History of Programming

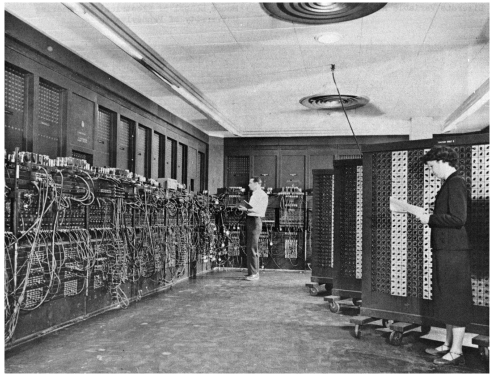
Markku Reunanen, Computational Art and Design
(never put dates on lecture slides)

The first programmer?

Ada Lovelace ~1843, Babbage's *Analytical Engine*

Mostly a matter of definition, as these firsties are

Digital computers



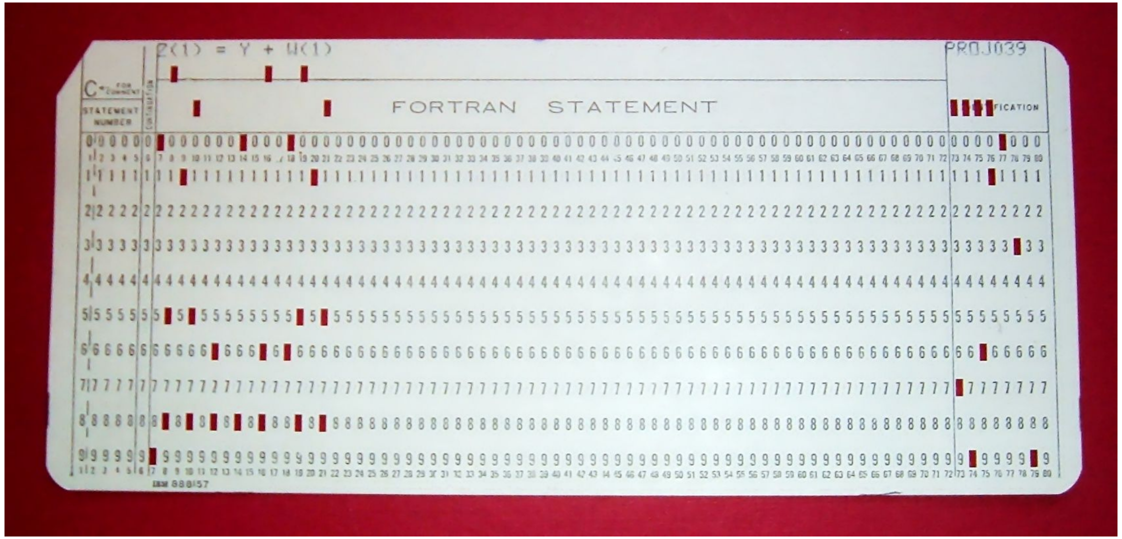
Digital computers appeared in the late 1940s and early 1950s

Initially large and crude machines

- Hardwired functionality
- Hand-coded machine language
- "Programming" might mean rewiring the machine
- Flicking switches and punch cards

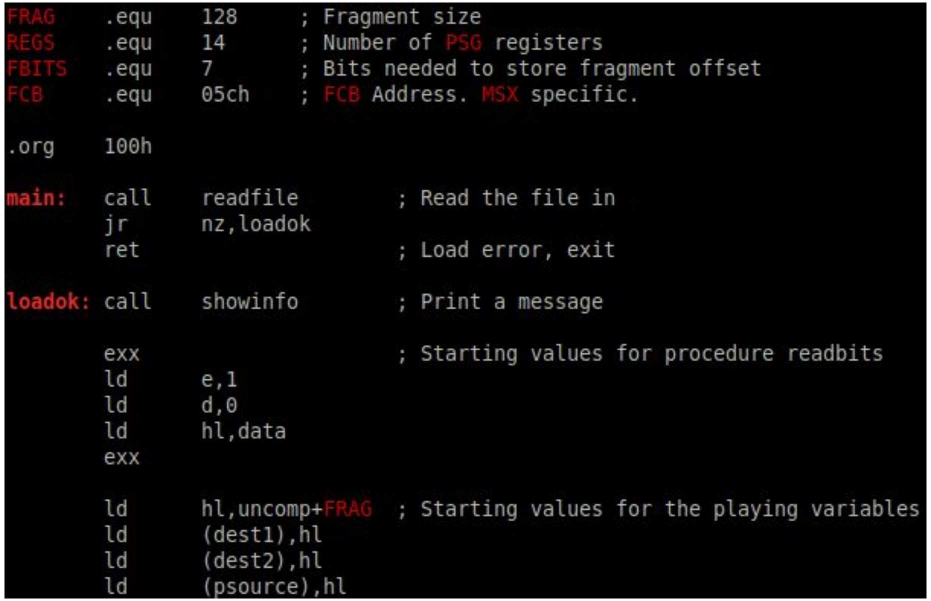
The space race (1950s–) and SAGE (1958–)

