# MNXB01 course - C++ module

Caterina Marcon

caterina.marcon@hep.lu.se

# Lecture's goals

You will learn:

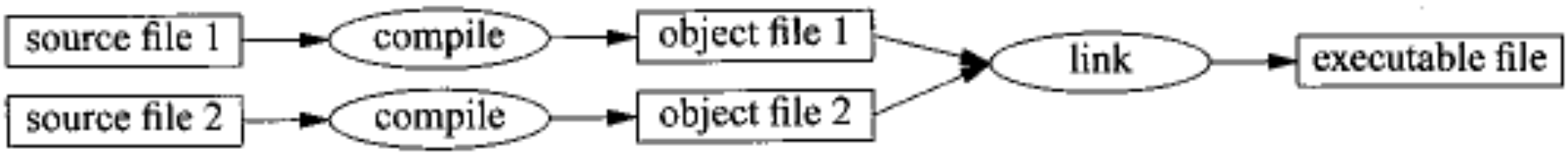
- to use if and if...else selection statements to choose among alternative actions;
- to use for, while repetition statements in a program repeatedly;
- how to read and write to text files;
- to use data structures to represent a set of data items.

# Bibliography


- Problem solving with C++", W. Savitch;
- C++ how to program, P.J. Deitel
- www.cplusplus.com

# Compiled languages

To recap:




- C++ is a compiled programming language (it compiles high-level language to low-level machine code);
- uses C syntax;
- procedural language (program consists of a number of sub procedures that are performed sequentially).

# Compile and run `helloworld.cpp`

Compile with g++ compiler and run executable

[ Terminal ]

```
$ g++ helloworld.cpp
$ ./a.out
Hello World!
```

Executable is always named `a.out`

We can rename it to our own name

[ Terminal ]

```
$ g++ -o helloworld.x helloworld.cpp
$ ./helloworld.x
Hello World!
```

# Control structures: if-then-else

[ C++ ] Control structures: if, else

```
1 if(condition)
2 {
3   statement;
4 }
5 else
6 {
7   statement;
8 }
```

- if evaluates the condition. If it is true, the statement is executed;
- if it is false, the statement in the optional else clause is executed;
- if and else can be nested.

# Example

[ C++ ] Example 1

```
1  int i=7, j=2;
2  if(j==3)
3  {
4    i = 10;
5  }
6  else
7  {
8    i=2;
9  }
10 cout << i << endl; // i is equal to 2
```

# Control structures: if-else if-else

## [ C++ ] Control structures: if-else if-else

```
1  if(condition1)
2  {
3      statement1;
4    // execute if condition1 is true
5  }
6  else if(condition2)
7  {
8      statement2;
9    // execute if condition1 is false and condition2 is true
10 }
11 else
12 {
13     statement3;
14   // execute if condition1 and condition2 are both false
15 }
```

# Example

## [ C++ ] Example 2

```
1  int i=7;
2  if(i==10)
3  {
4   std::cout << "i is 10" << std::endl;
5  }
6  else if (i == 15)
7  {
8    std::cout << "i is 15" << std::endl;
9  }
10 else
11 {
12
13   std::cout << "i is not present" << std::endl;
14 }
```

# Control structures: for-loop

## [ C++ ] for structure

```
1 for(initialize; condition; increment)
2 {
3   statement;
4   // execute body as long as condition is true
5   // after code has been executed, do increment
6 }
```

## [ C++ ] Adding some integers

```
1 int sum = 0; // initialize sum to 0
2 for(int i = 0; i < 5; ++i) // loop over i
3 {
4   sum += i; // shorthand for sum = sum + i
5 }
6 cout << "Sum:" << sum << endl; // what is the sum?
```

- `i++` increments `i` by `1` and returns incremented value.

# Control structures: while-loop

## [ C++ ] while structure

```
1 initialization;
2 while(condition)
3 {
4     statement;
5     increment;
6   // execute body as long as condition is true
7 }
```

## [ C++ ] Adding some integers using while

```
1 int sum = 0;
2 int i = 0;
3 while(i < 10)
4 {
5   sum += i;
6   i++; // increment i
7 }
8 cout << "Sum:" << sum << endl;
```

# Control structures: do-while

## [ C++ ] do-while-loop

```
1 do
2 {
3 statement // execute once, then execute while condition is true
4 }
5 while(condition);
```

## [ C++ ] Adding some integers using do while

```
1 int counter = 1;//initialization counter
2 do
3 {
4   couter++;//increment counter
5 }
6 while (counter <= 10);
```

# Control structures: for-loop, while-loop and do-while

- The for and while loops execute statements while some condition is met; they are functionally equivalent;
- use a for loop when you know how many iterations you want to do;
- use a while loop when number of iterations is unknown (e.g. when the stopping condition depends on user input);
- the do-while loop works like a while loop but the condition is checked at the end of the loop instead of at the beginning; this guarantees that the statement will be executed at least once.

