UNIVERSITY OF LIVERPOOL

Application Development with C++ (ELEC362)

Lecture 4: Control structures and loops

mihasan@liverpool.ac.uk

Previous lecture

- The data types of variables, their declaration and initialisation methods were discussed.
- Operators in C++ and their precedence were discussed.
- Variable casting was discussed.
- Arrays declarations and initialisations were discussed.
- Further reading: http://www.cplusplus.com/doc/tutorial/

This lecture

- What is covered in this lecture?

  Control structures in C++ programmes

- Why it is covered?

  Control structures allow for complex algorithms to be implemented.

- How are topics covered in this lecture:

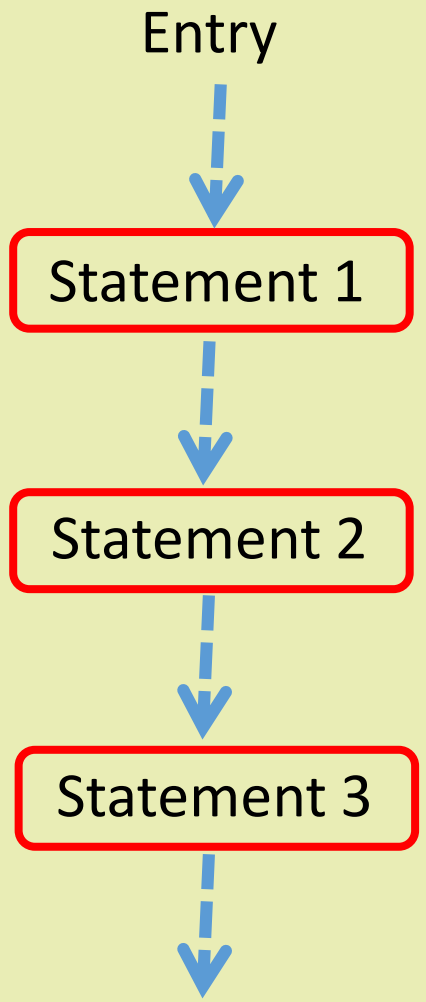
  5 source codes and a practical example.

Control structure definition and types

- Control structures are parts of codes which are used to alter the flow of execution of a programme.
- Up until this point all the codes we have seen are *sequential* (i.e. executes one statement after another).
- In addition to sequential structures, there are *selection* structures and *repetition* structures.

Practical note:

- When explaining algorithms always use flow charts.