Punch cards



Give a stack to the operator, get the results later

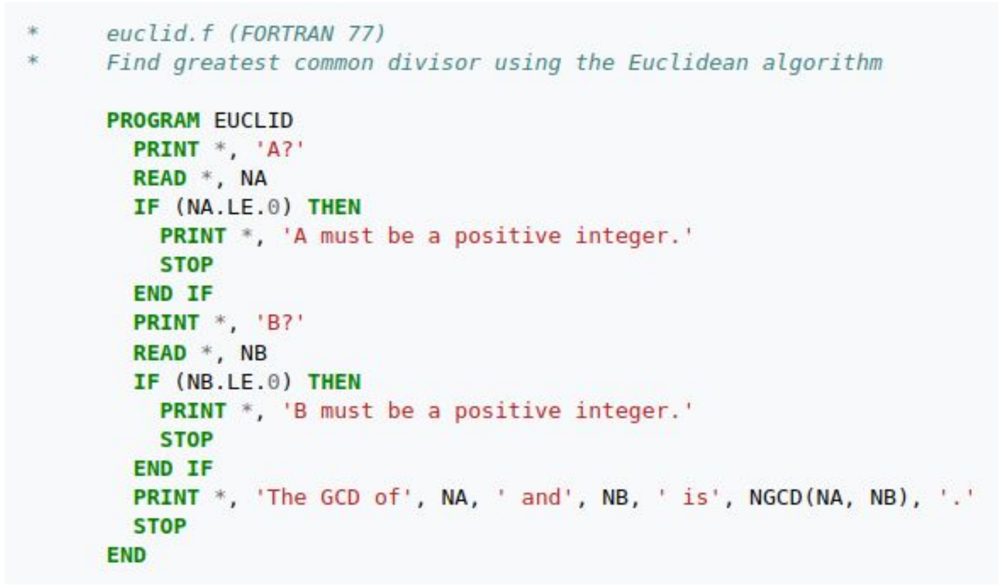
Assembly language



```
FRAG    .equ    128     ; Fragment size
REGS    .equ    14      ; Number of PSG registers
FBITS   .equ    7       ; Bits needed to store fragment offset
FCB     .equ    05ch    ; FCB Address. MSX specific.

.org    100h

main:   call    readfile        ; Read the file in
        jr      nz,loadok
        ret                     ; Load error, exit

loadok: call    showinfo        ; Print a message

        exx                     ; Starting values for procedure readbits
        ld      e,1
        ld      d,0
        ld      hl,data
        exx

        ld      hl,uncomp+FRAG  ; Starting values for the playing variables
        ld      (dest1),hl
        ld      (dest2),hl
        ld      (psource),hl
```

Human-readable machine language, *low-level language*

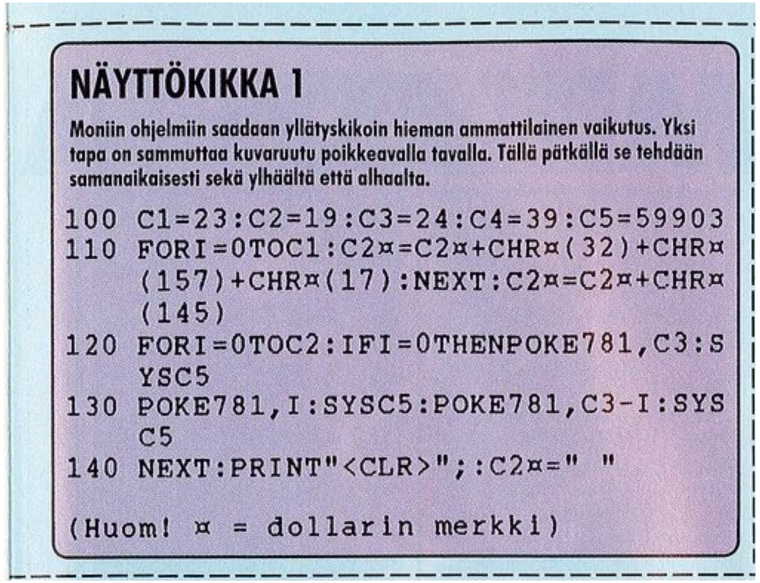
FORTRAN (~1957) by John Backus



```
*     euclid.f (FORTRAN 77)
*     Find greatest common divisor using the Euclidean algorithm

      PROGRAM EUCLID
        PRINT *, 'A?'
        READ *, NA
        IF (NA.LE.0) THEN
          PRINT *, 'A must be a positive integer.'
          STOP
        END IF
        PRINT *, 'B?'
        READ *, NB
        IF (NB.LE.0) THEN
          PRINT *, 'B must be a positive integer.'
          STOP
        END IF
        PRINT *, 'The GCD of', NA, ' and', NB, ' is', NGCD(NA, NB), '.'
        STOP
      END
```

