Introduction to Programming

Part I

The Introductory Class

Mansur Khazeev
Eugene Zouev
Fall Semester 2021
Innopolis University

Introduction to Programming

Why the Course?

Introduction to Programming

Why the Course?

- **Programming** is the fundamental skill in computer science – whatever area you choose in your professional career.
- A professional should know several programming languages…
- …Moreover: (s)he should be able to quickly learn any new language, software technology or a framework…
- And for that, you should know basic concepts that are common to many (if not all) programming languages: type, algorithm, control flow, expressions/statements, syntax & semantics, software lifecycle, OOP, and many other.

Organization

Contents

- **Lectures**:
  Theory, general stuff.
  Language concepts will be presented first
- **Tutorials**:
  Extra stuff. Examples to illustrate what was presented during the lecture + particular aspects
- **Labs**:
  Allow you to get practcal experince in programming

Organization

Contents

- **Lectures**:
  Theory, general stuff.
  Language concepts will be presented first
- **Tutorials**:
  Extra stuff. Examples to illustrate what was presented during the lecture + particular aspects
- **Labs**:
  Allow you to get practcal experince in programming

Moodle

- ***All*** information will be on Moodle (http://moodle.innopolis.university)
- There you will find:
  - the lecture material, just after the class (sometime before)
  - the lab sessions with exercises and information about projects and assignments
- Plus any other information and all your grades

Exams, Evaluation & Grading

The Schedule

- Lectures: each Friday
- Tutorials & Labs: each Monday

Examinations

- Assignments & projects:
  to be evaluated each 1-2 week
- Mid-term examination:
  written form (quiz; ~13th Oct.)
- Final exam:
  written form (program tasks)

Exams, Evaluation & Grading

Up to 5 extra points

The Schedule

- Lectures: each Friday
- Tutorials & Labs: each Monday

Examinations

- Assignments & projects:
  to be evaluated each 1-2 week
- Mid-term examination:
  written form (quiz; ~13th Oct.)
- Final exam:
  written form (program tasks)

Assessment

- Mid-term Exam (25%),
- Final Exam (30%)
- Lab assignments (40%)
- Lab attendance (5%)

Grading

- **A** [90, 100]
- **B** [75, 90)
- **C** [60, 75)
- **D** [0, 60)

Required Background & Workload

The course is intended to be self-contained, requiring basic knowledge of math including binary calculus and common sense ☺

The will to learn is a key prerequisite !

Overall the course should take on average **12 hours per week** of your life ☺

Prof M. Mazzara: only 50% of material will be given on lectures/tutorials; the other is the matter of **your own study**

The Overall Structure of the Course

Three main parts of the course

- **The C language**

  Small, system-level (but still general-purpose) language

The Overall Structure of the Course

Three main parts of the course

The Fall semester

- **The C language**
  Small, system-level (but still general-purpose) language
- **The Java language**
  Powerful application language

The Overall Structure of the Course

Three main parts of the course

The Fall semester

- **The C language**

  Small, system-level (but still general-purpose) language

- **The Java language**

  Powerful application language

The Spring semester

- **The C++ language**

  Fast and powerful general-purpose language **with deep semantics**

The Teaching Team

- **Eugene Zouev** lectures
- **Mansur Khazeev** tutorials
- **Sami Sellami**
  Alex Shikulin
  Munir Makhmutov
  teaching assistants

Who Is This Guy? ☺

- **Eugene Zouev**
- Have been working at Moscow Univ.,
  Swiss Fed Inst of Technology (ETH Zürich),
  EPFL (Lausanne); PhD (1999, Moscow Univ).
- Prof. interests:
  compiler construction, language design, PL semantics.
- The author of the 1st Russian C++ front-end compiler
  - Interstron Ltd., Moscow, 1999-2000.
- Zonnon language implementation for .NET & Visual Studio
  - ETH Zürich, 2005.
- Swift prototype compiler for Tizen & Android
  - Samsung Research, 2015
- Six (or seven? ☺) books; the latest are
  - «Редкая профессия», ДМК Пресс, Москва 2014.
  - Software Design for Resilient Computer Systems, Springer, 2019

I'm sure you have some experience in **practical** programming.
But do you really understand (and can explain) **notions** used in your code?