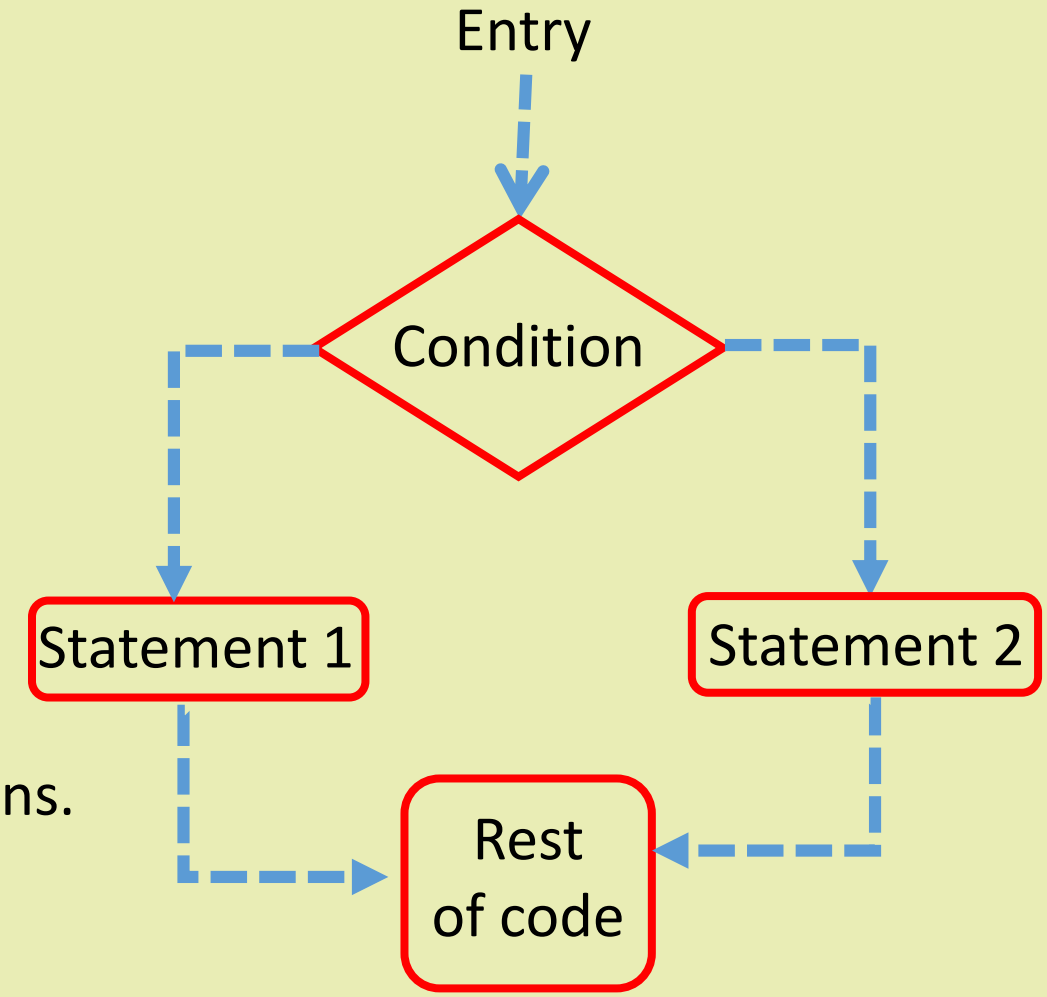


Selection structure : if-else

- The main selection structures in C++ are *if-else* and *switch* statements.
- The syntax of if-else structure is as follows:

```
if (condition)
     { //statements (condition is true);
     }
else {//statements(condition is false);
      }
```

