

Tishk International University
Cybersecurity Department
Course Code: CBS 113



Programming Fundamentals

Lecture 6



Do-While & Nested Loops

Fall 2024

Hemin Ibrahim, PhD
hemin.ibrahim@tiu.edu.iq



Outline



- The do-while loop
- While vs do-while
- Sentinels
- Nested Loops

Objectives



By the end of this lesson, you will be able to

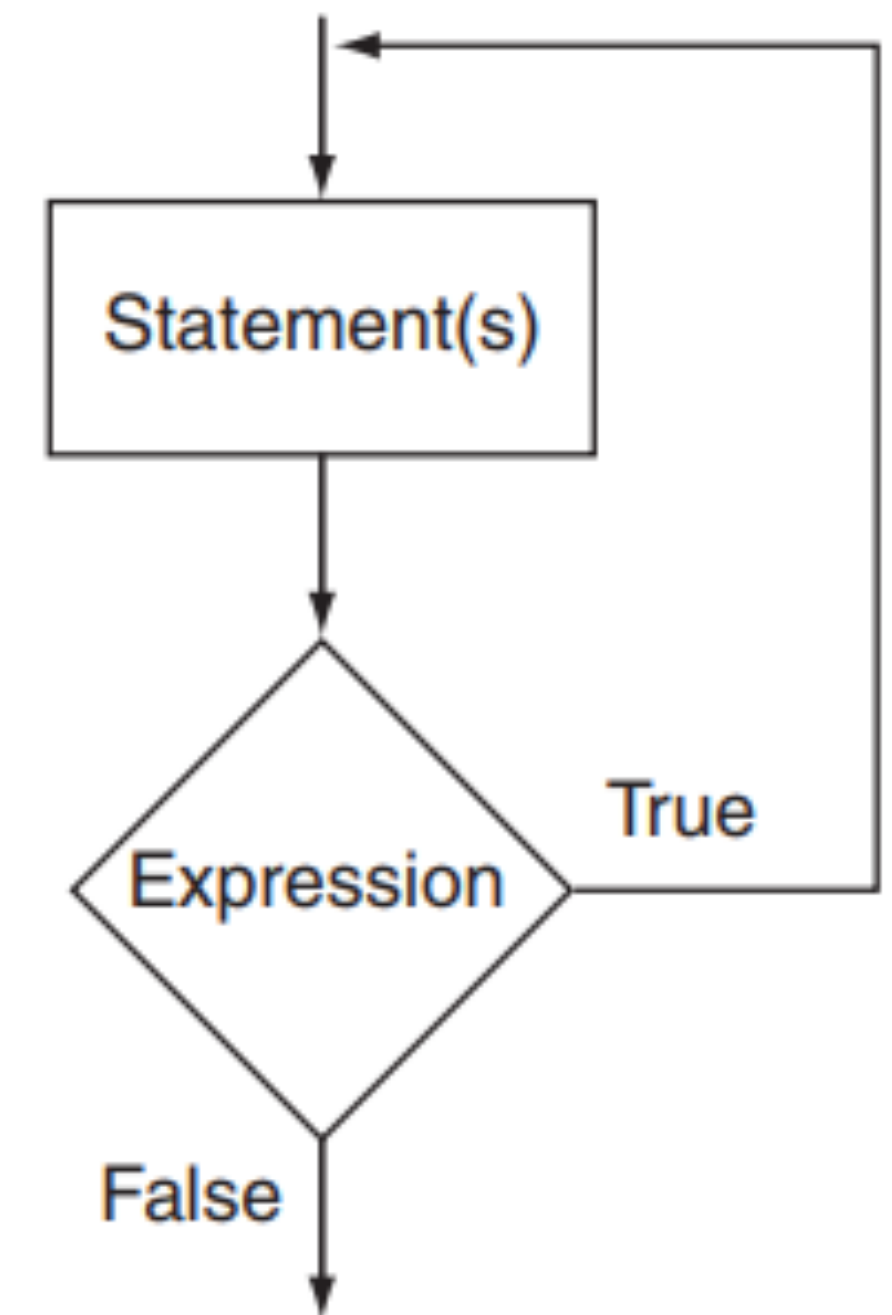
- Understand and utilize the do-while loop for executing code repeatedly, ensuring at least one execution.
- Differentiate between while and do-while loops, understanding their distinct execution methods and choosing the appropriate loop based on program needs.
- Learn about sentinels, special values marking the end of input or signaling conditions within loops, ensuring proper loop termination and effective data handling.
- Grasp nested loops' concept for creating intricate patterns, traversing multi-dimensional structures, and solving problems requiring repetitive operation.

