Programming and Data Structures - I

Lecture 2

Kripabandhu Ghosh

CDS, IISER Kolkata

PROGRAMMING LANGUAGE

Definition

Programming Language

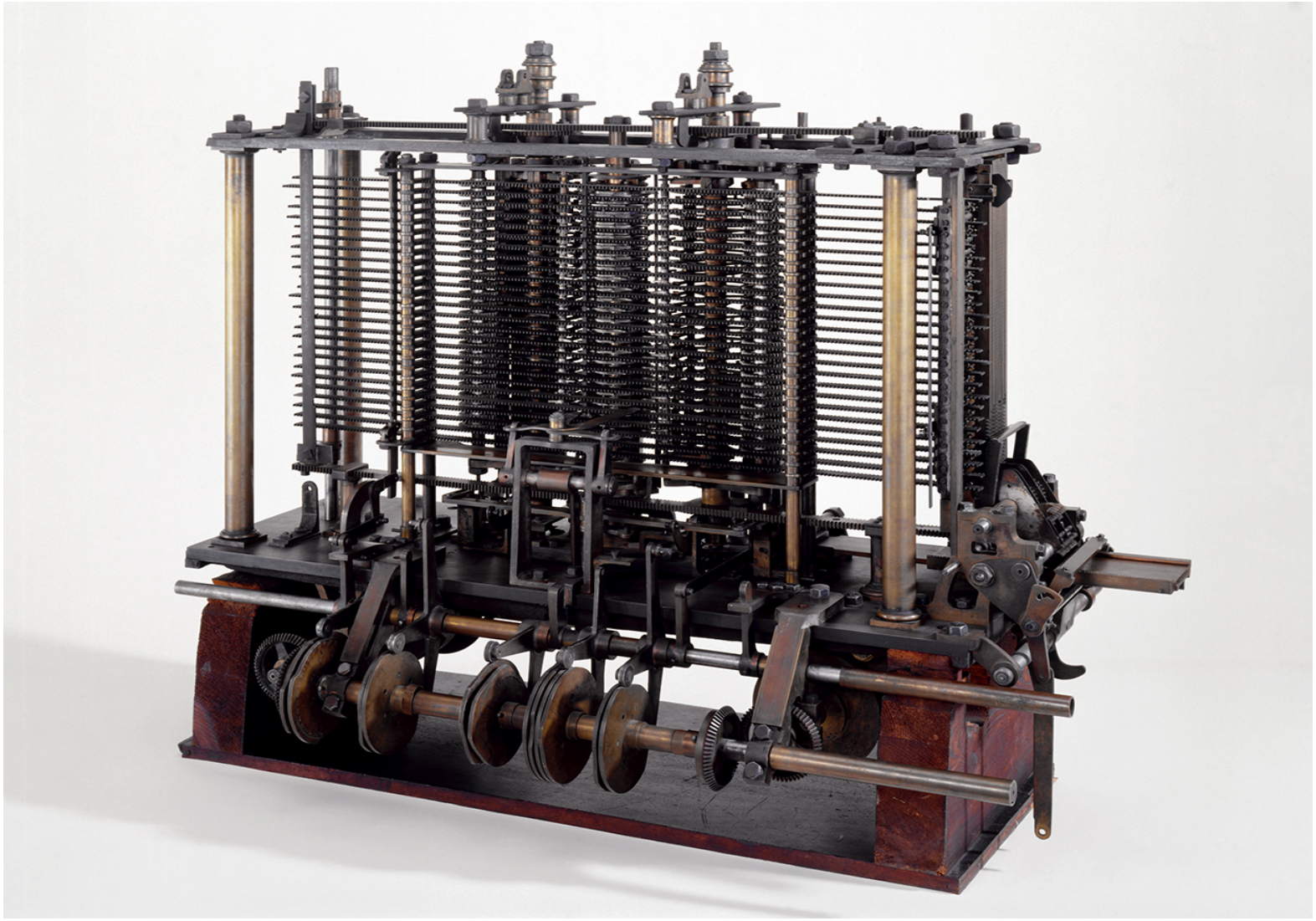
Computer-understandable language for the implementation of an algorithm (a program)

The First Program[1]

During 1842 – 1849, **Lady Ada Lovelace** is believed to have written the first program on Charles Babbage's Analytical Engine to compute Bernoulli numbers

