Intermediate programming(C++)-

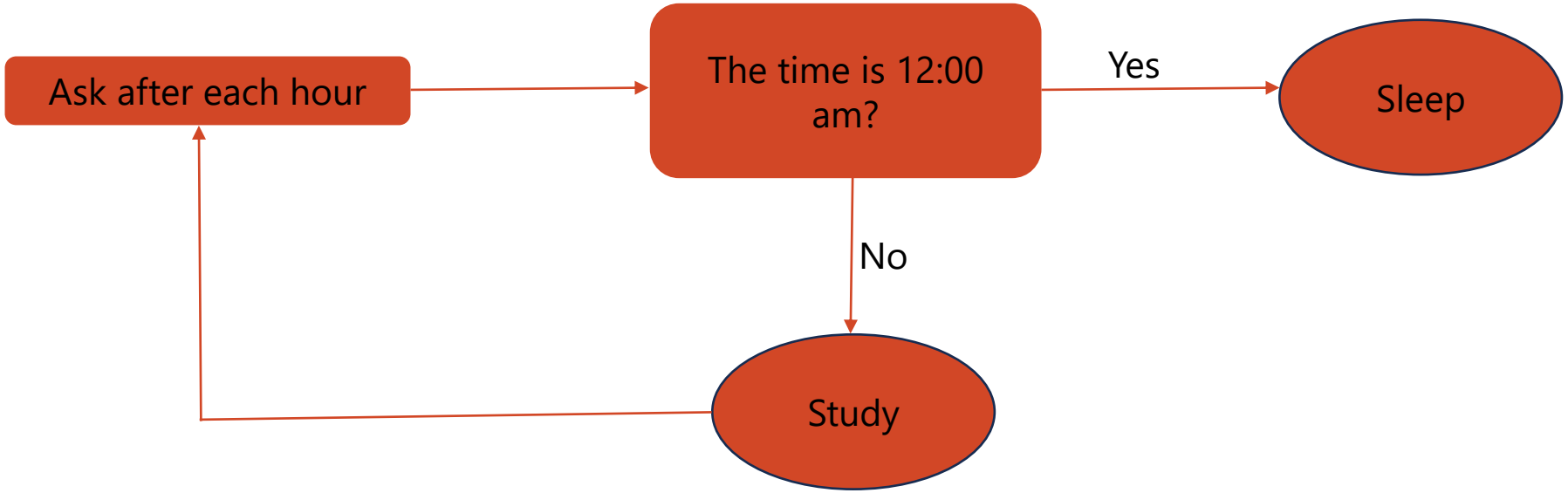
Lab 3 - Loops

Content

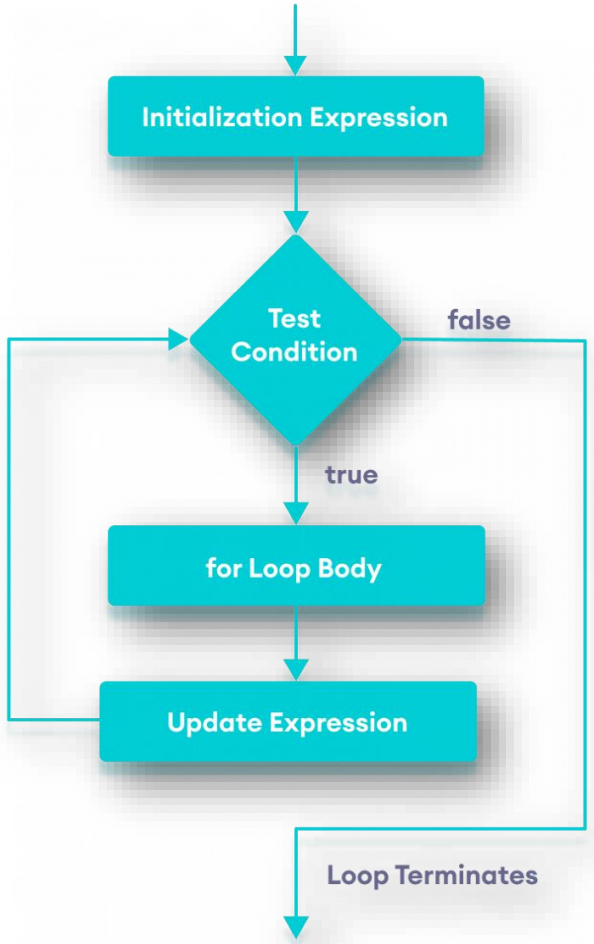
- For loop
- While loop
- Do-While loop
- Posttest and pretest.
- Nested loops
- Infinite loop
- Break statement
- Continue statement