The do-while Loop

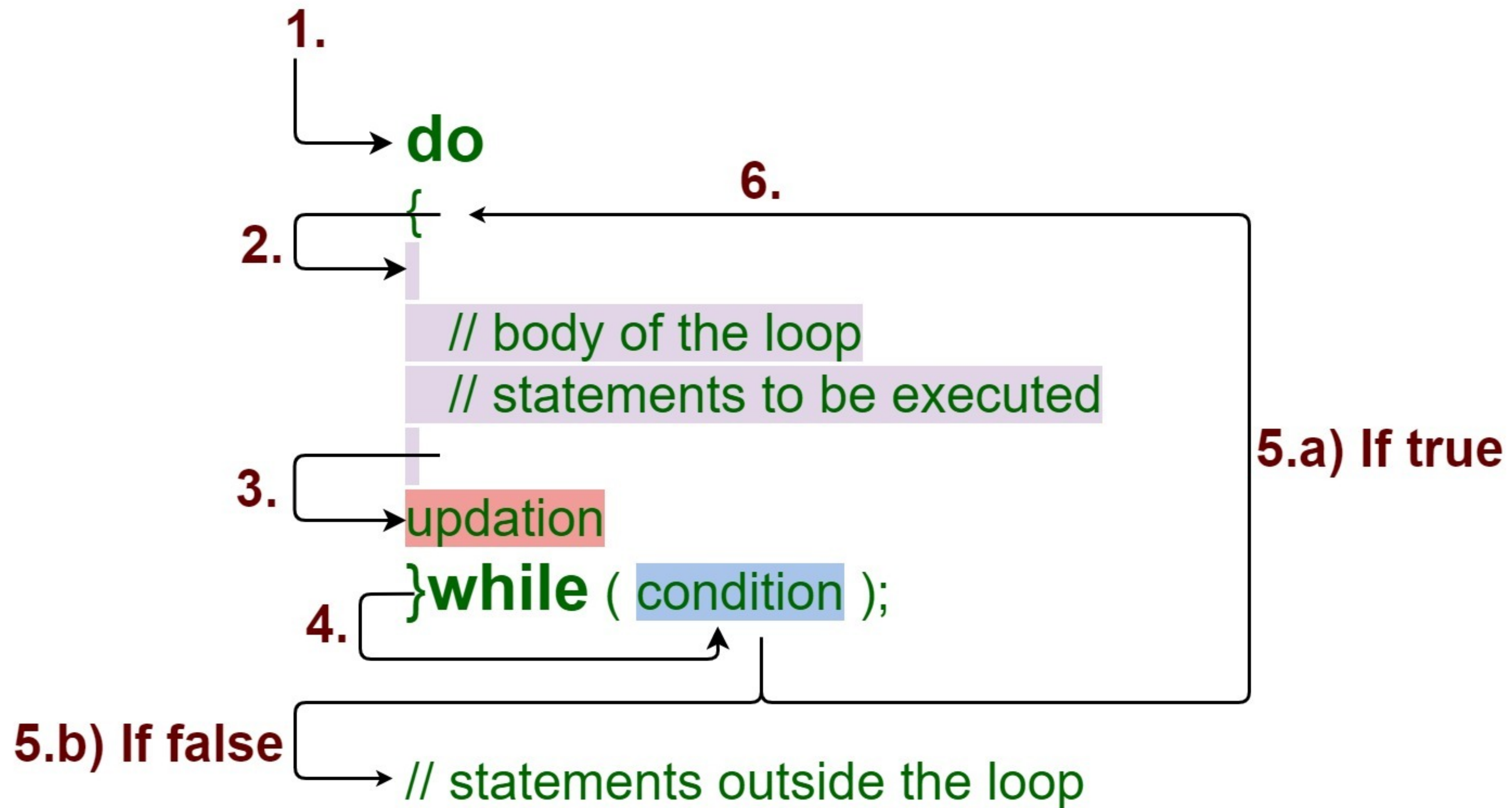


- The do-while loop is a posttest loop.
- It tests its expression after each iteration.
- It always **executes at least one iteration**, even if the expression is initially false.
- While loops test their expression before the first iteration, whereas do-while loops test their expression after the first iteration.
- Format of a do-while loop with a single statement in its body:

```
do
{
    statement;
    statement;
    // Place as many statements here
    // as necessary.
} while (expression);
```



Do - While Loop



Example #1-1



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

int main() {
    int number;

    do {
        cout << "Enter a positive number: ";
        cin >> number;
    } while (number <= 0);

    cout << "Thank you for entering a positive number!\n";

    return 0;
}
```

Output

```
Enter a positive number: -4
Enter a positive number: 4
Thank you for entering a positive number!
```

Example #1-2



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

    int number;

    do {
        cout << "Enter a positive number: ";
        cin >> number;

        if(number <= 0) {
            cout << "Please enter a number greater than 0!\n";
        }
    } while(number <= 0);

    cout << "Thank you! You entered: " << number << endl;

    return 0;
}
```

Output

```
Enter a positive number: -3
Please enter a number greater than 0!
Enter a positive number: -5
Please enter a number greater than 0!
Enter a positive number: 6
Thank you! You entered: 6
```

Compare?!