Still used in scientific computing, *high-level language*

BASIC (1964)



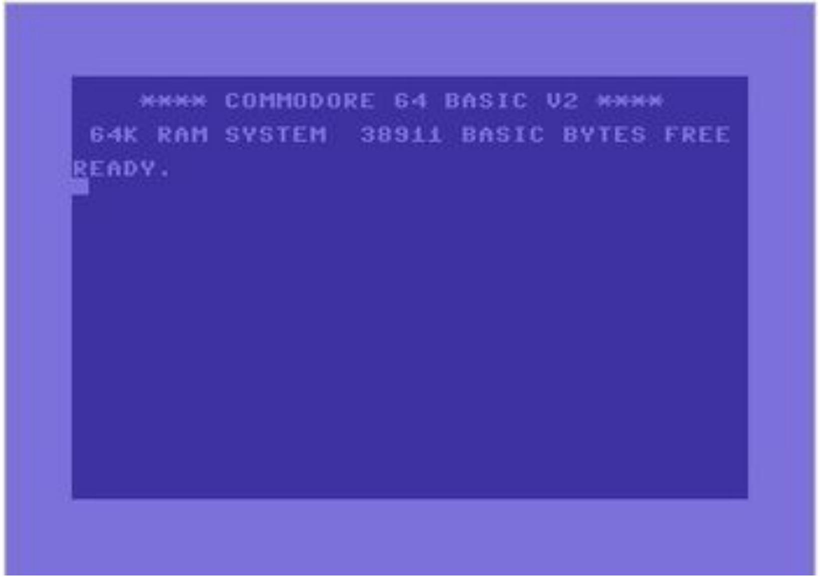
NÄYTTÖKIKKA 1

Moniin ohjelmiin saadaan yllätyskikoin hieman ammattilainen vaikutus. Yksi tapa on sammuttaa kuvaruutu poikkeavalla tavalla. Tällä pätkällä se tehdään samanaikaisesti sekä ylhäältä että alhaalta.

```
100 C1=23:C2=19:C3=24:C4=39:C5=59903
110 FORI=0TOC1:C2¤=C2¤+CHR¤(32)+CHR¤
    (157)+CHR¤(17):NEXT:C2¤=C2¤+CHR¤
    (145)
120 FORI=0TOC2:IFI=0THENPOKE781,C3:S
    YSC5
130 POKE781,I:SYSC5:POKE781,C3-I:SYS
    C5
140 NEXT:PRINT"<CLR>";:C2¤=" "

(Huom! ¤ = dollarin merkki)
```

Evolved from Fortran with a beginner-friendly intent

Commodore 64 (1982) boot screen



```
    **** COMMODORE 64 BASIC V2 ****
 64K RAM SYSTEM  38911 BASIC BYTES FREE
READY.
```

BASIC as the user interface for early home computers

Compilers and interpreters

Generally, a *compiler* turns human-readable source code into machine language

- Compilation needed, fast outcome, platform-specific

Generally, an *interpreter* runs human-readable source code directly

- No need to compile, interactive debugging, slower outcome

The line is blurred by *just-in-time* (JIT) compilers and *bytecode*

New paradigms

1960s and 1970s, procedural languages:

- Pascal (1970), still in use as Embarcadero and Free Pascal
- C (1972), still in wide use, the forefather of C++, C#, Java and several others

