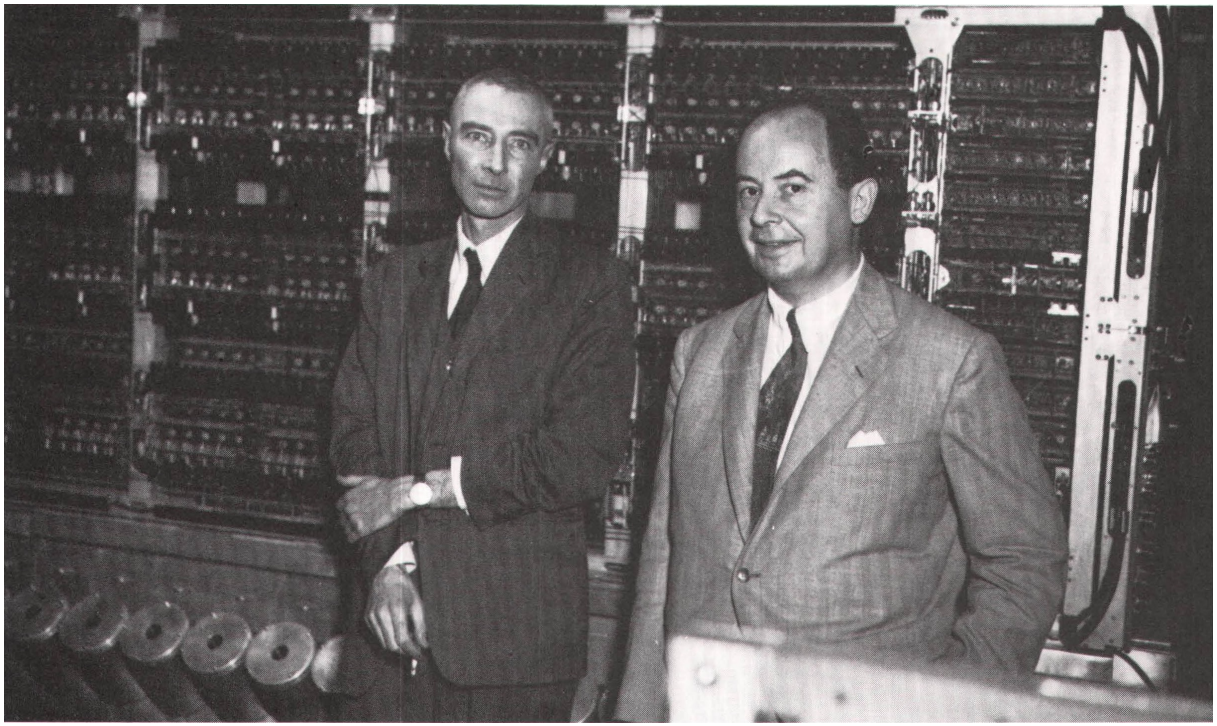
Computational thinking

Victor Eijkhout and Charlie Dey

spring 2017

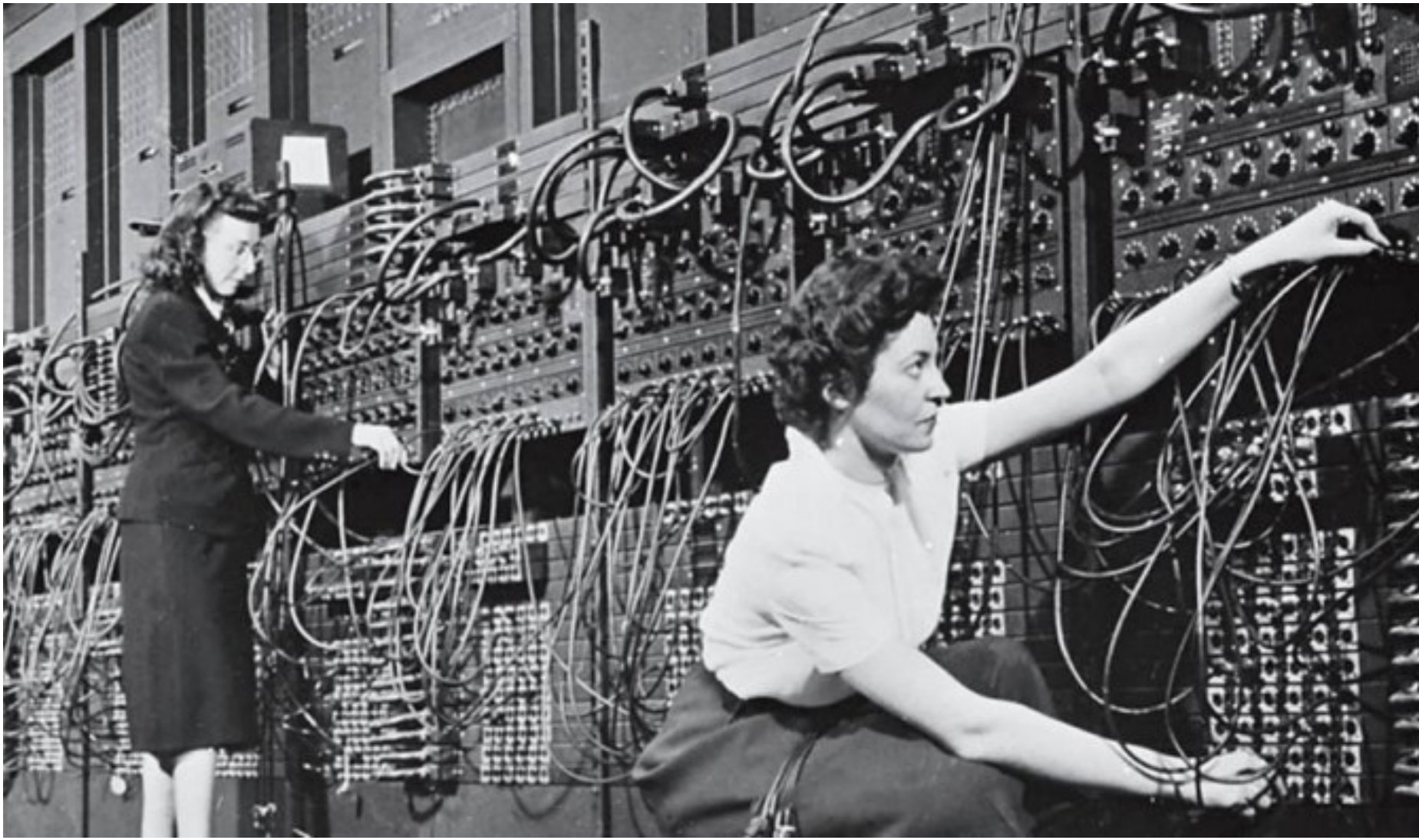
Earliest computers

Historically, computers were used for big physics calculations, for instance, atom bomb calculations



Hands-on programming

Very early computers were hardwired



Program entry

Later programs were written on punchcards

The first programming language

Initial programming was about translating the math formulas; after a while they made a language for that: FORmula TRANslation

Programming is everywhere

Programming is used in many different ways these days.

- You can make your own commands in *Microsoft Word*.
- You can make apps for your *smartphone*.
- You can solve the riddles of the universe using big computers.

This course is aimed at people in the last category.

Programming is not simple

Programs can get pretty big

Examples of computational thinking

- Looking up a name in the phone book
  - start on page 1, then try page 2, et cetera
  - or start in the middle, continue with one of the halves.
- Elevator scheduling: someone at ground level presses the button, there are cars on floors 5 and 10; which one do you send down?

Abstraction

- The elevator programmer probably thinks: ‘if the button is pressed’, not ‘if the voltage on that wire is 5 Volt’.
- The Google car programmer probably writes: ‘if the car before me slows down’, not ‘if I see the image of the car growing’.
- . . . but probably another programmer had to write that translation.

Data abstraction

What is the structure of the data in your program?

Stack: you can only get at the top item

items get added in the back, processed at the front

Queue:

Do you have to know much about hardware?

Yes, it's there, but we don't think too much about it in this course.

`https://youtu.be/JEpsKnWZrJ8`

What is an algorithm?

An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time

[A. Levitin, Introduction to The Design and Analysis of Algorithms, Addison-Wesley, 2003]

The instructions are in some language:

- We will teach you C++ and Fortran;
- the compiler translates those languages to machine language
- Abstraction: a program often defines its own language that implements concepts of your application.

Program steps

- Simple instructions: arithmetic.
- Compicated instructions: control structures
  - conditionals
  - loops

Program data

- Input and output data: to/from file, user input, screen output, graphics.
- Data during the program run:
  - Simple variables: character, integer, floating point
  - Arrays: indexed set of characters and such
  - Data structures: trees, queues
    - Defined by the user, specific for the application
    - Found in a library (big difference between C/C++!)

Comparing two languages

Python vs C++ on bubblesort:

```
for i in range(n-1):
  for j in range(n-i-1):
    if numbers[j+1]<numbers[j]:
      swap = numbers[j+1]
      numbers[j+1] = numbers[j]
      numbers[j] = swap
```

```
for (int i=0; i<n-1; i++)
  for (int j=0; j<n-1-i; j++)
    if (numbers[j+1]<numbers[j])
      int swap = numbers[j+1];
      numbers[j+1] = numbers[j];
      numbers[j] = swap;
  }
```

```
[] python bubblesort.py 5000
Elapsed time: 12.1030311584
[] ./bubblesort 5000
Elapsed time: 0.24121
```

The right language is not all

Python with quicksort algorithm:

```
numpy.sort(numbers,kind='quicksort')
```

```
[] python arraysort.py 5000
Elapsed time: 0.00210881233215
```