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

int main() {
    int number;

    do {
        cout << "Enter a positive number: ";
        cin >> number;
    } while (number <= 0);

    cout << "Thank you for entering a positive number!\n";

    return 0;
}
```

```
Enter a positive number: -4
Enter a positive number: 4
Thank you for entering a positive number!
```

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

    int number;

    do {
        cout << "Enter a positive number: ";
        cin >> number;

        if(number <= 0) {
            cout << "Please enter a number greater than 0!\n";
        }
    } while(number <= 0);

    cout << "Thank you! You entered: " << number << endl;

    return 0;
}
```

```
Enter a positive number: -3
Please enter a number greater than 0!
Enter a positive number: -5
Please enter a number greater than 0!
Enter a positive number: 6
Thank you! You entered: 6
```

Example #2



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

    string name;
    int quiz1, quiz2, quiz3;
    double average;
    char again; // To hold Y/N

    do{
        cout<<"Input student name: ";
        cin>>name;
        cout << "Enter the mark of 3 quizzes: ";
        cin >> quiz1 >> quiz2 >> quiz3;

        // Calculate and display the average.
        average = (quiz1 + quiz2 + quiz3) / 3.0;
        cout << "Name: "<<name<<".\t The average: " << average << ".\n";

        // Does the user want to average another set?
        cout << "Do you want to average another set? (Y/N) ";
        cin >> again;
    } while (again == 'Y' || again == 'y');