- Conditions can be Boolean variables or expressions.
- Go to L4D1.cpp



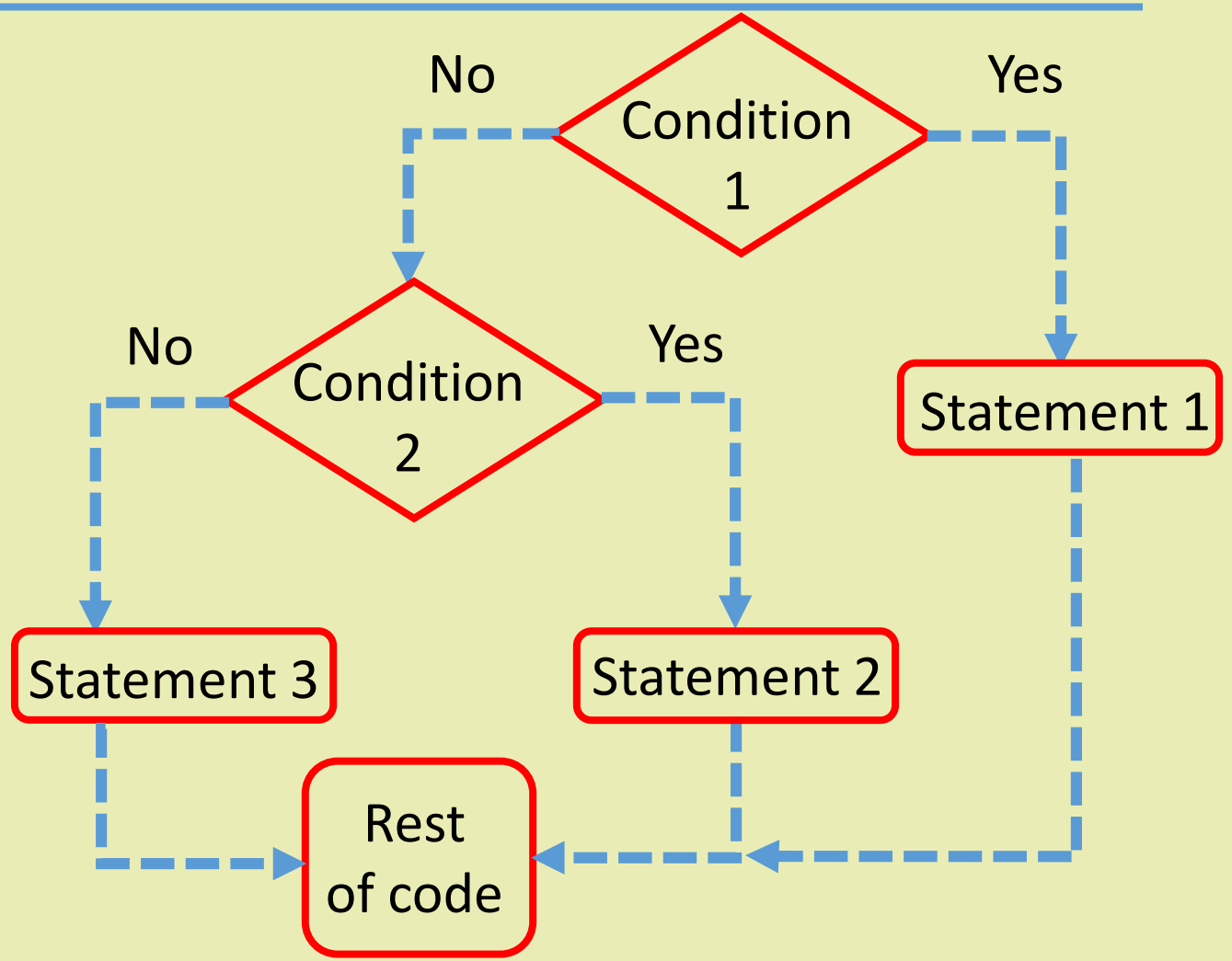
Selection structure: if-else

- *if-else* structure can be nested.

```
if (condition1)
     { //Statement1
     }
else if (condition2)
          {//Statement2
          }
      else {//Statement3 }
```

- A compact way to write if-else is using the operator “?”:

```
Condition? If_true : If_false
```

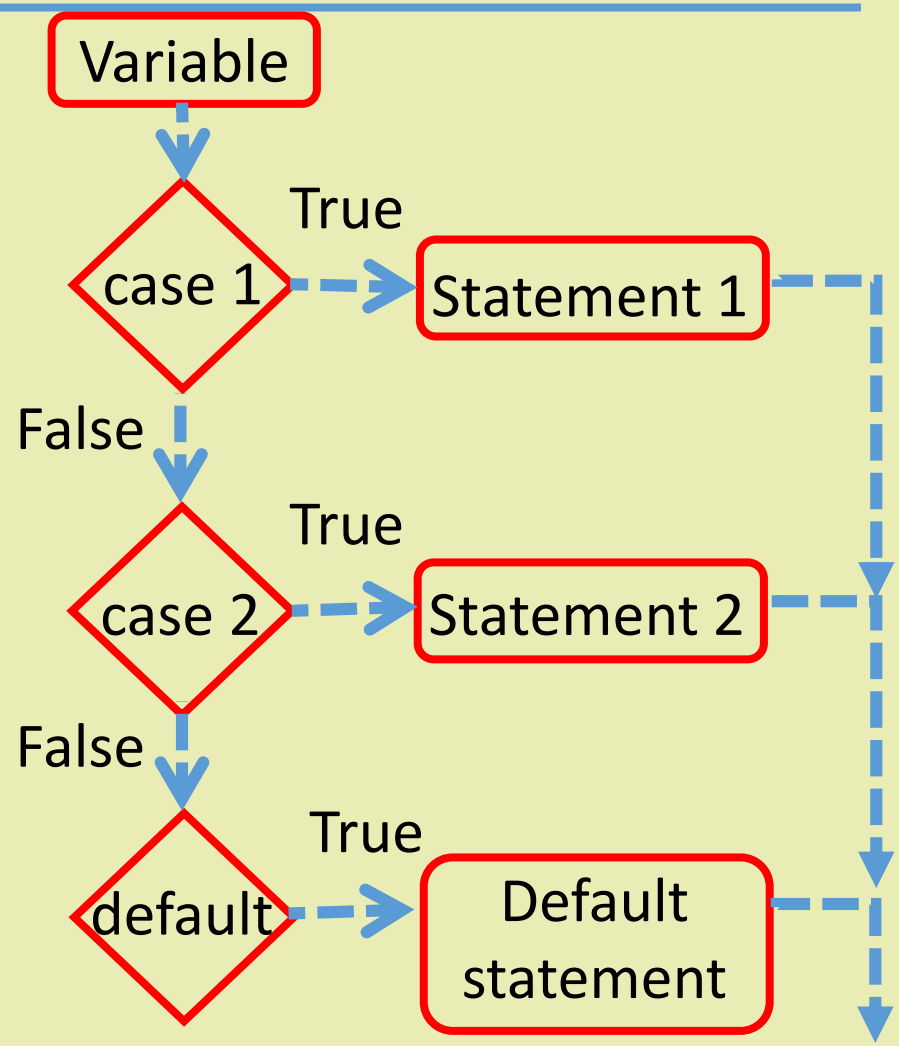


Selection structure: switch-case

- *Switch-case* is more efficient than nested *if-else.*
- The syntax of switch-case structure is as follows:

```
switch (variable)
        {case 1:
        //do something
        break;
        case 2:
        // do something else
        break;
        default:
        // do this when out of options
        break;}
```

