Streams in C++

Introduction

Console input/output using streams in C++

- three streams are defined in C++ in `iostream`:
  - output stream (to the stdout) `cout`
  - input stream (from the stdin) `cin`
  - error stream (to the stderr) `cerr`
- two **overloaded operators** <<, >> are defined for input/output streams

- streams `cout, cin, cerr` are objects, *not commands* (i.e. three variables of "object type" (class) `ostream` and `istream`, already declared in libraries
  - functionality (printing) is "hidden" in overloading of operators <<, >> (i.e. function is "assigned" to operators << and >> in `ostream` and `istream` classes
- you can still use `printf, scanf`

- example:

```
#include <iostream>
int main(void)
{
   int cp,ca;
   cout << "Enter count of people and
  count of animals: ";
   cin >> cp >> ca;
   cout << "The count of people is "
  << cp <<
   ", the count of animals is " << ca
  << '.';
}
```

Help

output stream - data is transferred **to** the stream

cout << "Bye";

input stream - data is transferred **from** the stream

cin >> ca;

Manipulators

- are used to change formatting parameters on streams and to insert or extract certain special characters
- defined in `iomanip, iostream, ios`

```
// new line
cout << "Bye" << endl;
// hexadecimal output
cout << hex << x;
// hexadecimal input
cin >> hex >> a;
```

`dec`	decimal base
`hex`	hexadecimal base
`oct`	octal base
`endl`	new line + “flush”
`setw(int n)`	set field width to *n* chars
`setfill(int c)`	set *c as fill char*
`setprecision(int n)`	set decimal precision to *n* decimal positions
`showpos`	‘+’ sign is printed
`noshowpos`	+ sign is cancelled
`boolalpha`	prints true/false

- more in documentation

Namespaces

Namespaces

- example of collision, if both file with the same definitions file are included together

rail.h

```
const int x=10;
typedef enum
{
  STOP,CONT
} States;
```

logic.h

```
const int x=5;
typedef enum
{
  DIV,NOT_DIV
} States;
```

```
#include "rail.h"
#include "logic.h"
void main(void)
{

  States s1,s2;
  int a = x+3;
}
```

Which States?

from rail.h or
from logic.h

- solution: to nest definitions into *different namespaces*

rail.h

```
namespace Rail {
   const int x=10;
   typedef enum
   {
      STOP,CONT
   } States;
}
```

logic.h

```
namespace Logic {
   const int x=5;
  typedef enum
   {
      DEV,NOT_DIV
   } States;
}
```

- out of the namespace, identifiers are not accessible, the reference to the namespace must be stated

```
Logic::States
```

```
#include "rail.h"
#include "logic.h"

void main(void)
{
  Rail::States h1;
  Logic::States h2;
  int a = Logic::x+3;
}
```

- to repeat the same space-name in the code is a bit "boring" and non-effective
- it is possible to reach direct visibility of function/constants/ from some namespace with directive **`using`**

```
#include "rail.h"
#include "logic.h"

using namespace Rail;
void main(void)
{
  States h1;
  Logic::States h2;
  using namespace Logic;
  int a = x+3;
}
```

Notes

- namespace declarations can be nested

```
namespace Rail {

 const int x = 5;
 namespace Train {
    const int Train_Length = 500;
 }

 namespace ETCS {
    const int ETCS_Level = 1;
 }
}
```

- we put in the code

```
Rail::x
Rail::Train::Train_Length
Rail::ETCS::ETCS_Level
```

- namespace declarations are opened (they can be added)
  - the first .h file: **`namespace`** `A { … }`
  - the second .h file: **`namespace`** `A { … }`
- the predefined objects in header files, for example `cout,` are defined in `std` namespace
  - don't forgot put **`using namespace`** `std;`

- names can be abbreviated with aliases:
  - `namespace Log=Logic;`

Try...

```
#include <iostream>
#include <iomanip>
int main(int argc, char **argv)
{ int x;
  cout << "Hello, world!" << endl;
  cout << hex << 65 << endl;
  cout << "Enter number in octal radix: ";
  cin >> oct >> x;
  cout << dec << x << endl;
  cout << showpos << x << endl;
  cout << setw(5) << setfill('0') << x << endl;
  cout << boolalpha << true;
  return 0;
}
```

- the compiler doesn't compile this code (the error *'cout' was not declared in this scope* occurs)

Try…

```
#include <iostream>
#include <iomanip>
int main(int argc, char **argv)
{ int x;
  std::cout << "Hello, world!" << std::endl;
  std::cout << std::hex << 65 << std::endl;
  …
  return 0;
}
```

Try...

```
#include <iostream>
#include <iomanip>

using namespace std;
int main(int argc, char **argv)
{ int x;
  cout << "Hello, world!" << endl;
  cout << hex << 65 << endl;
  cout << "Enter number in octal radix: ";
  cin >> oct >> x;
  cout << dec << x << endl;
  cout << showpos << x << endl;
  cout << setw(5) << setfill('0') << x << endl;
  cout << boolalpha << true;
  return 0;
}
```