    return 0;
}
```

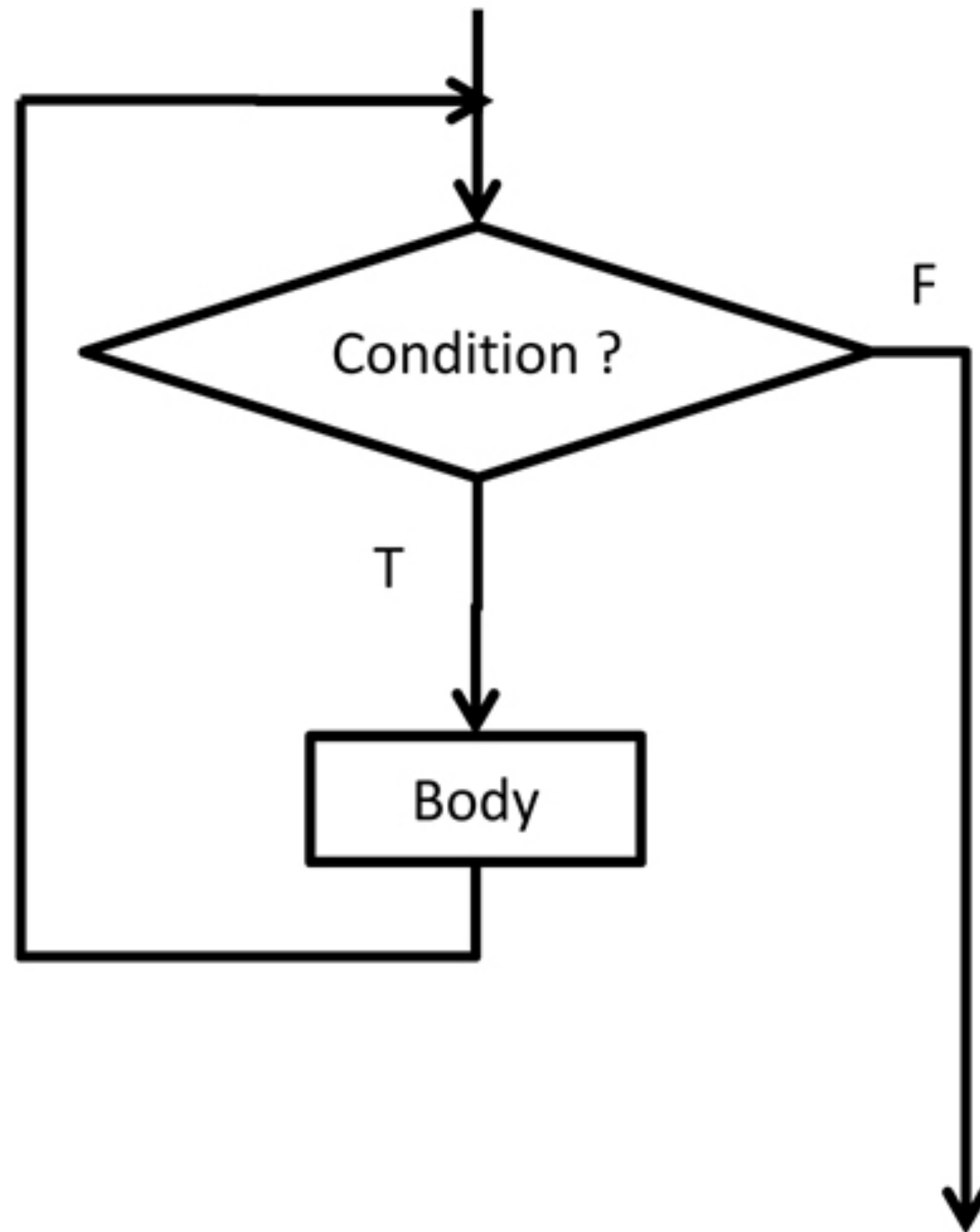
Output

```
Input student name: Alan
Enter the mark of 3 quizzes: 3 4 2
Name: Alan.  The average: 3.
Do you want to average another set? (Y/N) y
Input student name: Kamal
Enter the mark of 3 quizzes: 3 4 3