[1] Image source: `https://en.wikipedia.org/wiki/Ada_Lovelace`

Charles Babbage's Analytical Engine[2]



[2]Image source: https://en.wikipedia.org/wiki/Analytical_engine

The first program[3]

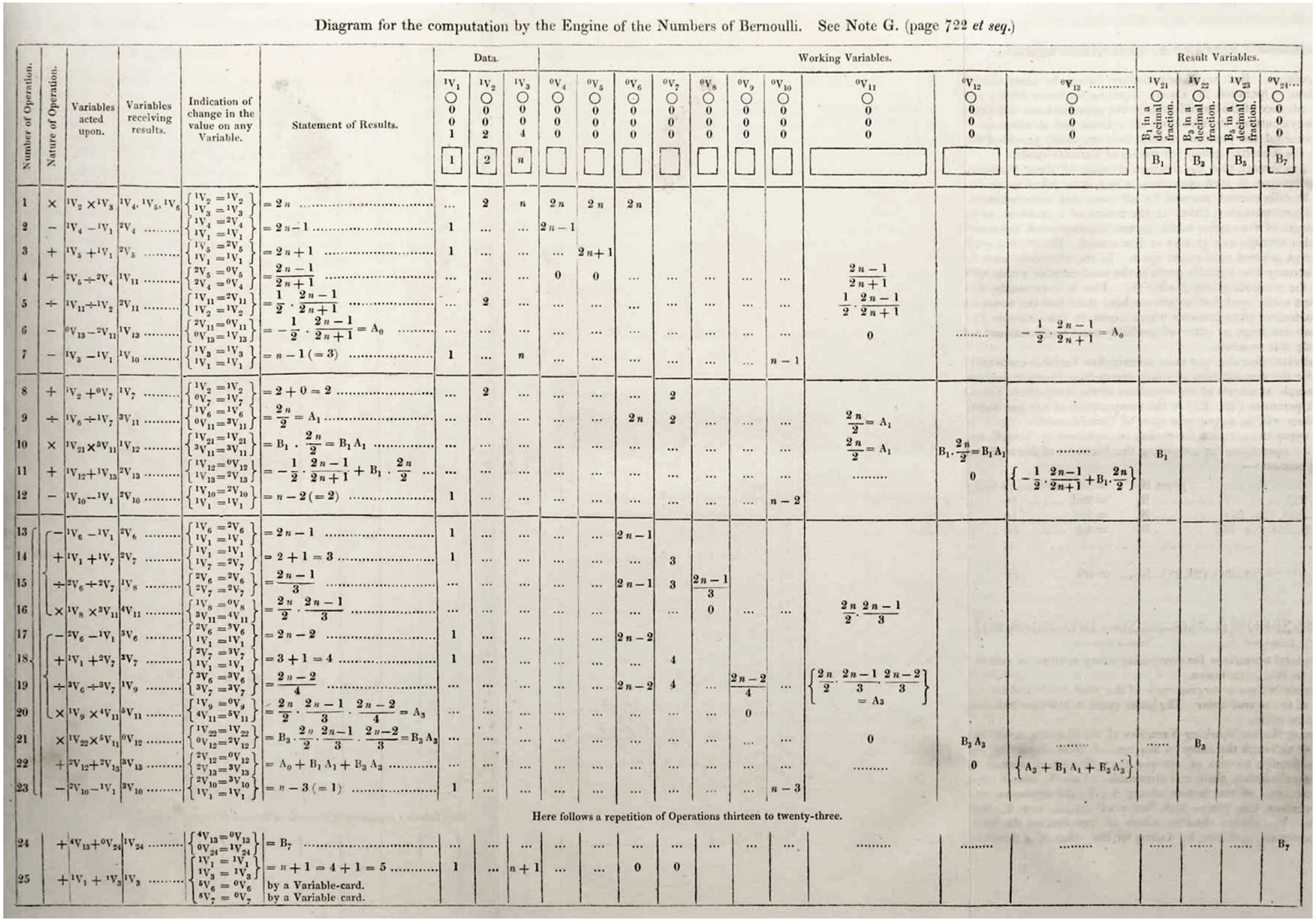


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data. $^1V_1$ 1	Data. $^1V_2$ 2	Data. $^1V_3$ n	Working Variables. $^0V_4$	$^0V_5$	$^0V_6$	$^0V_7$	$^0V_8$	$^0V_9$	$^0V_{10}$	$^0V_{11}$	$^0V_{12}$	$^0V_{13}$	Result Variables. $^1V_{21}$ B_1	$^1V_{22}$ B_3	$^1V_{23}$ B_5	$^0V_{24}$... B_7
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\{{}^1V_2 = {}^1V_2;\ {}^1V_3 = {}^1V_3\}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$\{{}^1V_4 = {}^2V_4;\ {}^1V_1 = {}^1V_1\}$	$= 2n-1$	1	...	...	$2n-1$													
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$\{{}^1V_5 = {}^2V_5;\ {}^1V_1 = {}^1V_1\}$	$= 2n+1$	1	...	...	...	$2n+1$												
4	÷	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$\{{}^2V_5 = {}^0V_6;\ {}^2V_4 = {}^0V_4\}$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$						
5	÷	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$\{{}^1V_{11} = {}^2V_{11};\ {}^1V_2 = {}^1V_2\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	−	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$\{{}^2V_{11} = {}^0V_{11};\ {}^0V_{13} = {}^1V_{13}\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	.....	$-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - {}^1V_1$	$^1V_{10}$	$\{{}^1V_3 = {}^1V_3;\ {}^1V_1 = {}^1V_1\}$	$= n-1\ (=3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^1V_2 + {}^0V_7$	$^1V_7$	$\{{}^1V_2 = {}^1V_2;\ {}^0V_7 = {}^1V_7\}$	$= 2+0=2$	...	2	...	...	...	...	2										
9	÷	$^1V_6 \div {}^1V_7$	$^3V_{11}$	$\{{}^1V_6 = {}^1V_6;\ {}^0V_{11} = {}^3V_{11}\}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
10	×	$^1V_{21} \times {}^3V_{11}$	$^1V_{12}$	$\{{}^1V_{21} = {}^1V_{21};\ {}^3V_{11} = {}^3V_{11}\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	.....	B_1			
