Simple Flow Of Controls

By: Mir Faraz Ali

Programming Fundamentals

A Simple Branching Mechanism

- Definition:
- Branching allows the program to make decisions and execute different statements based on conditions.
- It is also called decision control.

- Explanation:
- Checks a condition (true/false).
- Executes one block if condition is true, another if false.

- Example:

```
• if (marks >= 50) {
•     cout << "Pass";
• } else {
•     cout << "Fail";
}
```

Compound Statements

- Definition:

A group of statements enclosed within { } that acts as a single unit.

Explanation:

- Helps when multiple statements need to be executed under a single condition.
- Improves code readability.

- Example:

```
if (x > 0) {
    y = y + 1;
    cout << "Positive number";
}
```

Simple Loop Mechanisms

Definition:

A loop repeats a block of code until a condition becomes false.

Types:

- for loop – executes a fixed number of times.
- while loop – executes as long as the condition is true.
- do…while loop – executes at least once.

- **For Loop:**

```
#include <iostream>
using namespace std;

int main() {
    cout << "For Loop Example:" << endl;
    for (int i = 1; i <= 5; i++) {
        cout << i << " ";
    }
    return 0;
}
```

- **While Loop:**

```
#include <iostream>
using namespace std;

int main() {
    cout << "\nWhile Loop Example:" << endl;
    int i = 1;
    while (i <= 5) {
        cout << i << " ";
        i++;
    }
    return 0;
}
```

- **Do While Loop:**

```
#include <iostream>
using namespace std;

int main() {
    cout << "\nDo While Loop Example:" << endl;
    int i = 1;
    do {
        cout << i << " ";
        i++;
    } while (i <= 5);
    return 0;
}
```

Increment And Decrement Operators

Definition:

Increment (++) and Decrement (--) are unary operators used to increase or decrease the value of a variable by 1.

Types:

1. Pre-Increment / Pre-Decrement:

- First increases/decreases the value, then uses it.’

2-Post-Increment / Post-Decrement:

- First uses the value, then increases/decreases it.

Example:

```
int x = 5;
int y = ++x;   // x = 6, y = 6
```

Example:

```
int x = 5;
int y = x++;   // y = 5, x = 6
```

Example:

```
#include <iostream>
using namespace std;
int main() {
    int x = 5;

    cout << "Initial value of x: " << x << endl;
    cout << "Pre-increment (++x): " << ++x << endl;  // x=6, prints 6
    cout << "Post-increment (x++): " << x++ << endl; // prints 6, then x=7
    cout << "Pre-decrement (--x): " << --x << endl;  // x=6, prints 6
    cout << "Post-decrement (x--): " << x-- << endl; // prints 6, then x=5

    cout << "Final value of x: " << x << endl;
    return 0;
}
```

Program Style – Indenting

- Definition:
- Indentation means proper alignment of code with spaces or tabs.
- Importance:
- Improves readability.
- Helps identify blocks of code easily.

- Example:

- if (age > 18) {
- cout << "Eligible to vote";

}

Program Style – Comments

- Definition:
- Comments are non-executable notes written in a program.
- Types:
- Single-line comment: // text
- Multi-line comment: /* text */

- Example:

// This program checks age eligibility

Program Style – Naming Constants

- Definition:
- Constants are fixed values that do not change during program execution.
- Best Practice: Use meaningful names with const keyword.

- Example:

```
const float PI
= 3.14159;
cout << PI * r * r;
// Circle area
```

Using Boolean Expressions

Definition:

A Boolean expression is a logical statement that results in either true or false.

Operators Used:

- && → Logical AND
- || → Logical OR
- ! → Logical NOT

- **Example:**

```
#include <iostream>
using namespace std;

int main() {
int age = 20;
int marks = 65;
if (age >= 18 && marks >= 50) {
cout << "Eligible for admission";
} else {
cout << "Not eligible";
}
return 0;
}
```

- **Example 2 – Using OR (||):**

```
int temperature = 35;
if (temperature < 0 || temperature > 30)
{
    cout << "Extreme weather!";
} else {
    cout << "Normal weather.";
}
```

- **Example 3 – Using NOT (!):**

```
bool isRainy = false;


if (!isRainy) {
    cout << "You can go
outside without umbrella";
}
```

Enumeration Type (enum):

Definition:

Enumeration (enum) is a user-defined data type that assigns names to integer constants.

Explanation:

- Makes programs easier to read.
- Values are automatically assigned starting from 0.

Example:

```
enum Days { Mon, Tue, Wed, Thu, Fri };
Days today = Wed;
cout << today;   // Output: 2
```

End Of Class