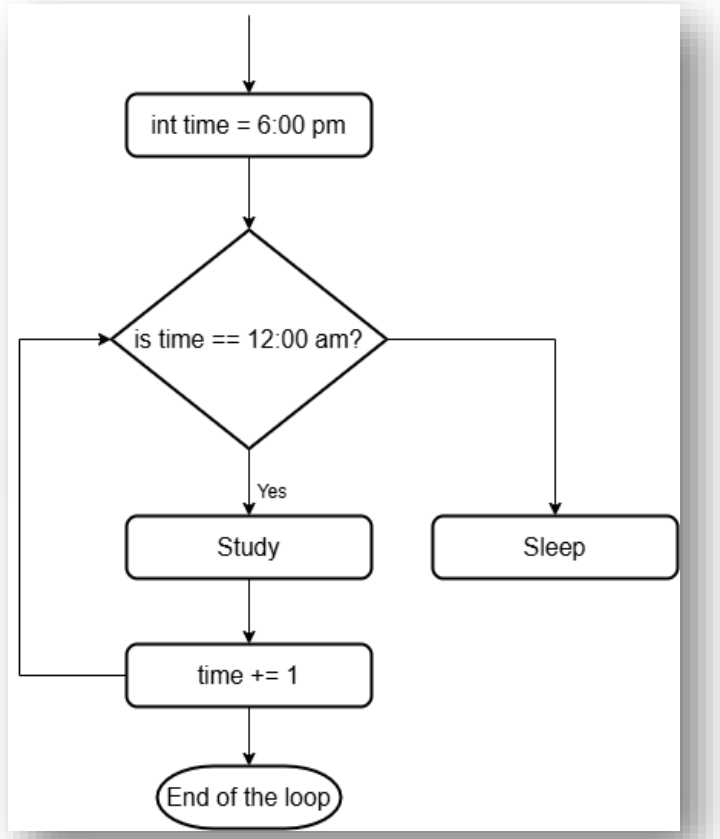
Loops

Iterate over a specific block of code while a certain condition is true, and break once the condition becomes false.



For loops





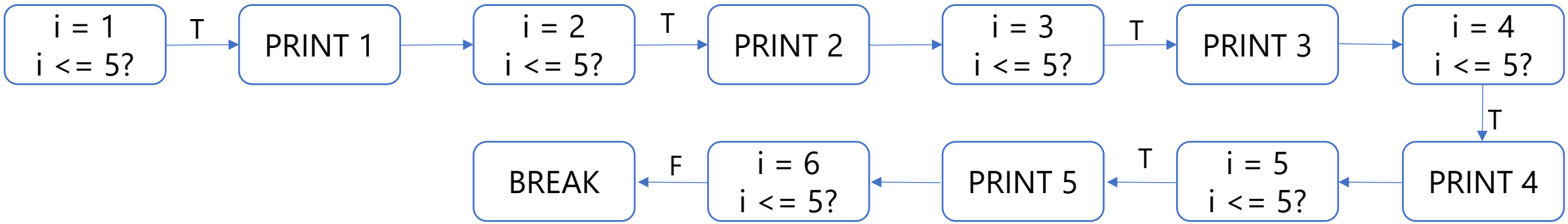

```
for (initialization; condition; update) {
   // body of-loop
}
```

```
for (int time = 6; time <= 12; time++) {
   // body of-loop
}
```

For loop - Example

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

int main() {
        for (int i = 1; i <= 5; ++i) {
        cout << i << " ";
    }
    return 0;
}
```

Output:
1 2 3 4 5



For loop – Example 2

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

int main() {
    int num, sum;
    sum = 0;

    cout << "Enter a positive integer: ";
    cin >> num;

    for (int i = 1; i <= num; ++i)
        sum += i;

    cout << "Sum = " << sum << endl;
    return 0;
}
```

Enter a positive integer: 5
Sum = 15

Output?