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$\{{}^1V_{12} = {}^0V_{12};\ {}^1V_{13} = {}^2V_{13}\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$ ...	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{-\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}\right\}$				
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\{{}^1V_{10} = {}^2V_{10};\ {}^1V_1 = {}^1V_1\}$	$= n-2\ (=2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	−	$^1V_6 - {}^1V_1$	$^2V_6$	$\{{}^1V_6 = {}^2V_6;\ {}^1V_1 = {}^1V_1\}$	$= 2n-1$	1	...	...	...	...	$2n-1$											
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$\{{}^1V_1 = {}^1V_1;\ {}^1V_7 = {}^2V_7\}$	$= 2+1=3$	1	...	...	...	...	...	3										
15	÷	$^2V_6 \div {}^2V_7$	$^1V_8$	$\{{}^2V_6 = {}^2V_6;\ {}^2V_7 = {}^2V_7\}$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$									
16	×	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$\{{}^1V_8 = {}^0V_8;\ {}^3V_{11} = {}^4V_{11}\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
17	−	$^2V_6 - {}^1V_1$	$^3V_6$	$\{{}^2V_6 = {}^3V_6;\ {}^1V_1 = {}^1V_1\}$	$= 2n-2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$\{{}^2V_7 = {}^3V_7;\ {}^1V_1 = {}^1V_1\}$	$= 3+1=4$	1	...	...	...	...	...	4										
19	÷	$^3V_6 \div {}^3V_7$	$^1V_9$	$\{{}^3V_6 = {}^3V_6;\ {}^3V_7 = {}^3V_7\}$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{\frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = A_3\right\}$							
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$\{{}^1V_9 = {}^0V_9;\ {}^4V_{11} = {}^5V_{11}\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0								
21	×	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$\{{}^1V_{22} = {}^1V_{22};\ {}^0V_{12} = {}^2V_{12}\}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	.....	.....	B_3		
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\{{}^2V_{12} = {}^0V_{12};\ {}^2V_{13} = {}^3V_{13}\}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...	.....	0	$\{A_3 + B_1 A_1 + B_3 A_3\}$				
23	−	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$\{{}^2V_{10} = {}^3V_{10};\ {}^1V_1 = {}^1V_1\}$	$= n-3\ (=1)$	1	...	...	...	...	...	...	...	...	$n-3$							

Here follows a repetition of Operations thirteen to twenty-three.