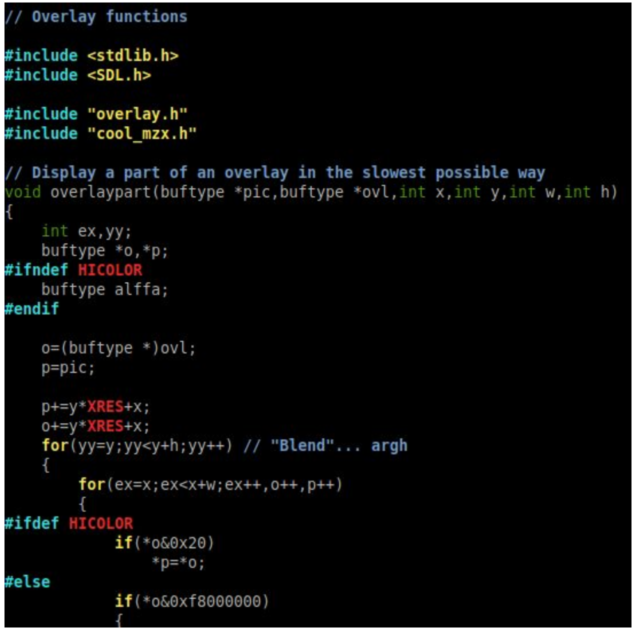
The advent of object-oriented languages:

- Smalltalk (1972) by Alan Kay
- C++ (1985) by Bjarne Stroustrup
- Java (1995), also Processing (2001)

Ken Thompson & Dennis M. Ritchie

Creators of Unix (~1970), Ritchie also C language

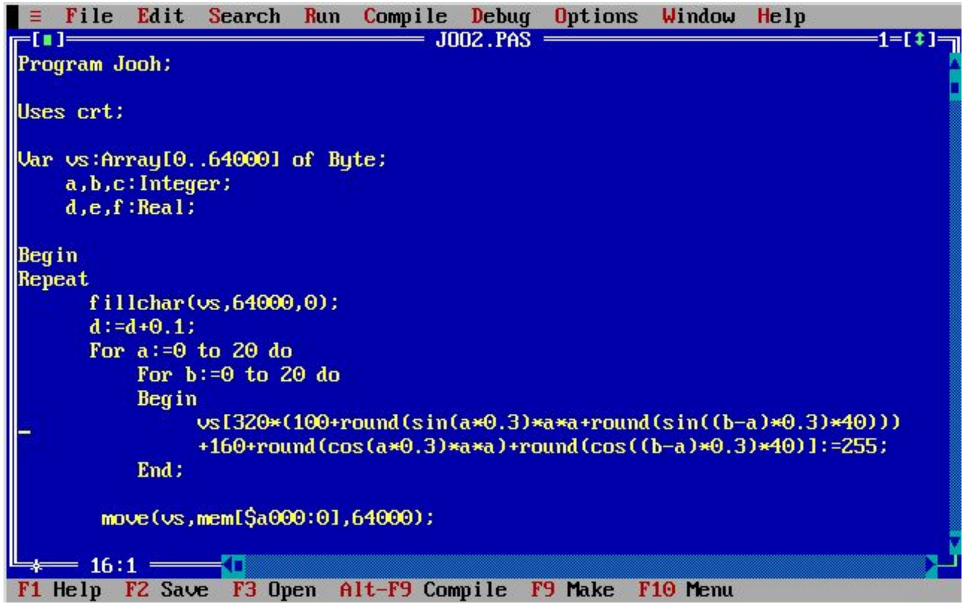
An example in C



```c
// Overlay functions

#include <stdlib.h>
#include <SDL.h>

#include "overlay.h"
#include "cool_mzx.h"

// Display a part of an overlay in the slowest possible way
void overlaypart(buftype *pic,buftype *ovl,int x,int y,int w,int h)
{
    int ex,yy;
    buftype *o,*p;
#ifndef HICOLOR
    buftype alffa;
#endif

    o=(buftype *)ovl;
    p=pic;

    p+=y*XRES+x;
    o+=y*XRES+x;
    for(yy=y;yy<y+h;yy++) // "Blend"... argh
    {
        for(ex=x;ex<x+w;ex++,o++,p++)
        {
#ifdef HICOLOR
            if(*o&0x20)
                *p=*o;
#else
            if(*o&0xf8000000)
            {
```

Should look familiar if you know Java, C++ or even JavaScript

An example in Pascal


≡ File Edit Search Run Compile Debug Options Window Help
JOO2.PAS
16:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
Program Jooh;

Uses crt;

Var vs:Array[0..64000] of Byte;
    a,b,c:Integer;
    d,e,f:Real;