While loop – Example 1

```cpp
// A program to print numbers from 1 to num
#include <iostream>
using namespace std;

int main() {
    int num;
    cin >> num;

    int i = 1;
    while(i <= num) {
        cout << i << " ";
    }
    return 0;
}
```

```
initialization
while (condition) {
   // body of-loop; Update;
}
```

Output:
1 2 3 4 5

While loop – Example 2

```cpp
// A program to find the sum of natural numbers
from 1 to specific num
#include <iostream>
using namespace std;

int main() {
    int num, sum;
    sum = 0;
    cout << "Enter a positive integer: ";
    cin >> num;

    int i = 1;
    while (i <= num) {
        sum += i;
        i++;
    }
    cout << "Sum = " << sum << endl;
    return 0;
}
```

Enter a positive integer: 5
Sum = 15

Output?

Do..while loop – Example 1

```cpp
// A program to print numbers from 1 to num
#include <iostream>
using namespace std;

int main() {
    int num;
    cin >> num;

    int i = 1;
    do{
        cout << i << " ";
        i++;
    } while(i <= num);
    return 0;
}
```

```
Initialization
Do {
   // body of-loop;
   // Update;
} while (condition);
```

Output:
1 2 3 4 5

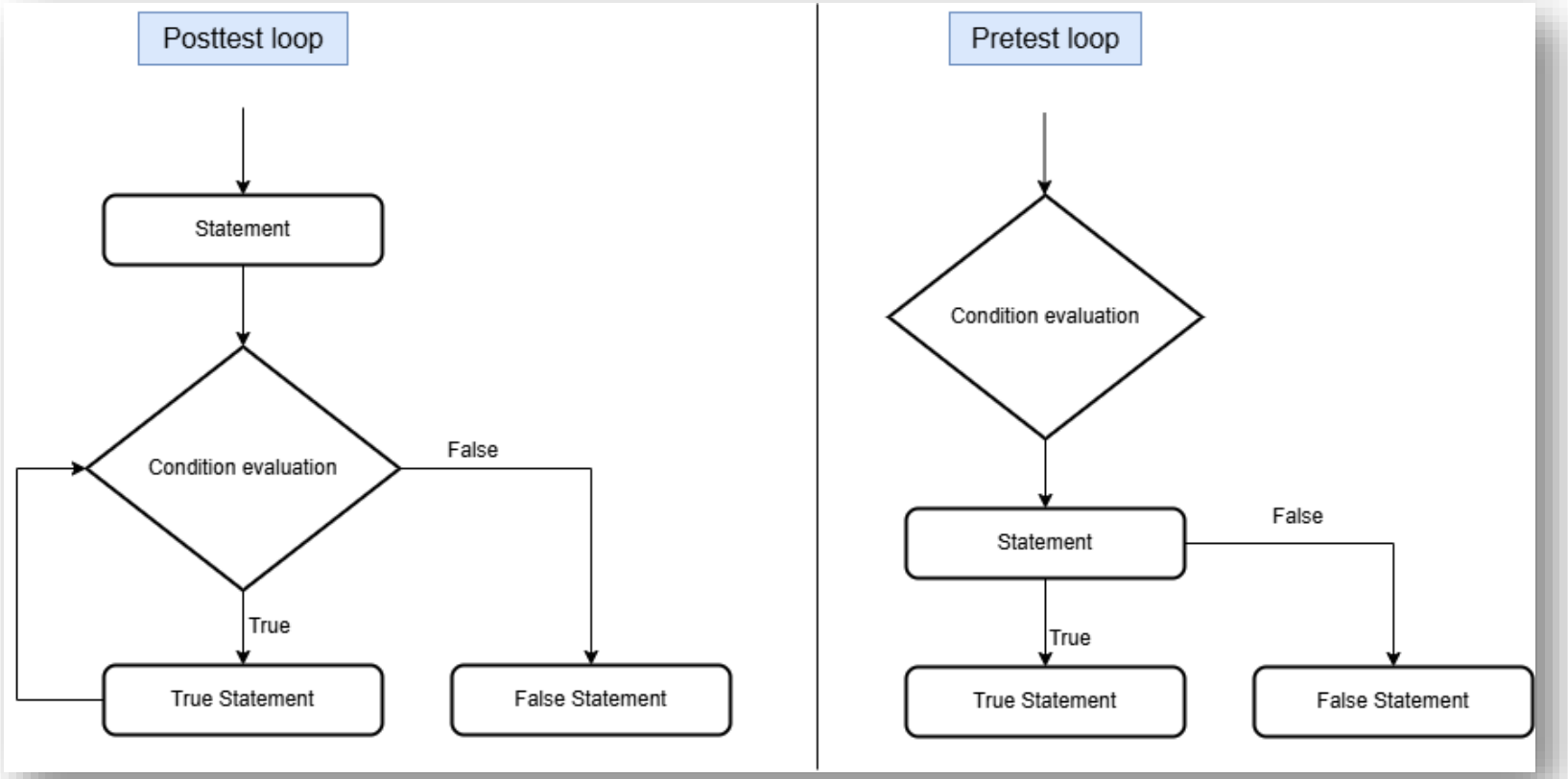
Do...while loop – Example 2

```cpp
// A program to find the sum of natural
numbers from 1 to specific num
#include <iostream>
using namespace std;

int main() {
    int num, sum = 0;
    cin >> num;

    int i = 1;
    do {
        sum += i;
        i++;
    } while(i <= num);

    return 0;
}
```