Name: Kamal.  The average: 3.33333.
Do you want to average another set? (Y/N) n
```

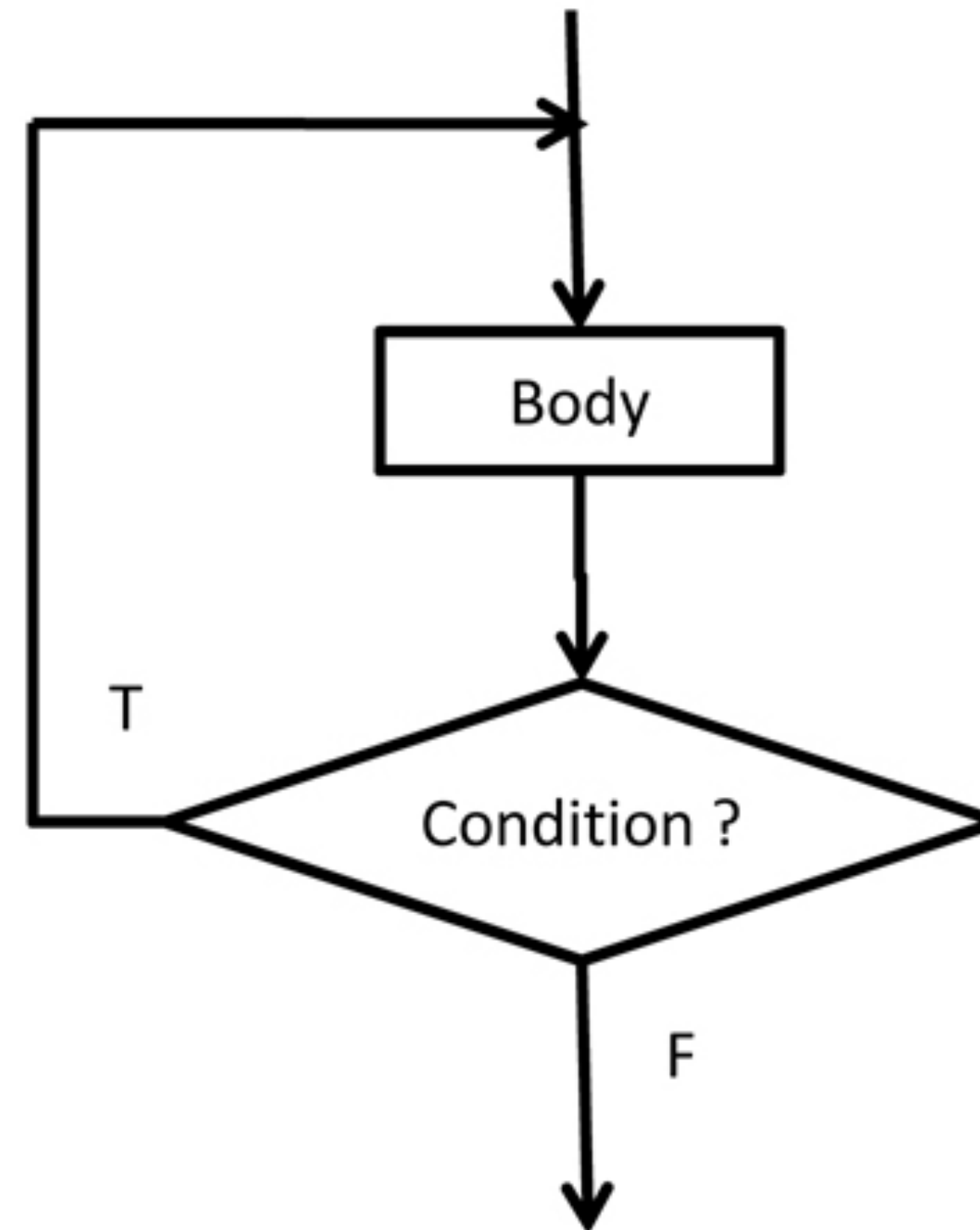
Do Whlie vs While loop

While versus Do-While Loops

while(condition)
body;

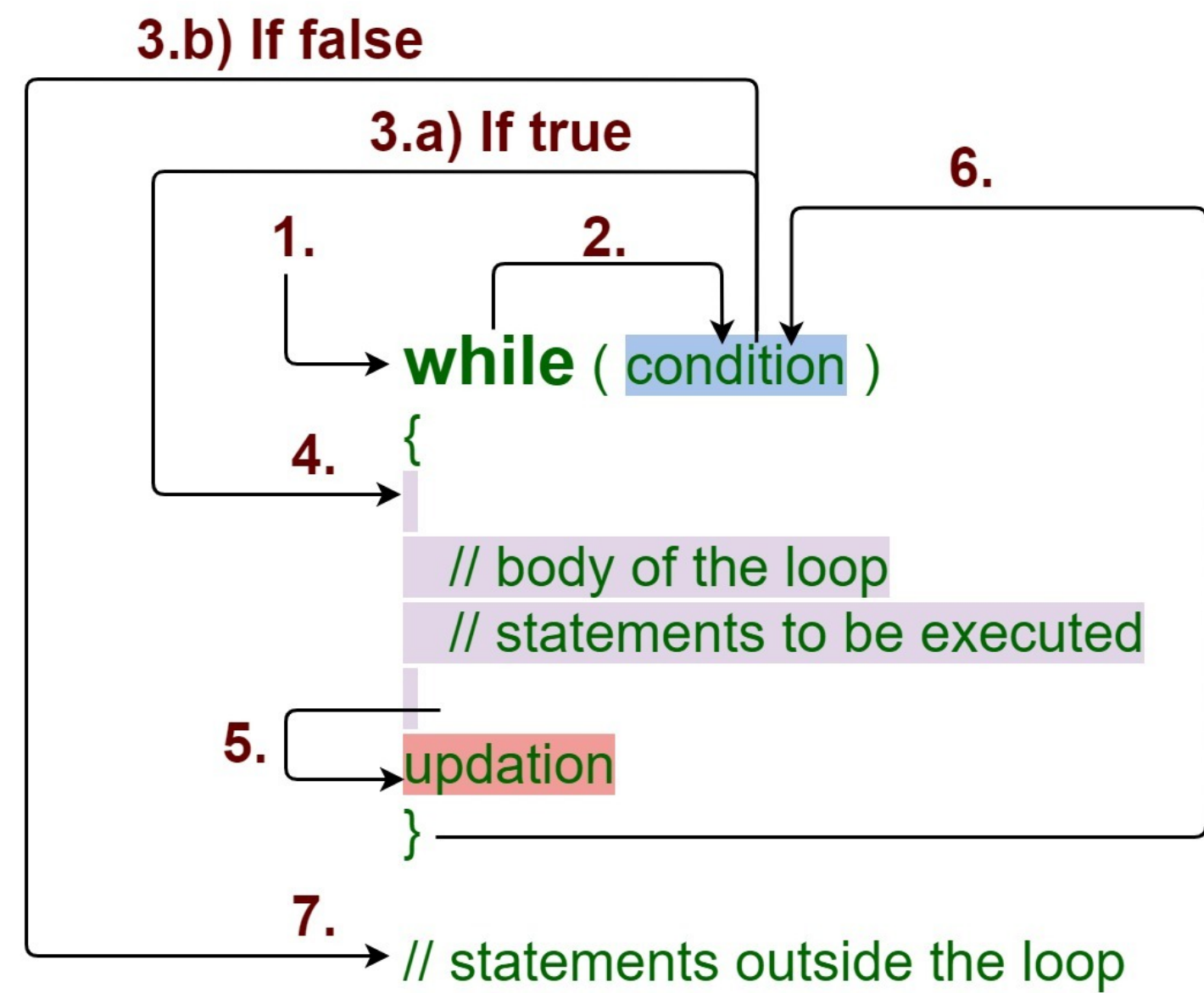


do {
body;
} while(condition);