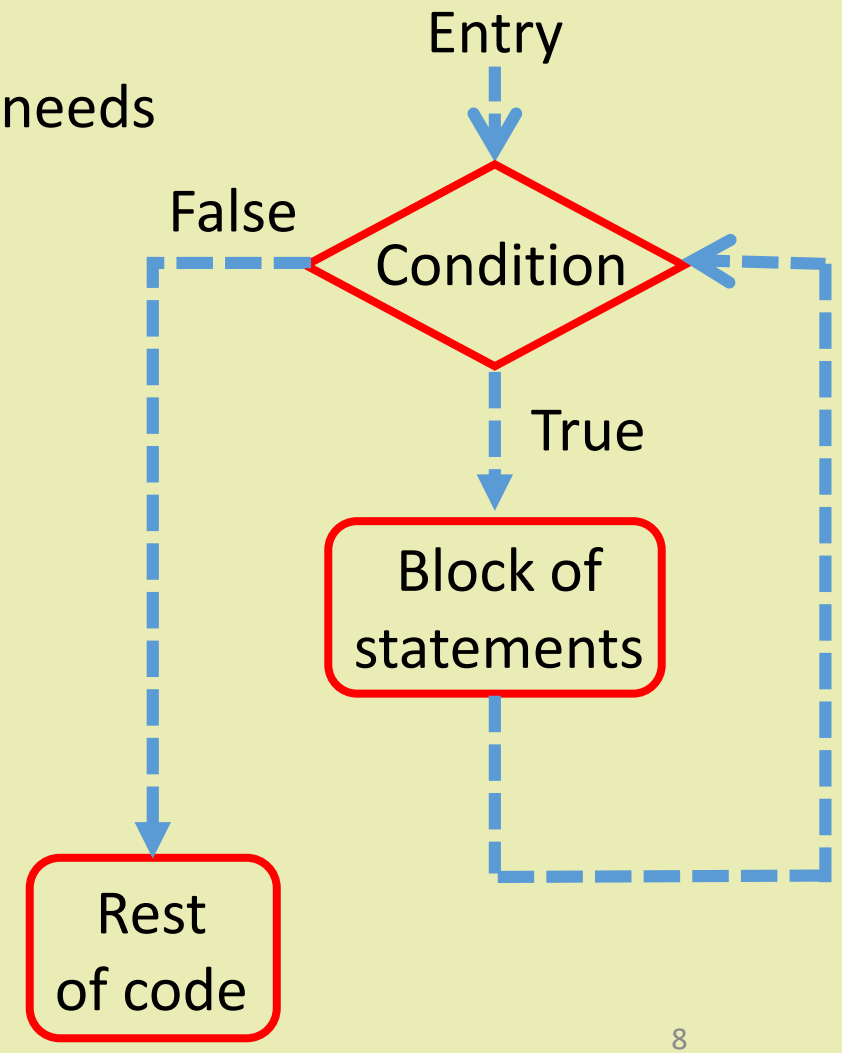
- Go to L4D2.cpp



Repetition structure

- Repetition structures are used when a part of the code needs to be repeated.
- The most common repetition structures in C++ are:

1. For loops: controlled by the number of runs.
2. While loops: controlled by a condition.
3. Do-while loops: a modified version of a while loop.



For loop

- Syntax:

```
for (unsigned int counter = 0; counter < N; counter++)
     { // statements;
     }
```

Counter variable initialisation

Condition checking Counter handling

- Mechanism:
  1. Loop counter is initialised.
  2. Condition is checked.
  3. If true, Statements in the loop's body are executed.
  4. Counter handling.
  5. Repeat steps 2 to 5.

- Go to L4D3.cpp

Management of For loops

- The execution of a For loop can be altered by using **`break`** , **`continue`**, or **`goto`**.

