

**قــســــــــــم الامــــــــــــــــــــــــــن الــــــــــــــــــــــــــــــــــــسيبرانــــــــــــــــــــــــــــــــــــي**

**Department of Cyber Security**

**Subject:**

**Programming Fundamentals**

**Class:**

**1st stage**

**Lecturer:**

**Dr. Abdulkadhem A. Abdulkadhem**

**Lecture: (10)**

**Control Statements (Jump Statements)**

**1. Introduction to Control Statements**

Control statements alter the flow of execution in a program. They are essential for decision-making and looping constructs.

**Types of Control Statements:**

* Conditional Statements (if, if-else, switch)
* Looping Statements (for, while, do-while)
* Jump Statements (break, continue, goto)

**2. The break Statement**

**Definition:** The break statement is used to exit a loop or switch case prematurely.

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Usage** | **Example** |
| break; | * Inside loops (for, while, do-while)
* Within switch cases
 | for (int i = 0; i < 10; i++) { if (i == 5) { break; // Exits loop when i is 5 } cout << i << endl;} |

**3. The continue Statement**

**Definition:** The continue statement skips the current iteration and moves to the next iteration in a loop.

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Usage** | **Example** |
| continue; | • Inside loops (for, while, do-while) | for (int i = 0; i < 10; i++) { if (i == 5) { continue; // Skips iteration when i is 5 } cout << i <<endl;} |

**4. The goto Statement**

**Definition:** The goto statement transfers control to a labeled statement in the same function.

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| goto label;...label: | int i = 0;start:if (i < 5) { cout << i << endl; i++; **goto** start; // Jumps to label 'start'} |

**Risks and Best Practices:**

* Avoid excessive use as it makes code hard to read and maintain.
* Prefer structured control statements like loops and conditionals.

**5. Comparison and Use Cases**

* break vs. continue: break exits the loop, while continue skips the current iteration.
* goto is rarely used; reserved for complex control flow scenarios or breaking out of deeply nested loops.