Statements and expressions

Victor Eijkhout and Charlie Dey

spring 2017

Basics

TACC

Edit, compile, run

Exercise: make a file zero.cc with the following lines:

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

int main() {
  return 0;
}
```

and compile it. Intel compiler:

```
icpc -o zeroprogram zero.cc
```

Alternative Gnu:

```
g++ -o zeroprogram zero.cc
```

Run this program: ./zeroprogram
It doesn't do anything.

TACC

Add this line:

```
cout << "Hello world!" << endl;
```

(copying from the pdf file is dangerous! please type it yourself)

Compile and run again.

TACC

Statements

Program statements

- A program contains statements, each terminated by a semicolon.
- 'Curly braces' can enclose multiple statements.
- A statement can correspond to some action when the program is executed.
- Some statements are definitions, of data or of possible actions.
- Comments are 'Note to self', short:

  ```
  cout << "Hello world" << endl; // say hi!
  ```

  and arbitrary:

  ```
  cout << /* we are now going
              to say hello
          */ "Hello!" << /* with newline */ endl;
  ```

TACC

Exercise 1

Take the 'hello world' program you wrote earlier, and duplicate the hello-line. Compile and run.

Does it make a difference whether you have the two hellos on the same line or on different lines?

Experiment with other changes to the layout of your source. Find at least one change that leads to a compiler error.

TACC

Fixed elements

You see that certain parts of your program are inviolable:

- There are *keywords* such as return or cout; you can not change their definition.
- Curly braces and parentheses need to be matched.
- There has to be a main keyword.
- The iostream and namespace are usually needed.

TACC

Exercise 2

Experiment with the cout statement. Replace the string by a number or a mathematical expression. Can you guess how to print more than one thing, for instance the string `One third is` and the result of $1/3$, with the same cout statement?

TACC

Variables

Typical variable lifetime

```
int i,j; // declaration
i = 5; // set a value
i = 6; // set a new value
j = i+1; // use the value of i
i = 7; // change the value of i
       // but this doesn't affect j
```

TACC

Variable declarations

Programs usually contain data, which is stored in a *variable*.
A variable has

- a *datatype*,
- a name, and
- a value.

These are defined in a *variable declaration* and/or *variable assignment*.

TACC

Variable names

- A variable name has to start with a letter,
- can contains letters and digits, but not most special characters (except for the underscore).
- For letters it matters whether you use upper or lowercase: the language is *case sensitive*.

TACC

Declaration

There are a couple of ways to make the connection between a
name and a type. Here is a simple *variable declaration*, which
establishes the name and the type:

```
int n;
float x;
int n1,n2;
double re_part,im_part;
```

Declarations can go pretty much anywhere in your program, but
need to come before use of the variable.

TACC

Assignments

Datatypes

Variables come in different types;

- We call a variable of type `int,float,double` a *numerical variable*.
- For characters: `char`. Strings are complicated.
- You can make your own types. Later.

TACC

Assignment

Once you have declared a variable, you need to establish a value.
This is done in an *assignment* statement. After the above
declarations, the following are legitimate assignments:

```
n = 3;
x = 1.5;
n1 = 7; n2 = n1 * 3;
```

You see that you can assign both a simple value or an *expression*;
see section **??** for more detail.

TACC

Assignments

A variable can be given a value more than once. You the following sequence of statements is a legitimate part of a program:

```
int n;
n = 3;
n = 2*n + 5;
n = 3*n + 7;
```

Special forms

Update:

```
x = x+2; y = y/3;
// can be written as
x += 2; y /= 3;
```

Integer add/subtract one:

```
i++; j--;  /* same as: */ i=i+1; j=j-1;
```

Pre/post increment:

```
x = a[i++]; /* is */ x = a[i]; i++;
y = b[++i]; /* is */ i++; y = b[i];
```

TACC

Exercise 3

Write a program that has several variables. Assign values either in an initialization or in an assignment. Print out the values.

TACC

Initialization

You can also give a variable a value a in *variable initialization*.
Confusingly, there are several ways of doing that. Here's two:

```
int n = 0;
double x = 5.3, y = 6.7;
double pi{3.14};
```

Exercise 4

Print out `true` and `false`. What do you get?

TACC

Input/Output

Floating point constants

- Default: `double`
- Float: `3.14f` or `3.14F`
- Long double: `1.41l` or `1.41L`.

This prevents numerical accidents:

```
double x = 3.;
```

converts float to double, maybe introducing random bits.

TACC

Truth values

So far you have seen integer and real variables. There are also *boolean values* which represent truth values. There are only two values: true and false.

```
bool found{false};
```

Terminal output

You have already seen cout:

```
float x = 5;
cout << "Here is the root: " << sqrt(x) << endl;
```

TACC

Exercise 5

Write a program that

- Displays the message Type a number,
- accepts an integer number from you (use cin),
- and then prints out three times that number plus one.

TACC

Turn it in!

- If you have compiled your program, do:
  `test3np1 yourprogram`
  where 'yourprogram' stands for the name of your program.
- Is it reporting that your program is correct? If so, do:
  `test3np1 -s yourprogram`
  where the -s flag stands for 'submit'.

TACC

Expressions

Terminal input

There is also a *cin*, which serves to take user input and put it in a numerical variable.

```
int i;
cin >> i;
```

However, this function is somewhat tricky.
http://www.cplusplus.com/forum/articles/6046/.

TACC

Better terminal input

It is better to use `getline`. This returns a string, rather than a
value, so you need to convert it with the following bit of magic:

```
#include <iostream>
#include <sstream>
using namespace std;
  /* ... */
  std::string saymany;
  int howmany;

  cout << "How many times? ";
  getline( cin,saymany );
  stringstream saidmany(saymany);
  saidmany >> howmany;
```

You can not use `cin` and `getline` in the same program.

TACC

Arithmetic expressions

- Expression looks pretty much like in math.
  With integers: 2+3
  with reals: 3.2/7
- Use parentheses to group 25.1*(37+42/3.)
- Careful with types.
- There is no 'power' operator: library functions. Needs a line
  `#include <cmath>`
- Modulus: %

TACC

Exercise 6

- What happens when you assign a positive floating point value to an integer variable?
- What happens when you assign a negative floating point value to an integer variable?
- What happens when you assign a `float` to a `double`? Try various number for the original float. Can you explain the result? (Hint: think about the conversion between binary and decimal.)

TACC

Exercise 7

Write two programs, one that reads a temperature in Centigrade
and converts to Fahrenheit, and one that does the opposite
conversion.

$$C = (F - 32) \cdot 5/9, \qquad F = 9/5\, C + 32$$

Can you use Unix pipes to make one accept the output of the
other?

TACC