Enter a positive integer: 5
Sum = 15

Output?

Posttest vs. pretest



Ex.: Do..While loop

Ex.: While loop

Nested loops

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

int main() {
    int sum = 0;

    for (int i = 1; i <= 5; i++) {
        for (int j = 1; j <= 5; j++) {
            cout << "*";
        }
        cout << endl;
    }

    return 0;
}
```

Output:

*****
*****
*****
*****
*****

Tracing:

I = 1; I <= 5? True
J = 1; j <= 5? True
Output: *
J++;

J = 2; j <= 5? True
Output: **
J++;

J = 3; j <= 5? True
Output: ***
J++;

J = 4; j <= 5? True
Output: ****
J++;

J = 5; j <= 5? True
Output: *****
J++;

J = 6; j <= 5? False
New line

I++; I = 2; I <= 5? True
I++;

J = 1; j <= 5? True
Output: *****
*
J++;

J = 2; j <= 5? True
Output: ****
**
J++;
Etc....

Infinite loop

An infinite loop occurs when a loop continues to execute without an end condition or with a condition that never becomes false.

How it can be created?

It has 2 forms:

- while (true) { } / while(1) { }
- for (; ;)

Disadvantage: consume both memory and CPU while processing, in addiiton to program crashes.

How to avoid it?

- Ensure loop conditions will eventually be false. [e.g. int num = 3; while(num < 20) { }]
- Update the loop control variable inside the loop. [e.g. while(num < 20) { ……..; num++;}]
- Use break statements to escape loops when necessary. [e.g. while(true) { if(……..) break; }]

Break statement

- The break statement is used to exit or terminate a loop prematurely, regardless of the loop condition.
- It is often used to break out of an iteration when a certain condition is met.
- When the break statement is encountered, the program control moves to the next statement after the loop.
- It can be used with for, while and do-while loops.

Example:

```
// A program to find the first multiple of 7 greater than 100:
#include <iostream>
using namespace std;
int main() {
    int num = 101;
    while (1) {
        if (num % 7 == 0)
            break;
        num++;
    }

    cout << "First multiple of 7 greater than 100: " << num << endl;
    // for(int num = 101; num % 7 == 0; num++) {}
    return 0;
}
```

Output:
First multiple of 7 greater than 100: 105

Continue statement

- The continue statement is used to skip the rest of the current iteration and move to the next iteration of a loop.
- It is often used to skip certain iterations based on a specific condition.
- When the continue statement is encountered, the program control jumps to the loop condition check.
- It can be used with for, while and do-while loops.

Example:

```
// A program to print odd numbers between 1 and 10:
#include <iostream>
using namespace std;

int main() {
    int i = 1;
    while (i <= 10) {
        if (i % 2 == 0) {
            i++;
            continue;
        }
        cout << i;
        i++;
    }
    return 0;
}
```

Output:
1 3 5 7 9

Lab tasks

- ***Task 1:***

Write a program that asks the user to enter 5 numbers. The program should print the first negative number entered.

Input:

Enter number 1: **10**

Enter number 2: **3**

Enter number 3: **-5**

Output: First negative number: -5

- ***Task 2:***

Write a program that prints the numbers from 1 to 10 but skips the multiples of 3.

Output: 1 2 4 5 7 8 10