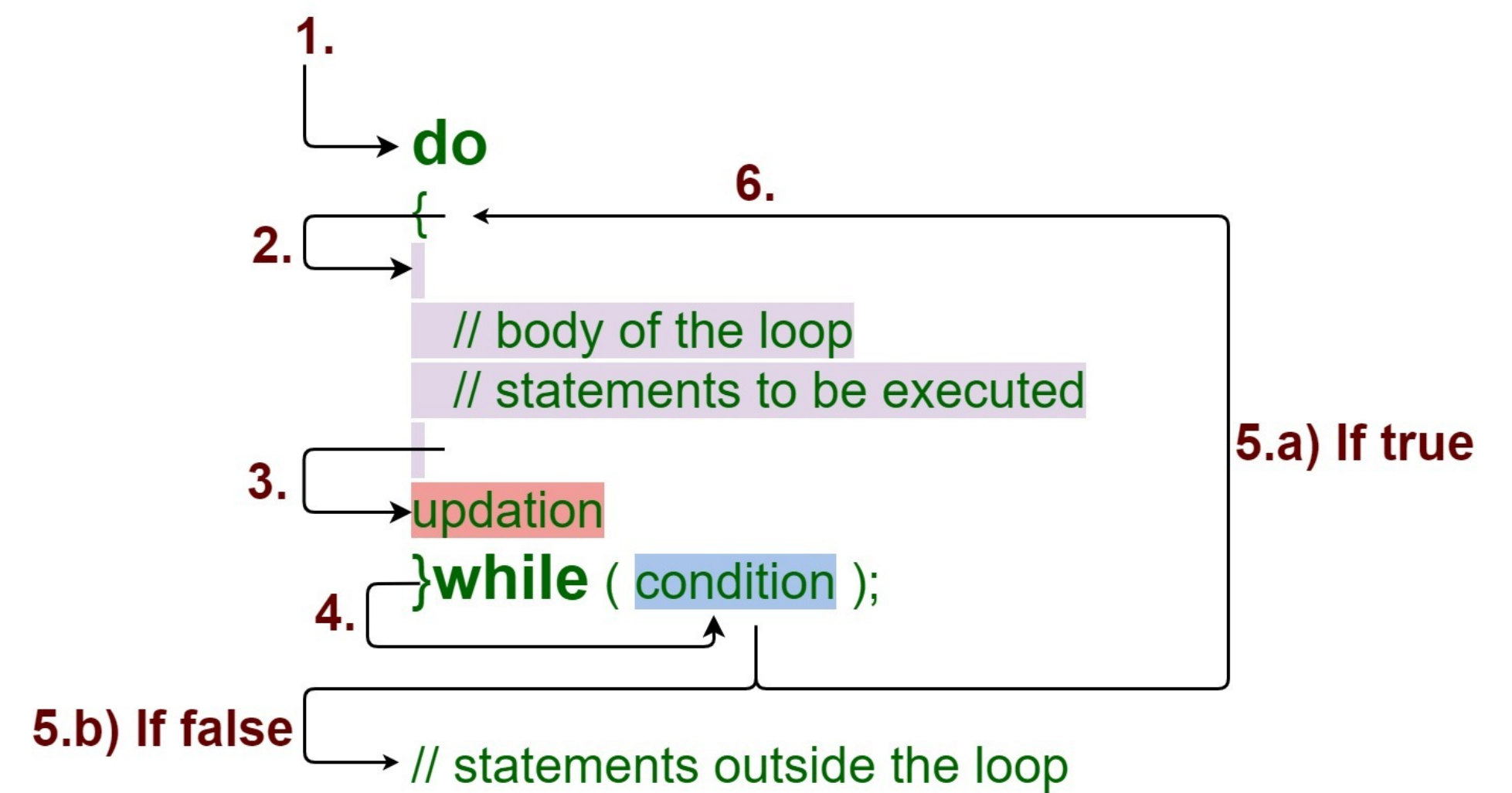


Do Whlie vs While loop

While Loop



Do - While Loop

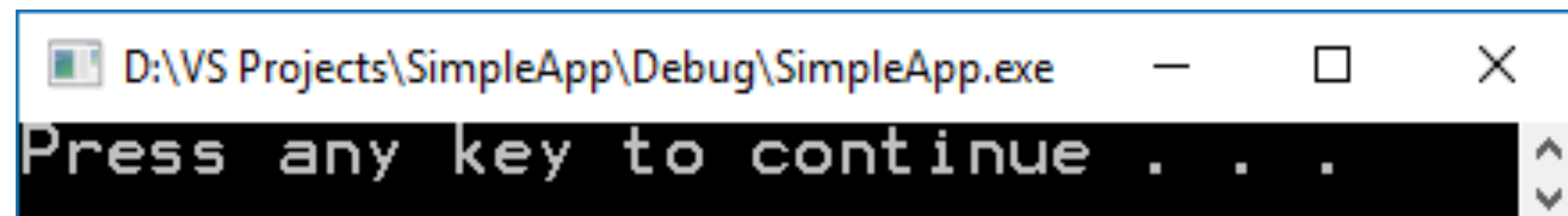


Do Whlie vs While loop



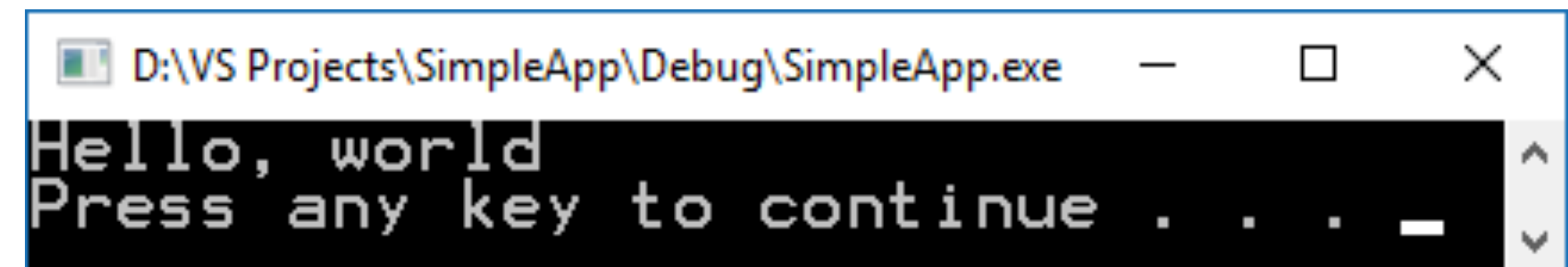
```
int a = 5;
while ( a <= 3)
{
    cout << "Hello, world" << endl;
    a++;
}
```

0 loop