Begin
Repeat
      fillchar(vs,64000,0);
      d:=d+0.1;
      For a:=0 to 20 do
          For b:=0 to 20 do
          Begin
               vs[320*(100+round(sin(a*0.3)*a*a+round(sin((b-a)*0.3)*40)))
               +160+round(cos(a*0.3)*a*a)+round(cos((b-a)*0.3)*40)]:=255;
          End;

       move(vs,mem[$a000:0],64000);
```

Conceptually close to C, here *Turbo Pascal*

Teletype

Moving toward interactive programming (1800s to 1960s)

Terminals (1960s onward)

Here a DEC VT52 from 1975

Changing computer market

First *minicomputers* (1960s) and then *microcomputers* (1970s) followed *mainframes*

Personal computers scarce until the late 1970s and the microcomputer revolution

Computing resources largely shared between multiple simultaneous users with "dumb" terminals

Compare to the Web, GeForce NOW and Aalto VDI

1990s and the Web (~1989)

The Web eventually became an important *application platform*

- JavaScript (1995)
- Macromedia Flash (1996), ActionScript (1998)
- PHP (1995)
- Java applets

Frontend vs. backend, many more used for the backend:
Python, Java, JavaScript, Ruby …

Other notable appearances

Logo (1968) for kids, developed by Wally Feurzeig & Seymour Papert. Turtle graphics.

Forth (1970), stack-based language

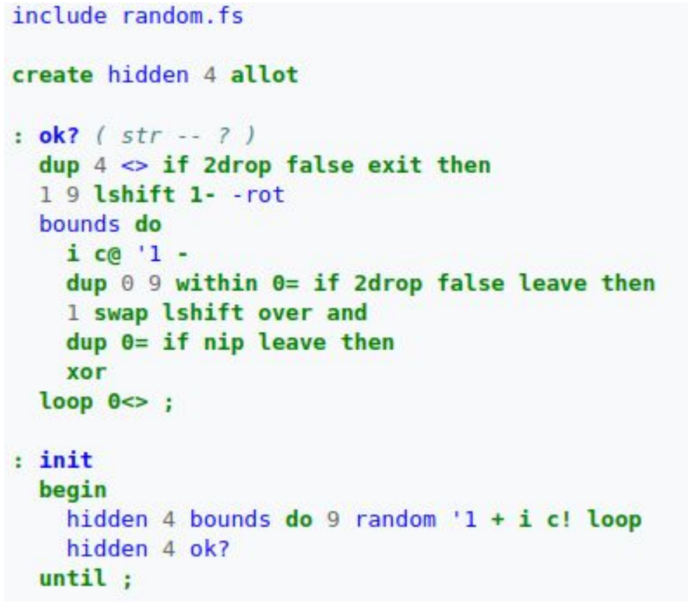
Max (mid-1980s), *Pure Data* (mid-1990s), visual programming

C# (2001), Microsoft Java-like

Perl (1987), *Python* (1991), scripting languages

… too many to cover here :)

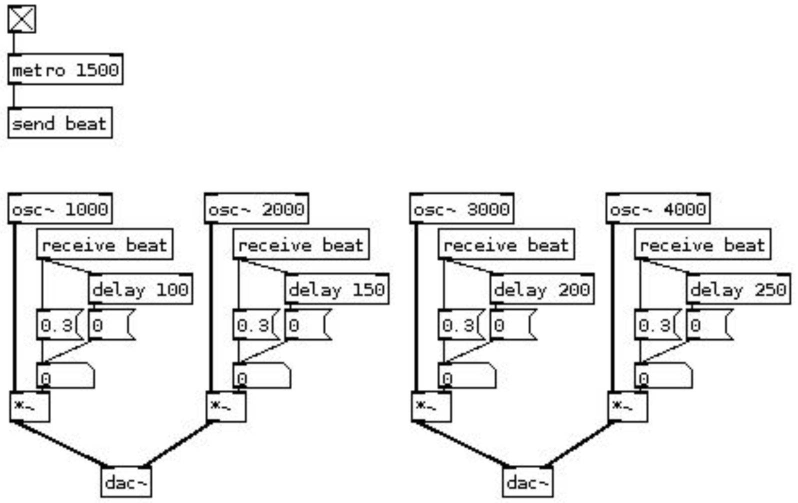
Forth example



```
include random.fs

create hidden 4 allot

: ok? ( str -- ? )
  dup 4 <> if 2drop false exit then
  1 9 lshift 1- -rot
  bounds do
    i c@ '1 -
    dup 0 9 within 0= if 2drop false leave then
    1 swap lshift over and
    dup 0= if nip leave then
    xor
  loop 0<> ;

: init
  begin
    hidden 4 bounds do 9 random '1 + i c! loop
    hidden 4 ok?
  until ;
```

Things happen on the *stack*

Visual programming



Not all programming is typing, here Pure Data

General trends

From low-level to high-level, less platform dependency

Object-oriented languages' popularity

Libraries, components, APIs

The Web as an application platform (Canvas, WebGL, WebAssembly, TypeScript …)

Free software, free tools – back to the roots

Detachment from everyday computer use