# Control structures: continue, break and switch

- The continue statement is used in loops to skip directly to the next iteration. It works in both for and while loops;

[ C++ ]

```
1 for (int i = 0; i < 10; i++) {
2 if (i == 5) continue; //5 won't be printed
3 std::cout << "i equas " << i << std::endl;
4 }
```

- The break statement is used to exit the loop entirely. It works in for and while loops as switch clauses (next slide).

[ C++ ]

```
1 for (int i = 0; i < 10; i++) {
2 if (i == 5)
3 break; //break loop only if x is 5
4 std::cout << "i is equal to: " << i << std::endl;
5 }
```

# Control structures: continue, break and switch

- The switch clause can be used to replace many of if statements;

[ C++ ]

```
1  switch(variable) {
2      case 0:
3      std::cout << "variable is 0" << std::andl;
4      break;
5
6      case 1:
7      std::cout << "variable is 1" << std::andl;
8      break;
9
10    default:
11    std::cout << "variable is neither 0 nor 1" << std::endl;
12 }
```

# Exercise 1

Write a program in C++ that, given today's temperature in Lund, suggests a suitable outfit :)
(Hint: use if-else if-else construct)

# Exercise 1

[ C++ ]

```
1  //Ex 1: if-else if-else
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6    double Temp;
7
8    cout << "What's the temperature in Lund today? " << endl;
9    cin >> Temp;
10
11   if ( Temp > 25 ) {
12     cout << "You should wear a t-shirt!" << endl;
13   } else if ( Temp > 18 ) {
14     cout << "You should wear a jumper" << endl;
15   } else {
16     cout << "You should wear a jacket" << endl;
17   }
18
19   return 0; //successful termination
20 }
```

# Exercise 2

Write a program in C++ that calculates and prints the sum of integers from 3 to 10.
Use a for loop

# Exercise 2

[ C++ ]

```
1  //Ex 2: for statement
2  #include <iostream>
3
4  using namespace std;
5
6  int main() {
7
8    int sum = 0; //initialize sum
9
10   for (int num = 3; num <= 10; num++) //num++ is equal to number=number+1
11
12     sum += num; //equal to sum = sum + num;
13
14   cout << "The sum is equal to: " << sum << endl; //display results
15
16   return 0; //successful termination
17
18 } //end main
```

# Exercise 3

Write a program in C++ that takes an integer as input and calculates the sum of the integers from 0 up to the inserted number. Print the result.

# Exercise 3

[ C++ ]

```
1  //Ex 3: while loop
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6    int Num;
7    int i = 0; //initialize index i
8    int sum = 0; //initialize sum
9
10   cout << "Enter a number" << endl;
11   cin >> Num;
12
13   while (i < Num) {
14     sum += i; //sum = sum + i
15     i++; //i = i+1 increment
16   }
17   cout << "Sum is equal to :" << sum << endl; //display result
18
19   return 0; //successful termination
20 } //end main
```

# Example: average temperature in Lund

Temperature in rainy Lund were measured from Monday to Sunday. Write a C++ program that takes in the temperatures as a user input, calculates and displays the average temperature of the week.

[ C++ ] Pseudo-code

```
1
2  Define variables;
3  Set total temperature to 0;
4  set the day counter to one;
5
6  While day counter is less than or equal to 7:
7      prompt the user to enter the next temperature;
8      input the next temperature;
9      add the temperature into the total temperature;
10     add one to the day counter;
11
12 Set the temperature average to the total temperature divided by 7;
13 Print the temperature average;
```

# Example: average temperature in Lund

[ C++ ]

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5  double total; //sum of the temperatures
6  int dayCounter; //number of day to be entered next
7  double temperature; //temperature value entered by the user
8  double average; //average of temperatures
9  total = 0;
10 dayCounter = 1;
11
12 while (dayCounter <= 7){
13     cout << "Enter temperature: ";
14     cin << temperature;
15     total = total + temperature;
16     dayCounter = dayCounter +1;
17 }
18 average = total/7;
19 cout << "Average temperature is " << average << endl;
20 return 0;
21 }
```

# Reading and writing files

- Reading and writing files is done using ifstream and ofstream imported from the fstream library. The following program reads numbers from a file (input.txt) and prints the sum to another file (output.txt).

[ C++ ] Reading and writing files - Part 1