```
int a = 5;
do
{
    cout << "Hello, world" << endl;
    a++;
} while ( a <= 3);
```

1 loop



- A sentinel is a special value denoting the end of a list of values.
- It is distinct from other values in the list, serving as a signal that no more values need to be entered.
- When the user inputs the sentinel value, the loop terminates.

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

    int grade,counter=0,total=0;

    cout << "Enter a grade (-1 to exit): ";
    cin >> grade;
    while (grade != -1){
        total = total + grade;
        counter++;
        cout << "Enter a grade (-1 to exit): ";
        cin >> grade;
    }
    cout << "Average of grades are: " << total / float(counter);

    return 0;
}
```

Output

```
Enter a grade (-1 to exit): 78
Enter a grade (-1 to exit): 57
Enter a grade (-1 to exit): 98
Enter a grade (-1 to exit): 65
Enter a grade (-1 to exit): 77
Enter a grade (-1 to exit): 83
Enter a grade (-1 to exit): -1
Average of grades are: 76.3333
```

Example



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

    int number;
    int count = 0;

    cout << "Enter numbers (-1 to stop):\n";

    do {
        cin >> number;

        if(number != -1) {
            count = count + 1;
        }
    } while(number != -1);

    cout << "You entered " << count << " numbers.\n";

    return 0;
}
```

Output

```
Enter numbers (-1 to stop):
3
4
6
-1
You entered 3 numbers.
```

Question



Write a C++ program that calculates the average of a set of grades entered by the user. The user should be able to enter multiple grades, and the input should terminate when the user enters -1. The program should then output the average grade. If no grades (other than -1) are entered, it does not need to display anything.

Answer

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

    int grade, count=0;
    int total=0;
    double avg;
    do {
        cout<<"Input the grade (-1 to stop): ";
        cin>>grade;
        if(grade!=-1){
            total=total+grade;
            count++;
        }
    } while(grade!=-1);
    if(count>0){
        avg=double(total)/count;
        cout<<"You input "<<count<<" numbers and the AVG= "<<avg;
    }
    return 0;
}
```

Deciding Which Loop to Use?



- **While Loop:**
 - Conditional loop repeating as long as a condition exists.
 - Pretest loop: It checks the condition before the iteration.
 - Suitable when the loop shouldn't iterate if the condition is false initially.
- **Do-While Loop:**
 - Conditional loop that iterates at least once.
 - Posttest loop: It checks the condition after the first iteration.
 - Ideal for scenarios where you always want the loop to run at least once, like repeating a menu.
- **For Loop:**
 - Pretest loop with built-in expressions for initialization, testing, and updating.
 - Convenient for controlling iterations using a counter variable.

- **Set Clear Objectives:** Before using a loop, define the purpose and goals of the loop.
- **Choose the Right Loop Type:** Understand the different types of loops available and choose the one that best fits your task.
 - Use a **for loop** for a known number of iterations,
 - a **while loop** for indefinite iterations with a condition, and
 - a **do-while loop** when you want to ensure the loop body executes at least once.
- **Initialize and Update Variables Carefully:** Initialize and update loop variables meticulously for accuracy.
- **Use Break wisely:** Utilize the break statement to exit a loop prematurely when a specific condition is met.

- **Definition:**

- Nested loops are loops within loops.

- **Structure:**

- Outer loop controls the iteration over rows.
- Inner loop manages the iteration over columns.

- **Usage:**

- Create complex patterns and structures.
- Traverse multi-dimensional arrays.
- Solve problems requiring repetitive operations.

- **Example:**

- Printing patterns, such as squares, triangles, or rectangles.
- Accessing elements in matrices or multi-dimensional arrays.

