UOW AUSTRALIA

CSCI251 Spring-2024

Advanced Programming

C++ Foundations III:
Control structures

Similarity with Java

- Control structures in Java and in C++ are very similar.
- We will go through an overview fairly quickly.
- If you are unfamiliar/uncomfortable with the concepts or syntax, the exercises in lab cases will be useful.

Control structures: if

```
if (Boolean expression is true){
        statements …
}
```

For example:

```
if ( age >= 18 )
  cout << "You must vote!" << endl;
```

With one line you can get away without using the braces { … }, but it's often a good idea using it anyway.

Control structures: if-else

```
if (Boolean expression is true){
   statements … }
else {
   other statements … }
```

For example:

```
if ( age >= 18 )
  cout << "You must vote" << endl;
else
  cout << "You cannot vote" << endl;
```

Control structures: if-else-if-else

```
if (Boolean expression is true){
    statements … }
else if (…){
    other statements … }
else {
    still more statements … }
```

For example:

```
 if ( age >= 18 ){
     cout << "You must vote!" << endl;
 }
 else if ( age == 17 ){
      cout << "You will be able to vote soon!" << endl;
 }
 else {
      cout << "You cannot vote!" << endl;
 }
```

Logic operators, … compound Boolean expressions

- NOT: `!`

```
if (countryCode!=61 )
```

- AND: `&&`

```
if ( age>=18 && countryCode==61 )
```

- OR: `||`

```
if ( countryCode==61 ||
  countryCode==64 )
```

Practice 1

Control structures: switch

- The “if” statement is good for Boolean tests, where there are only two possible outcomes.
- For multiple outcomes, the “switch - case” structure may be more suitable.
- The `expression` below is evaluated once.

```
switch(expression) {
   case 1:
                  actions;
                  break;
   case 2:
                  actions;
                  break;
   . . .
                  break;
   default:
                  cout << "The case is not defined" << endl;
}
```

The use of `break` and `default` is optional.

The actions for the cases that follow the matching case, including `default`, are applied until a `break` is reached.

Practice 2

Switch in C++17

- Switch has additional functionality from C++17…

```
g++ -std=c++17 code.cpp
```

- See: http://en.cppreference.com/w/cpp/language/switch
- It's a fairly minor change that supports the inclusion of an initialisation statement.
  - Most likely to be useful for declaring a variable only to be used within the switch.

```
switch (int num = randint(2); num){
   case 0: std::cout << "0" ; break;
   case 1: std::cout << "1" ; break;
   case 2: std::cout << "2" ; break;
 default: std::cout << "Something went wrong!" ;
}
```

```
int randint (int x) {
        static std::mt19937 mtg(time(0));
        return std::uniform_int_distribution<int> (0, x) (mtg);}
```

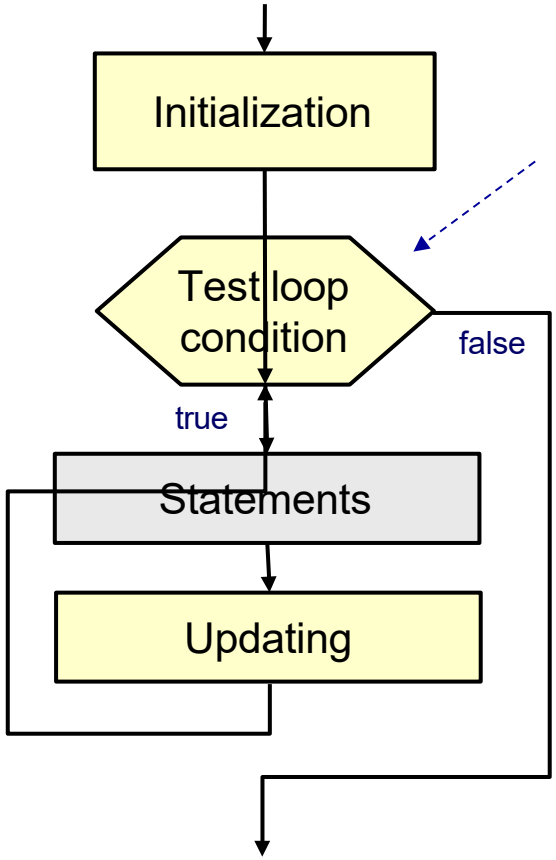
Control structures: Repetition

UOW AUSTRALIA

- Repetition statements are used to repeat an action as long as a condition remains `true`.
- As in Java there are three basic loop types:
  - Pre-test loop `for`
  - Pre-test loop `while`
  - Post-test loop `do…while`
- All have three components.
  - Initialize loop
  - Test loop condition
  - Update

