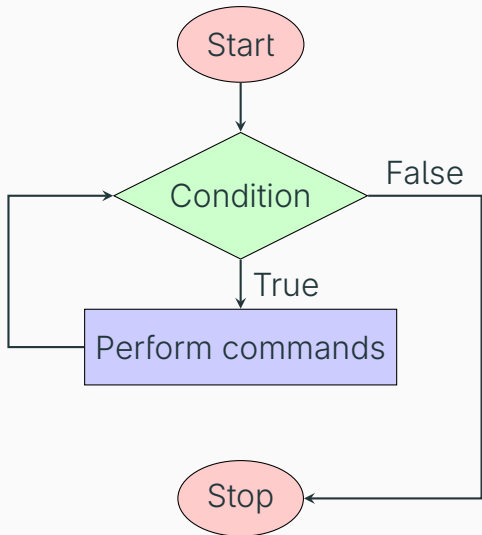
## Example:

```
1  #include <stdio.h>  
2  int main(){  
3      int i = 1;  
4      do{  
5          printf("%d ", i);  
6          i++;  
7      } while(i <= 5);  
8  }
```

# while vs do-while Loop

- `while`: condition checked *before* loop body
- `do-while`: condition checked *after* running the first iteration of the loop, so the loop runs at least once

# Flowchart: While vs Do-While



## **break and continue**

---

# The break Statement

- The break statement immediately terminates the loop or switch statement in which it is encountered
- Control of the program then transfers to the statement immediately following the loop or switch
- It is commonly used to exit a loop prematurely based on a certain condition



## Example: break

```
1  #include <stdio.h>
2
3  int main(){
4      for(int i = 1; i <= 10; i++){
5          if(i == 5){
6              break;    // Exit the loop when i is 5
7          }
8          printf("%d\n", i);
9      }
10     printf("\nLoop terminated");
11 }
```

# The continue Statement

- The continue statement skips the remaining statements in the current iteration of a loop and proceeds to the next iteration
- It is used when you want to bypass certain parts of the loop's body for specific conditions without exiting the entire loop

## Example: continue

```
1  #include <stdio.h>
2
3  int main(){
4      for(int i = 1; i <= 5; i++){
5          if (i == 3){
6              continue;    // Skip printing when i is 3
7          }
8          printf("%d\n", i);
9      }
10     printf("\nLoop finished");
11 }
```

# Nested Loops

---

# The `switch` Statement

---

# Exercise

---

# Exercise

Write C programs:

- ① To check whether a number (user input) is positive or negative or zero
- ② To check whether a year (user input) is a leap year
- ③ To check whether an integer is even or odd
- ④ To find the number of real-valued solution(s) to a quadratic equation, ( $ax^2 + bx + c = 0$ ). Take a, b and c as user inputs. Then calculate the value of the discriminant, then show the appropriate output

## Exercise (cont.)

- ⑤ To print the first  $n$  (user input) natural numbers using a `for` loop. And another program to do the same using a `while` loop
- ⑥ To compute the sum of numbers from 1 to  $n$  using a `for` loop. And another program to do the same using a `while` loop
- ⑦ To find the factorial of an integer (user input)
- ⑧ To print the first  $n$  (user input) terms of the fibonacci series
- ⑨ To print the first  $n$  (user input) terms of the following arithmetic progression sequence: 1, 4, 7, 10, 13...



## Exercise (cont.)

- 10 To repeatedly take user input and print its square, until a negative number is entered (use `while` loop)
- 11 To repeatedly take user input as exam marks and print the corresponding letter grade, until a negative number is entered (use `while` loop and `if` statement)
- 12 To find the GCD of two integers using the Euclidean algorithm
- 13 To find the LCM of two integers