Nested Loops - Example #1



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

int main() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << "* ";
        }
        cout << endl;
    }
    return 0;
}
```

Output

```
* * *
* * *
* * *
```

Nested Loops - Example #2

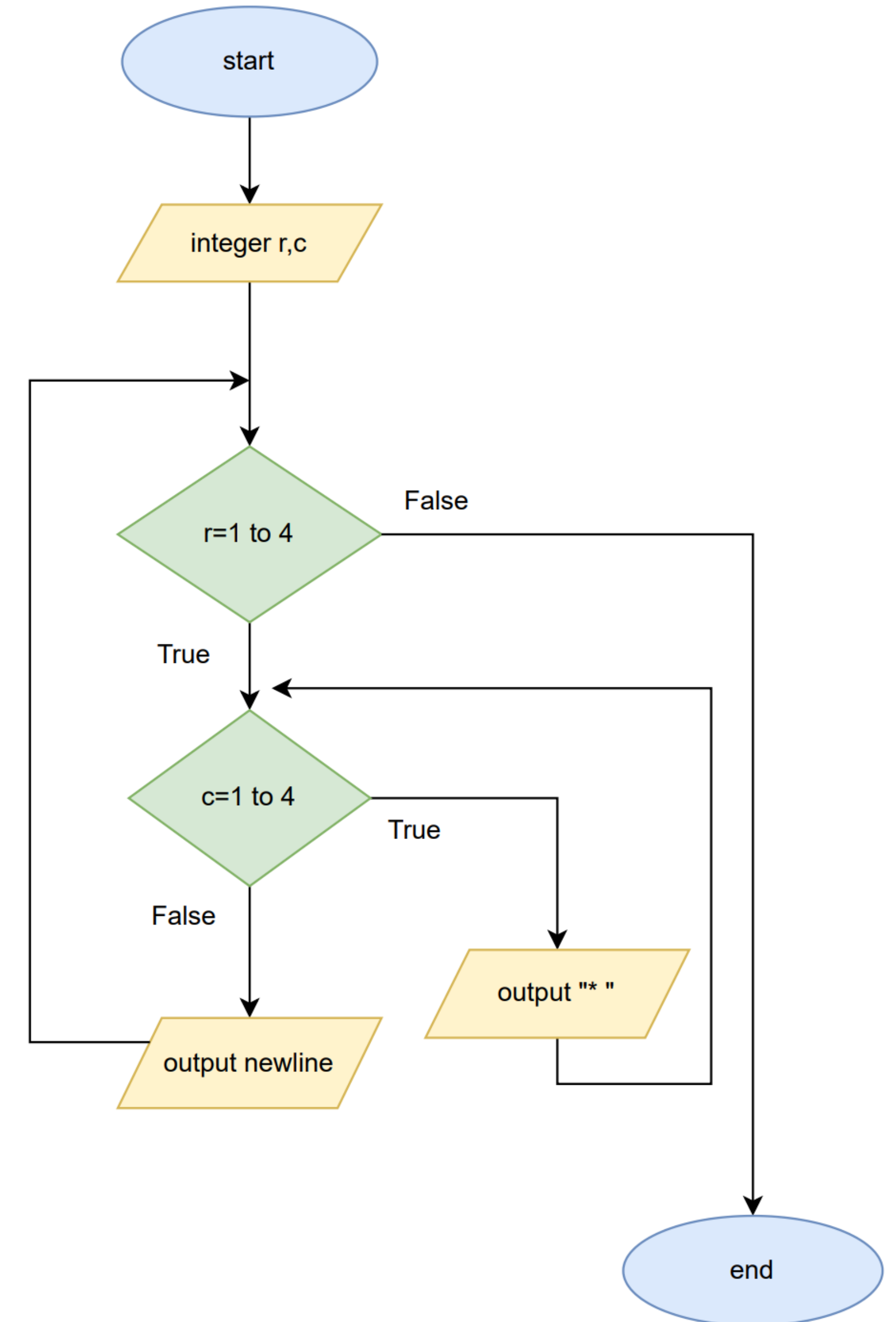
```
#include <iostream>
using namespace std;
int main(){

    for (int r = 1; r <= 4; r++){
        for (int c = 1; c <= r; c++){
            cout << "* ";
        }
        cout << endl;
    }

    return 0;
}
```

Output

```
*
* *
* * *
* * * *
```



Nested Loops - Example #3



```
1  #include <iostream>
2  using namespace std;
3  int main(){
4  for (int i = 1; i <= 10; i++) {
5      for (int j = 1; j <= 10; j++) {
6          cout << i * j << "\t";
7      }
8      cout << endl;
9  }
10 }
```

Output

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Nested Loops - Example #3 (Another way)

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      for (int i=1;i<=10;i++){
7          for (int j=i;j<=i*10;j=j+i){
8              cout<<j<<"\t";
9          }
10         cout<<endl;
11     }
12     return 0;
13 }
```



Output

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Question



1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

Thank You