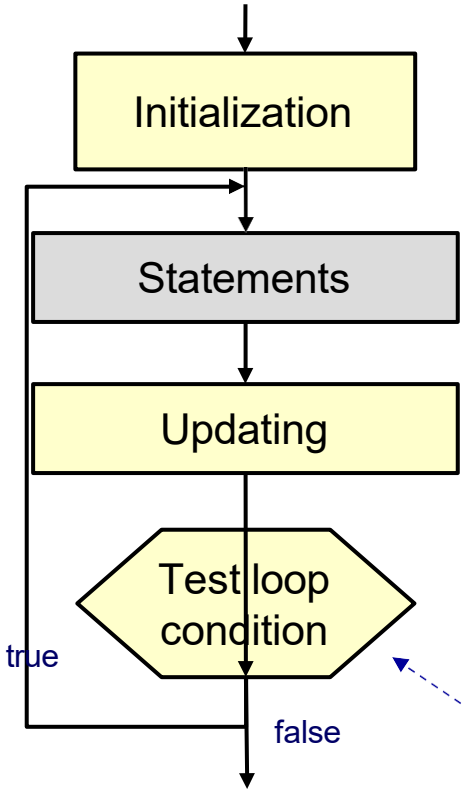
Pre-test and Post-test loops

Pre-test loop



The loop condition is tested before the statements are executed

Post-test loop



The loop condition is tested after the statements are executed

For loops…

- The for loop is a version of a pre-test loop that has a more convenient syntax to implement a determined number of repetitions.

```
for( initialisation; condition; update) {
        // do things …
}
```

- We might want to list the first 10 terms of the series resulting from summing the sequence of negative powers of 2…

```
1 + 1/2 + 1/4 + 1/8 + ...     =>     1, 1.5, 1.75, 1.875, …
```

```
float term = 1, x=2;

for (int counter=1; counter<11; counter++) {
    std::cout << "Term : " << counter << " is " << term << std::endl;
    term +=1/x;
    x*=2;
}
```

```
Term : 1 is 1
Term : 2 is 1.5
Term : 3 is 1.75
Term : 4 is 1.875
Term : 5 is 1.9375
Term : 6 is 1.96875
Term : 7 is 1.98438
Term : 8 is 1.99219
Term : 9 is 1.99609
Term : 10 is 1.99805
```

Variations on the `for` loop

Several initialization expressions separated by commas.

```
for( int factorial=1, counter=1; counter <= n; ++counter)
    factorial *= counter;
```

No initialization expressions.

```
for(   ; n > 0; n-- )
    printf("*");
```

A simple implementation of a delay (the actual delay time is platform dependent).

```
for( int counter=0; counter < 1000; counter++ ) ;
```

An infinite loop (until it is terminated inside the loop body, by `break`, `return`, …).

```
for(  ;  ;  )
{
   . . .
}
```

The range `for` loop

UOW AUSTRALIA

- C++, C++11 on, supports a range for statement that allows us to step through the elements in a sequence and operate on each in the same way.

```
for( declaration : expression)
    statement
```

- The `declaration` defines the variable to be used when accessing the elements in the sequence, while `expression` is an object representing a sequence.

```
string str("This is a string");
for (char c : str)
      cout << c << endl;
```

- C++20 extends the range `for` to include an initialisation statement.

while and do-while

```
while( condition ) {
        // do things …
}
```

```
do {
        // do things …
} while( condition );
```

- Consider we want to explore the same series as before …

`1 + 1/2 + 1/4 + 1/8 + ...  =>  1, 1.5, 1.75, 1.875, …`

- … but only while the changes are of at least a certain size.

```
const float DIF = 0.000001;

int main(){
    float oldResult = 0, newResult=1, x=2;

    while ( newResult - oldResult > DIF ) {
        std::cout << newResult << "\t" << newResult-oldResult << std::endl;
        oldResult = newResult;
        newResult +=1/x;
        x*=2;
     }
    return 0;
}
```

```
1        1
1.5      0.5
1.75     0.25
1.875    0.125
1.9375   0.0625
1.96875  0.03125
1.98438  0.015625
1.99219  0.0078125
1.99609  0.00390625
1.99805  0.00195312
1.99902  0.000976562
1.99951  0.000488281
1.99976  0.000244141
1.99988  0.00012207
1.99994  6.10352e-05
1.99997  3.05176e-05
1.99998  1.52588e-05
1.99999  7.62939e-06
2        3.8147e-06
2        1.90735e-06
```

Practice 3