**Decision Control Structures**

1. **If Statement**  
   The if statement is used to execute a block of code if a specified condition is true.  
   **Syntax**:

c

if (condition) {

// Code to execute if condition is true

}

**Example**:

int a = 5;

if (a > 0) {

printf("a is positive");

}

**Explanation**: If a > 0, it prints "a is positive."

1. **If-else Statement**  
   The if-else statement adds an alternative block of code if the condition is false.  
   **Syntax**:

if (condition) {

// Code if condition is true

} else {

// Code if condition is false

}

**Example**:

int a = -5;

if (a > 0) {

printf("a is positive");

} else {

printf("a is negative");

}

**Explanation**: If a is not greater than 0, it prints "a is negative."

1. **Nested If**  
   The nested if is used when one if statement is inside another if statement.  
   **Syntax**:

if (condition1) {

if (condition2) {

// Code if both conditions are true

}

}

**Example**:

int a = 5, b = 3;

if (a > 0) {

if (b > 0) {

printf("Both are positive");

}

}

1. **If-else Ladder**  
   This is a chain of if-else statements used when multiple conditions are checked in sequence.  
   **Syntax**:

if (condition1) {

// Code for condition1

} else if (condition2) {

// Code for condition2

} else {

// Code if no condition is true

}

**Example**:

int marks = 85;

if (marks >= 90) {

printf("Grade A");

} else if (marks >= 75) {

printf("Grade B");

} else {

printf("Grade C");

}

1. **Switch Case Statement**  
   The switch statement allows choosing between multiple cases based on the value of a variable.  
   **Syntax**:

switch (variable) {

case value1:

// Code for value1

break;

case value2:

// Code for value2

break;

default:

// Code if none of the values match

}

**Example**:

int day = 2;

switch (day) {

case 1:

printf("Monday");

break;

case 2:

printf("Tuesday");

break;

default:

printf("Other day");

}

**Explanation:** The switch case for 2 is executed, so "Tuesday" is printed.

**Iterative Statements**

1. **For Loop**  
   The for loop repeats a block of code a specific number of time

**Syntax**:

for (initialization; condition; increment) {

// Code to be repeated

}

**Example**:

for (int i = 0; i < 5; i++) {

printf("%d ", i);

}

// Output: 0 1 2 3 4

1. **While Loop**

A While loop repeats a block of code as long as a condition is true  
**Syntax**:

while (condition) {

// Code to be repeated

}

**Example**:

int i = 0;

while (i < 5) {

printf("%d ", i);

i++;

}

// Output: 0 1 2 3 4

1. **Do-while Loop**

A do-while loop runs the block of code at least once, then checks the condition.  
**Syntax**:

do {

// Code to be repeated

} while (condition);

**Example**:

int i = 0;

do {

printf("%d ", i);

i++;

} while (i < 5); //Output: 0 1 2 3 4

**Conditional Statements**

* **Break**: Exits the loop or switch case immediately.
  + **Example**:

for (int i = 0; i < 5; i++) {

if (i == 3) break; // Loop exits when i is 3

printf("%d ", i);

}

// Output: 0 1 2

* **Continue**: Skips the current iteration and moves to the next iteration of the loop.
  + **Example**:

c

Copy code

for (int i = 0; i < 5; i++) {

if (i == 2) continue; // Skips when i is 2

printf("%d ", i);

}

// Output: 0 1 3 4

**Storage Classes**

* **auto**: Default for local variables, stored in memory.
* **static**: Retains value across function calls.
* **extern**: Accesses variables from another file.
* **register**: Suggests that the variable be stored in a CPU register.

**Arrays**

* **Declaration**: Arrays are collections of similar data types stored in contiguous memory locations.
  + **Syntax**: data\_type array\_name[size];
  + **Example**: int arr[5];
* **Initialization**:

c

Copy code

int arr[5] = {1, 2, 3, 4, 5};

**Types of Arrays**

1. **Single-Dimensional Array**: A linear array.
   * **Example**: int arr[5] = {1, 2, 3, 4, 5};
2. **Two-Dimensional Array**: An array of arrays, similar to a matrix.
   * **Example**:

c

Copy code

int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};

**Address Calculation of an Element in 2-D Array**

* For a 2D array arr[m][n], the address of an element arr[i][j] can be calculated using the formula:

css

Copy code

Address of arr[i][j] = base\_address + (i \* n + j) \* size\_of\_element

* **Example**: In a 2D array arr[2][3], to calculate the address of arr[1][2] if the base address is 1000 and size\_of\_element is 4 bytes:

yaml

Copy code

Address = 1000 + (1 \* 3 + 2) \* 4 = 1020