Actually, you don't ☺

Actually, you cannot ☹

Before we start…

A remark about language syntax & semantics

Syntax:
A set of rules that regulate
the structure of programs
and their parts (constructs)

Before we start…

A remark about language syntax & semantics

Syntax:
A set of rules that regulate
the structure of programs
and their parts (constructs)

Semantics:
The **meaning** of the constructs

Static semantics:
- How programs get compiled

Dynamic semantics:
- How programs get executed.

Before we start…

A remark about language syntax & semantics

"Usual" view at a language:

Syntax:
A set of rules that regulate **the structure** of programs and their parts (constructs)

Semantics:
The **meaning** of the constructs

Static semantics:
- How programs get compiled

Dynamic semantics:
- How programs get executed.

Before we start…

A remark about language syntax & semantics

"Usual" view at a language:

Syntax:
A set of rules that regulate **the structure** of programs and their parts (constructs)

Semantics:
The **meaning** of the constructs

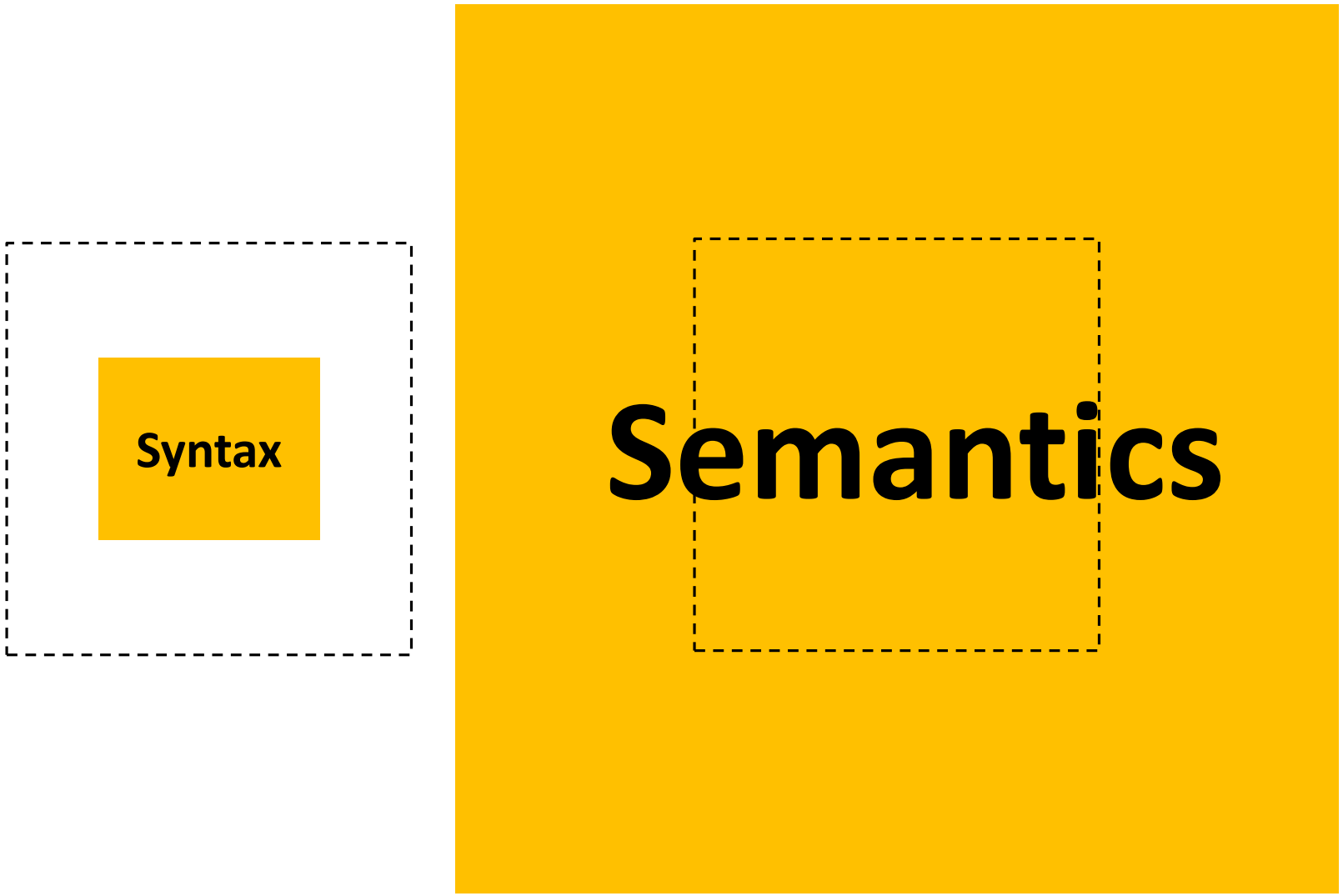
Static semantics:
- How programs get compiled