```
1 #include <iostream> //for cout
2 #include <fstream> //for ifstream and ofstream
3
4 int main() {
5   std::ifstream inFile("input.txt"); //name of the file to read from
6   if(!inFile) {
7     std::cout << "Error: could not read from file input.txt" << std::endl;
8     return 1; //a non-zero return value indicates failure
9   }
```

# Reading and writing files

## [ C++ ] Reading and writing files - Part 2

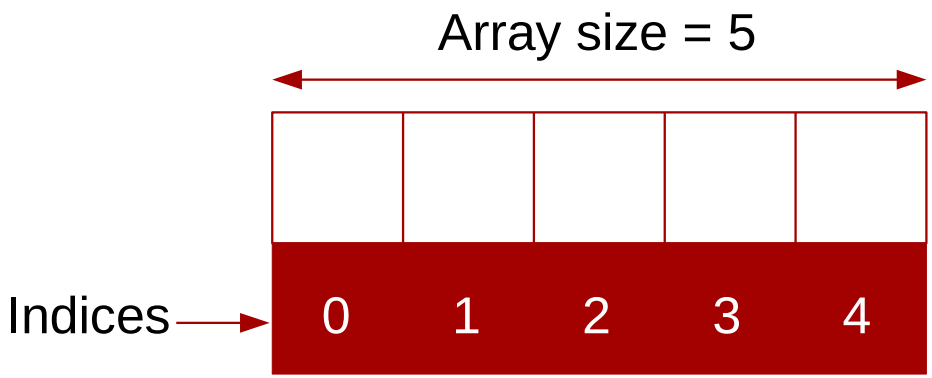
```
1   double variable = 0.;
2   double sum = 0.;
3   while(inFile >> variable) { //read number until we hit the end of the file
4     sum += variable
5   }
6   inFile.close();
7   std::ofstream outFile("output.txt");
8   if(!outFile) {
9     std::cout << "Error: could not write to file output.txt" << std::endl;
10    return 1; //a non-zero return value indicates failure
11  }
12  outFile << sum << std::endl;
13  outFile.close();
14  return 0;
15 }
```

# Containers: arrays

- A container is a holder object that stores a collection of objects (its elements). There are several types of containers including arrays, vectors and strings;
- Arrays are fixed-size sequence containers: they hold a specific number of elements ordered in a strict linear sequence;
- An array must be declared before it is used:

[ C++ ] Array declaration

```
1 type name [number of elements];
2 int arr [5];
```



# Containers: arrays

- The values of any of the elements in an array can be accessed using the following syntax:

[ C++ ] Arrays declaration

```
1 name_array[index]
2 arr [2] = 70;//the value 70 is stored in the THIRD element of the array
```

- multidimensional arrays can be defined as:

[ C++ ] multidimensional arrays

```
1 type name [size1][size2]...[sizeN]
2
3 int arr [3][5]; //bidimensional array example
```

- arrays allocated on the heap are deleted with the delete[] operator;
- try to avoid using arrays in C++; use vectors instead (see next lecture).

# Exercise 4

Write a C++ program that reads from an input file the temperatures of the week, calculates the average value and prints the result in an output file.
Hints:

- in the same folder as the source code, create a new file called input.txt;
- write the code, compile, run and check the result in output.txt.

# Exercise 4 - Part 1

[ C++ ]

```cpp
#include <iostream>
#include <fstream> //for ifstream and ofstream
using namespace std;

int main(){

std::ifstream inFile("input.txt"); //name of the file to read from
  if(!inFile) {
    std::cout << "Error: could not read from file input.txt" << std::endl;
    return 1; //a non-zero return value indicates failure
  }

double temperature = 0.; // temperature represents the values in the input file
double total = 0.; //sum of the temperatures
double average; //average of temperatures
int i = 0;
while(inFile >> temperature){
    total = total + temperature;
    i = i +1;
}
```

# Exercise 4 - Part 2

[ C++ ]

```
1
2 average = total/i;
3
4 inFile.close();
5   std::ofstream outFile("output.txt");
6   if(!outFile) {
7     std::cout << "Error: could not write to file output.txt" << std::endl;
8     return 1; //a non-zero return value indicates failure
9   }
10  outFile << average << std::endl;
11  outFile.close();
12  return 0;
13 }
```

Input file example:

[ C++ ]

```
1 12.5 13.4 11.7 15 16.9 15.7 12
```