**QUE 1 : What are conditional statements in C++? Explain the if-else and switch statements.**

**What are Conditional Statements?**

* **Conditional statements** allow your program to **make decisions**.
* They **execute different code** depending on **whether a condition is true or false**.

**if-else Statement**

* **if** checks a condition.
* If the condition is **true**, it **executes** a block of code.
* If the condition is **false**, the **else** block (optional) can run **alternative code**.

**Syntax:**

**if (condition) {**

// code if condition is true

} else {

// code if condition is false

}

**Example:**

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter a number: ";

cin >> number;

if (number > 0) {

cout << "Positive number";

} else {

cout << "Non-positive number";

}

return 0;

}

If the user inputs a number greater than 0, it prints "Positive number", otherwise "Non-positive number".

**switch Statement**

* **switch** is used when you want to **select one out of many options** based on a variable’s value.
* It's like a **cleaner alternative** to multiple if-else-if statements.

**Syntax:**

switch (expression) {

case value1:

// code for value1

break;

case value2:

// code for value2

break;

...

default:

// code if no case matches

}

* **break** stops the switch after a case is executed.
* **default** is optional and runs if no case matches.

**Example:**

#include <iostream>

using namespace std;

int main() {

int day;

cout << "Enter day number (1-3): ";

cin >> day;

switch (day) {

case 1:

cout << "Monday";

break;

case 2:

cout << "Tuesday";

break;

case 3:

cout << "Wednesday";

break;

default:

cout << "Invalid day";

}

return 0;

}

If the user enters 2, it prints "Tuesday".  
 If the user enters a wrong number like 5, it prints "Invalid day".

**Quick Comparison:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **if-else** | **switch** |
| Use for | Checking **ranges**, **complex** conditions | Checking **equality** of one variable |
| Syntax | Longer for many conditions | Neater for many options |
| Flexibility | Very flexible | Limited to discrete values (integers, characters) |

**QUE 2 : What is the difference between for, while, and do-while loops in C++?**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **for Loop** | **while Loop** | **do-while Loop** |
| Condition Check | Before each iteration | Before each iteration | After each iteration |
| Runs at least once? | No | No | Yes |
| Best use case | Known number of repetitions | Unknown repetitions | Must execute at least once |
| Structure | All in one line (init, cond, update) | Only condition given | Only condition given (after running) |

**QUE 3 : How are break and continue statements used in loops? Provide examples.**

**break Statement**

* **break** is used to **completely exit** a loop when a certain condition is met.
* After break, control **jumps out** of the loop immediately.

**Example:**

#include <iostream>

using namespace std;

int main() {

for (int i = 1; i <= 5; i++) {

if (i == 3) {

break; // Exit the loop when i is 3

}

cout << i << " ";

}

return 0;

}

**Output:**  
1 2

Loop stops completely when i == 3.

**continue Statement**

* **continue** skips **only the current iteration** and goes to the **next** loop cycle.
* It **does not stop** the whole loop.

**Example:**

#include <iostream>

using namespace std;

int main() {

for (int i = 1; i <= 5; i++) {

if (i == 3) {

continue; // Skip printing when i is 3

}

cout << i << " ";

}

return 0;

}

**Output:**  
1 2 4 5

When i == 3, it **skips printing**, but the loop **continues** with i == 4.

**QUE 4 : Explain nested control structures with an example.**

**What Are Nested Control Structures?**

* **Nested control structures** are when you put one **control statement** (like if, for, while, switch) **inside another**.
* You can **nest any type** of control structure inside another, creating multiple levels of decision-making or looping.

**Example of Nested if Statements**

You can **nest if statements** to check multiple conditions.

**Example:**

#include <iostream>

using namespace std;

int main() {

int age = 20;

char gender = 'M';

// Outer if statement

if (age >= 18) {

// Inner if statement

if (gender == 'M') {

cout << "Adult Male";

} else {

cout << "Adult Female";

}

} else {

cout << "Not an Adult";

}

return 0;

}

**Output:**

Adult Male

Here, the outer if checks if the person is an adult, and the inner if checks the gender.

**Example of Nested Loops (for inside for)**

You can **nest loops** to perform tasks that require multiple levels of iteration.

**Example:**

#include <iostream>

using namespace std;

int main() {

for (int i = 1; i <= 3; i++) {

for (int j = 1; j <= 2; j++) {

cout << "i: " << i << ", j: " << j << endl;

}

}

return 0;

}

**Output:**

i: 1, j: 1

i: 1, j: 2

i: 2, j: 1

i: 2, j: 2

i: 3, j: 1

i: 3, j: 2

The outer for loop runs 3 times, and for each iteration, the inner for loop runs 2 times.

**Key Points:**

* **Nested if** allows checking multiple conditions at different levels.
* **Nested loops** help when you need to **perform multiple iterations** on a set of data.