Dynamic semantics:
- How programs get executed.

Before we start…

A remark about language syntax & semantics

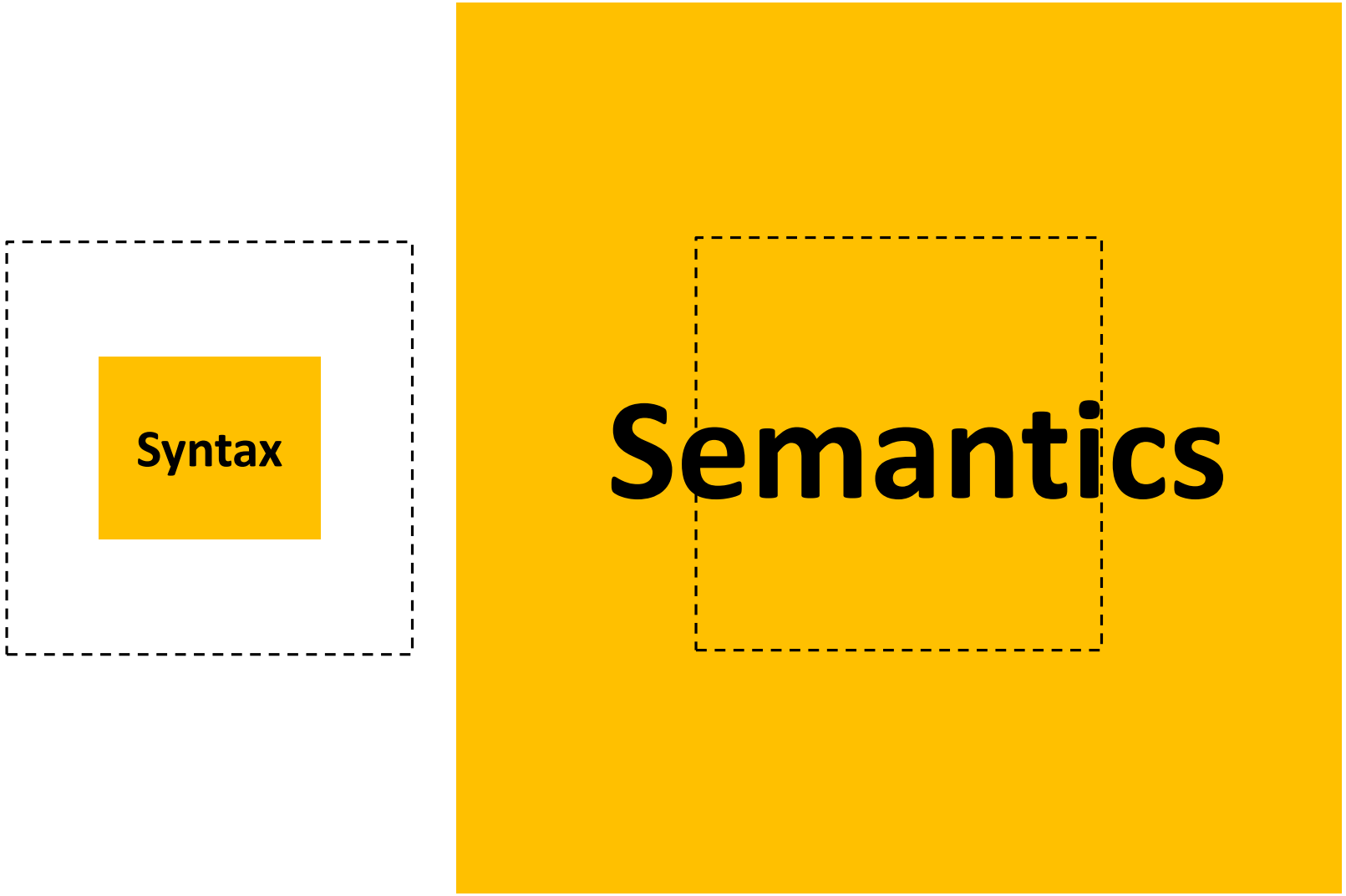
Reality:



Before we start…

A remark about language syntax & semantics

Reality:



Conclusion for programmers:

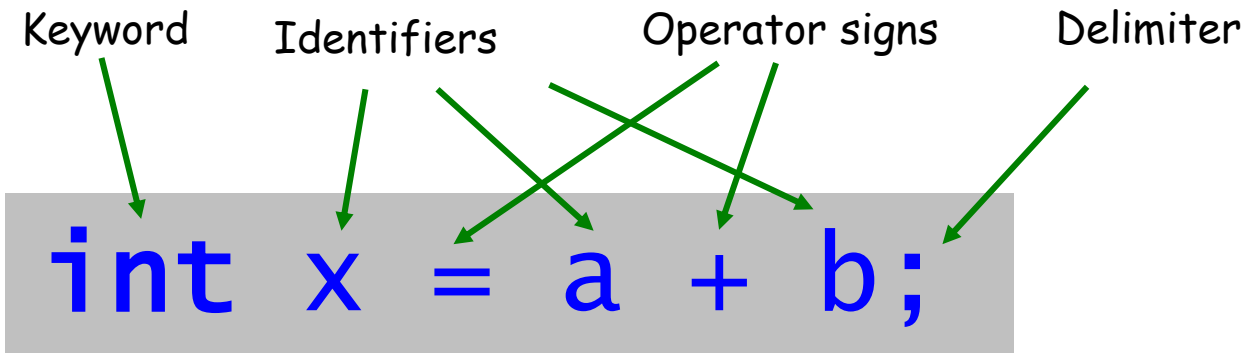
- Pay most attention on the language semantics rather then on syntax