Statement	functionality
`break`	Terminates the loop even if the condition on the loop counter is true
`continue`	Starts a new iteration in the loop
`goto`	Can be used to jump from any part of the code to another

- Statements including **`break`** and **`goto`** can appear anywhere in the code (not necessarily in a loop).

Good practice note:

- **`goto`** is a depreciated statement and is no longer in use by professional C++ developers.

Management of For loops

- Example:

```
for (unsigned int counter = 0; counter < 10; counter++)
    {

      if ( counter == 5) continue; // Skips 5

      if (counter > 6 ) goto Last; //Moves execution to "Last" statement

      if (counter == 9) break;

     Last: std::cout << counter <<endl;
     }
```

While loop and Do-while loop

- Syntax:

```
while (condition)
      { // statements;
      }
```

Boolean expressions

```
do { // statements;
     }
while (condition)
```

While loop Mechanism:

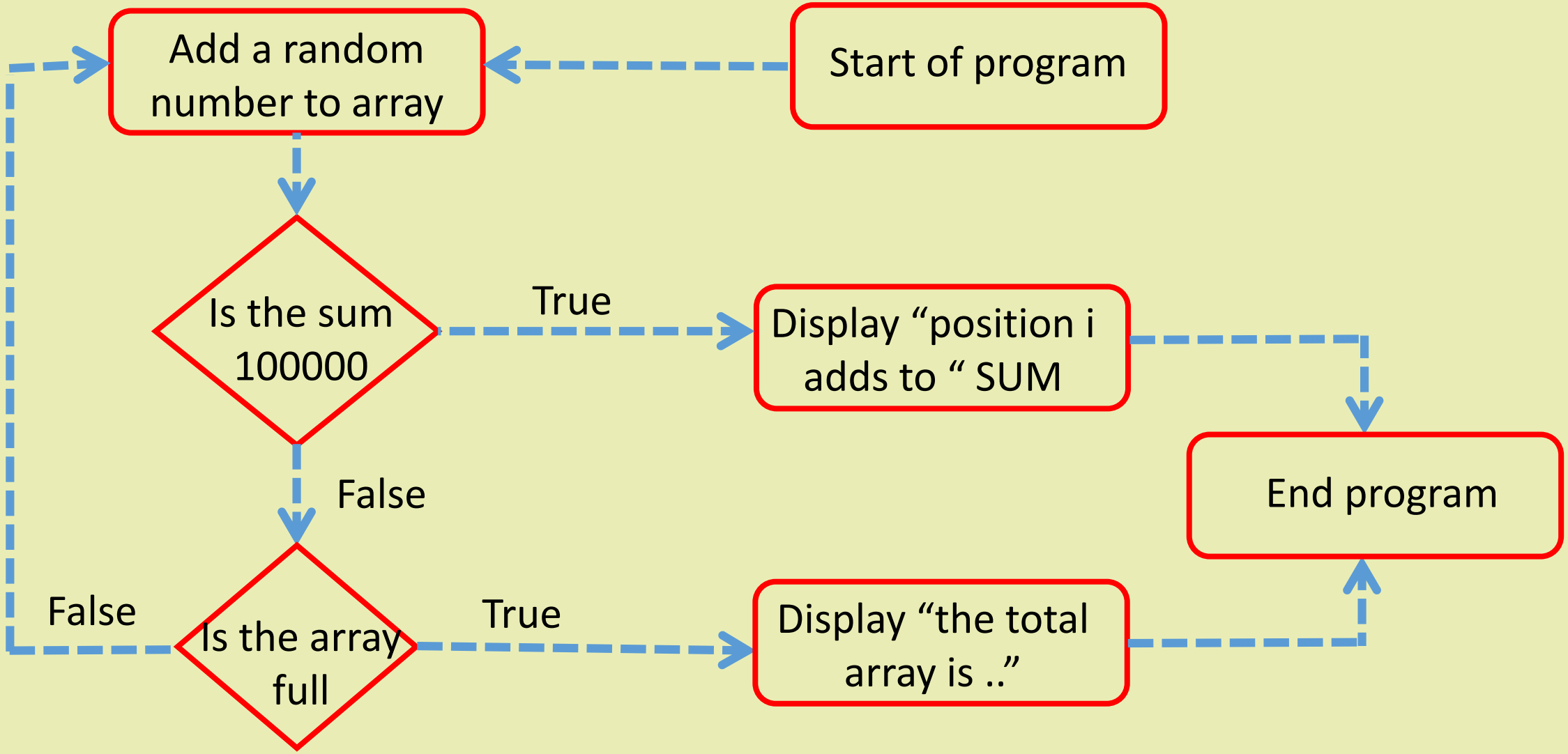
1. Check condition.
2. If true, execute statements.
3. Repeat steps 1-2.