C: Syntax vs Semantics

```
int x = a + b;
```

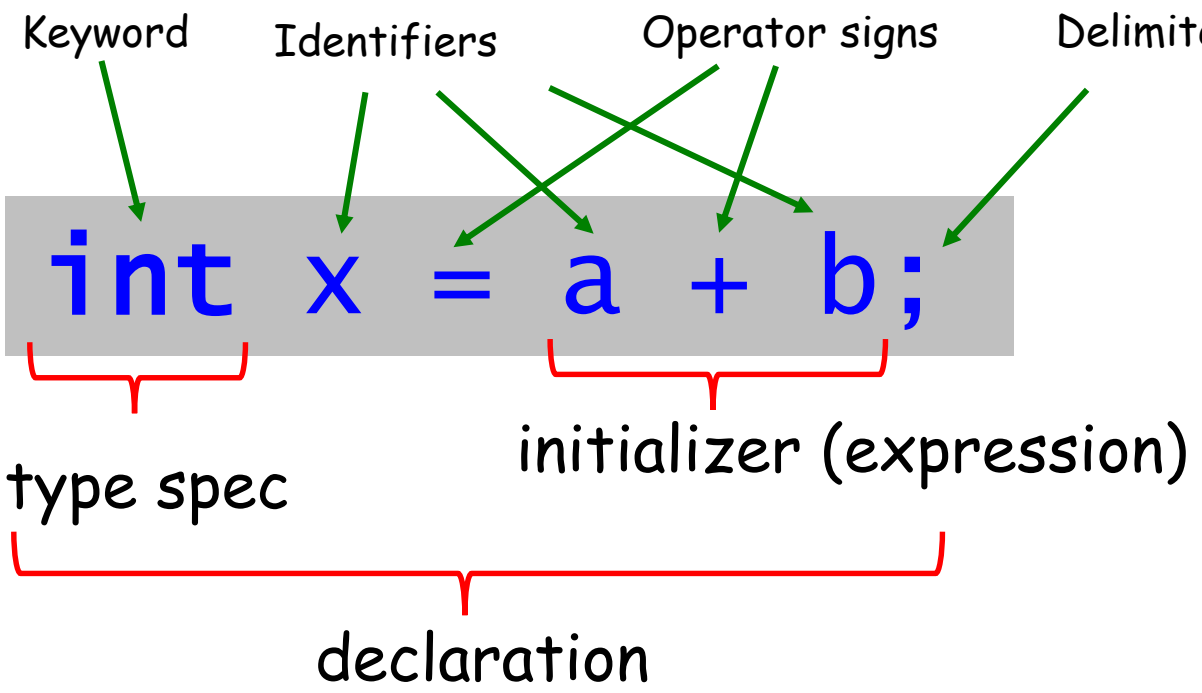
C: Syntax vs Semantics

Lexics



C: Syntax vs Semantics

Lexics



Syntax

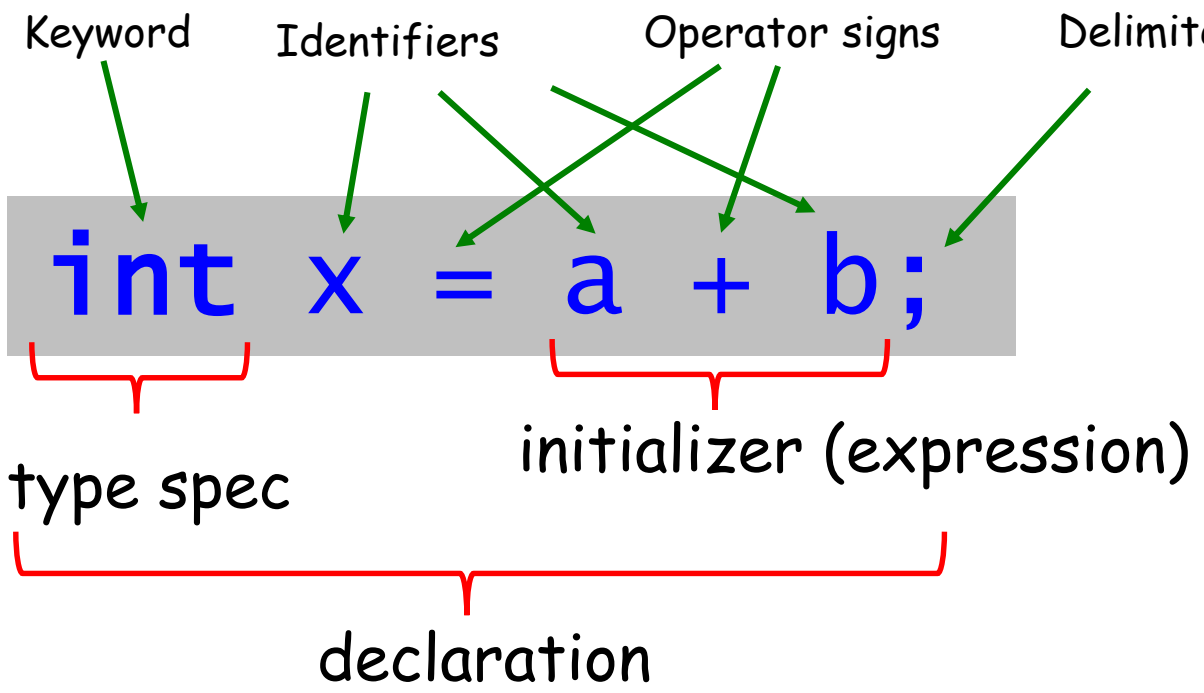
type spec

initializer (expression)

declaration

C: Syntax vs Semantics

Lexics



Syntax

type spec

initializer (expression)

declaration

Semantics

- Allocate memory for the new integer variable (in stack)
- Calculate (the value of) the expression from initializer
- Perform type conversion(s) to integer, if necessary
- Store the value of the expression
- Make x available in the current context

Program Lifecycle: Compilation

Program source text

```
int main()
{
  Stack<double> stack1;
  Stack<int> stack2(5);
  int y = 1;
  double x = 1.1;
  int i, j;
  cout << "\n pushed values into stack1: ";
  for ( i=1; i<=11; i++)
  {
    if (stack1.push(i*x))
      cout << endl << i*x;
    else
      cout << "\n stack1 is full";
  }
  cout << "\n\n popd values from stack1:\n";
  for (i=1; i<=6; i++)
    cout << stack1.pop() << endl;
  ...
```

Is this a program? ☺

Program Lifecycle: Compilation

Program source text

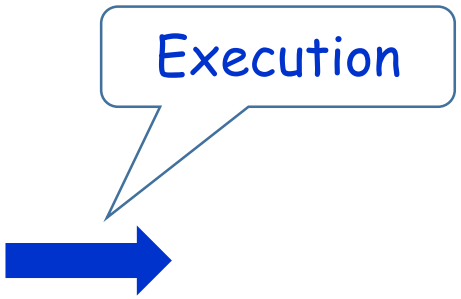
```
int main()
{
  Stack<double> stack1;
  Stack<int> stack2(5);
  int y = 1;
  double x = 1.1;
  int i, j;
  cout << "\n pushed values into stack1: ";
  for ( i=1; i<=11; i++)
  {
    if (stack1.push(i*x))
      cout << endl << i*x;
    else
      cout << "\n stack1 is full";
  }
  cout << "\n\n popd values from stack1:\n";
  for (i=1; i<=6; i++)
    cout << stack1.pop() << endl;
  ...
```

Is this a program? ☺

- **No**: this is just a text

Machine code

```
0x006 77 22378EE
0x007 00 0000001
0x008 33 1017700
0x009 7B 00178AB
0x00A 7B 00178AB
0x00B 72 037CEFF
0x00C 3D AFFFFED
0x00E 72 037CEFF
0x00F 3D AFFFFED
0x00D 7B 00178AB
0x00E 3D CAFEBEB
0x00F 3D 00011FF
...
```

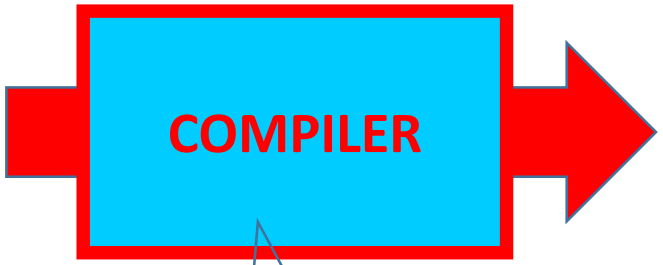


This is a program

Program Lifecycle: Compilation

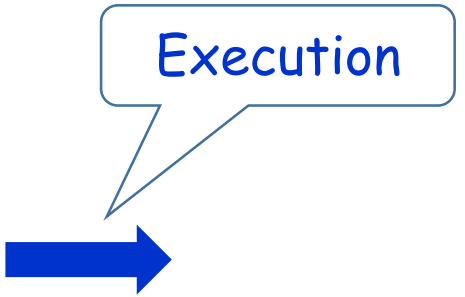
Program source text

```
int main()
{
  Stack<double> stack1;
  Stack<int> stack2(5);
  int y = 1;
  double x = 1.1;
  int i, j;
  cout << "\n pushed values into stack1: ";
  for ( i=1; i<=11; i++)
  {
    if (stack1.push(i*x))
      cout << endl << i*x;
    else
      cout << "\n stack1 is full";
  }
  cout << "\n\n popd values from stack1:\n";
  for (i=1; i<=6; i++)
    cout << stack1.pop() << endl;
  ...
```



Machine code

```
0x006 77 22378EE
0x007 00 0000001
0x008 33 1017700
0x009 7B 00178AB
0x00A 7B 00178AB
0x00B 72 037CEFF
0x00C 3D AFFFFED
0x00E 72 037CEFF
0x00F 3D AFFFFED
0x00D 7B 00178AB
0x00E 3D CAFEBEB
0x00F 3D 00011FF
...
```



Is this a program? ☺
- **No**: this is just a text

This is a program

Compiler transforms the program text into a semantically equivalent sequence of machine instructions

We will consider **other kinds** of program lifecycles later

Questions?