Do-while loop Mechanism:

1. Execute statements.
2. Check condition.
3. If true, Repeat steps 1-2.

- Remember to control the condition from within the loop to avoid infinite loop.
- Go to L4D4.cpp

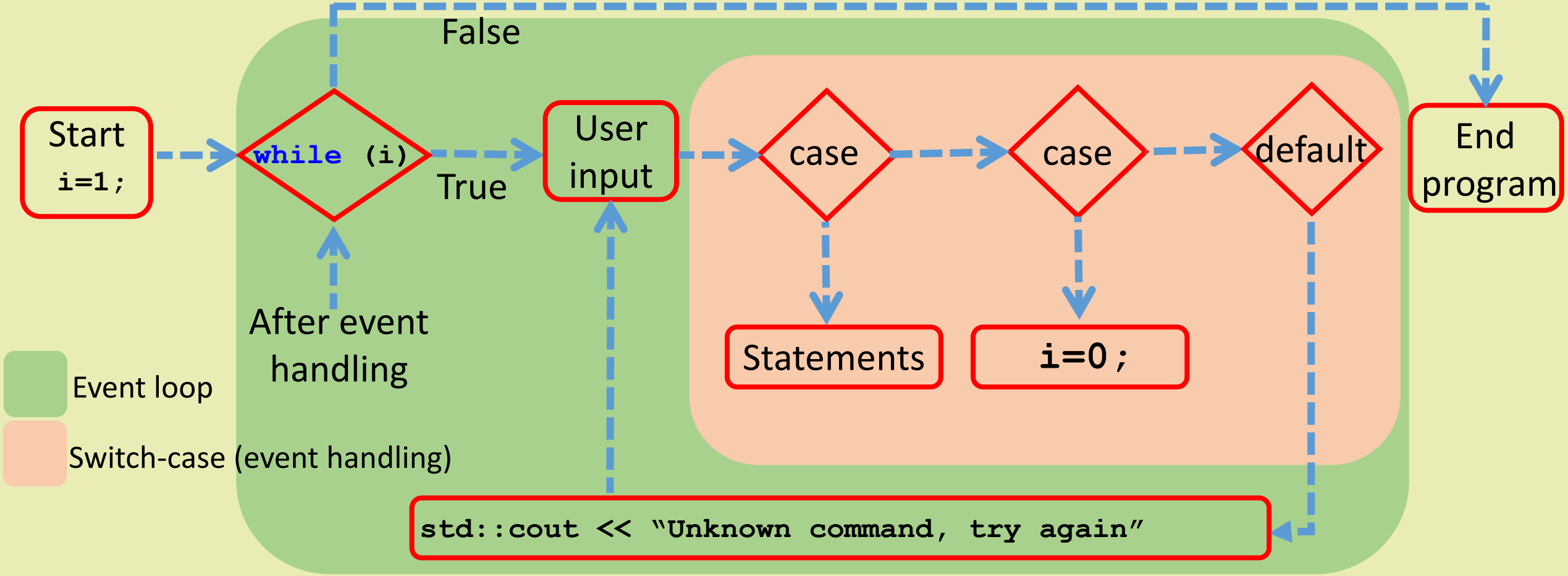
How L4D4 works



Practical Example: Event loop

- The core of any programme is a code structure known as “Event loop”.
- When the programme starts, an infinite loop is initialised.
- The body of the loop waits for the user’s input (events).
- When users input something, their input is dealt with by a switch-case structure.
- For every possible input, there is one case in the switch-case structure that handles the input (this is known as event handler).
- One of the cases is assigned to terminate the loop (i.e. close the programme).
- Example :cmd.exe (in Windows).
- Go to L4D5.cpp

Practical Example: Event loop



Practical note: Implement an event loop for any console based application you develop.

Summary

- Three types of control structures (sequential, selection, and repetition structures) were discussed.
- If-else and switch-case structures were discussed under selection structures.
- For loop, While and Do-while loops were discussed under repetition structures.
- The concept of event loops was discussed.