24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$\{{}^4V_{13} = {}^0V_{13};\ {}^0V_{24} = {}^1V_{24}\}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	.....	.....	.....	.....	.....	.....	B_7
25	+	$^1V_1 + {}^1V_3$	$^1V_3$	$\{{}^1V_1 = {}^1V_1;\ {}^1V_3 = {}^1V_3;\ {}^5V_6 = {}^0V_6;\ {}^5V_7 = {}^0V_7\}$	$= n+1 = 4+1=5$ ..... by a Variable-card. by a Variable card.	1	...	$n+1$	...	...	0	0										

[3] Image source: https://en.wikipedia.org/wiki/Algorithm#/media/File:Diagram_for_the_computation_of_Bernoulli_numbers.jpg

The First High Level Programming Language[4]

The first high-level programming language *Plankalkül* written by **Konrad Zuse** between 1942 and 1945

[4] Image source: `https://en.wikipedia.org/wiki/Konrad_Zuse`

Plankalkül code to find maximum of three variables

```
P1 max3 (V0[:8.0],V1[:8.0],V2[:8.0]) → R0[:8.0]
max(V0[:8.0],V1[:8.0]) → Z1[:8.0]
max(Z1[:8.0],V2[:8.0]) → R0[:8.0]
END
P2 max (V0[:8.0],V1[:8.0]) → R0[:8.0]
V0[:8.0] → Z1[:8.0]
(Z1[:8.0] < V1[:8.0]) → V1[:8.0] → Z1[:8.0]
Z1[:8.0] → R0[:8.0]
END
```

The First General Purpose High Level Programming Language[5]

FORTRAN was invented in 1954 at IBM by a team led by **John Backus**; FORTRAN was the first widely used high-level general purpose programming language

[5] Image source: `https://en.wikipedia.org/wiki/John_Backus`

FORTRAN code[6]

```
C AREA OF A TRIANGLE - HERON'S FORMULA
C INPUT - CARD READER UNIT 5, INTEGER INPUT, NO BLANK CARD FOR END OF DATA
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPAYS ERROR MESSAGE ON OUTPUT
  501 FORMAT(3I5)
  601 FORMAT(" A= ",I5,"  B= ",I5,"  C= ",I5,"  AREA= ",F10.2,
     $"SQUARE UNITS")
  602 FORMAT("NORMAL END")
  603 FORMAT("INPUT ERROR OR ZERO VALUE ERROR")
      INTEGER A,B,C
   10 READ(5,501,END=50,ERR=90) A,B,C
      IF(A=0 .OR. B=0 .OR. C=0) GO TO 90
      S = (A + B + C) / 2.0
      AREA = SQRT( S * (S - A) * (S - B) * (S - C) )
      WRITE(6,601) A,B,C,AREA
      GO TO 10
   50 WRITE(6,602)
      STOP
   90 WRITE(6,603)
      STOP
      END
```

[6] Image source: https://en.wikibooks.org/wiki/Fortran/Fortran_examples

Other Programming languages

- LISP released in 1958
- COBOL (COmmon Business Oriented Language), business purpose language, released in 1959
- BASIC (Beginners' All-purpose Symbolic Instruction Code), a general-purpose, high-level programming languages designed for ease of use, released in 1964

COBOL code[7]

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

PROCEDURE DIVISION.
    A-PARA.
    PERFORM DISPLAY 'IN A-PARA'
    END-PERFORM.
    PERFORM C-PARA THRU E-PARA.

    B-PARA.
    DISPLAY 'IN B-PARA'.
    STOP RUN.

    C-PARA.
    DISPLAY 'IN C-PARA'.

    D-PARA.
    DISPLAY 'IN D-PARA'.

    E-PARA.
    DISPLAY 'IN E-PARA'.
```

[7] Image source: https://www.tutorialspoint.com/cobol/cobol_loop_statements.htm

BASIC code[8]

```
10 INPUT "What is your name: "; U$
20 PRINT "Hello "; U$
30 INPUT "How many stars do you want: "; N
40 S$ = ""
50 FOR I = 1 TO N
60 S$ = S$ + "*"
70 NEXT I
80 PRINT S$
90 INPUT "Do you want more stars? "; A$
100 IF LEN(A$) = 0 THEN GOTO 90
110 A$ = LEFT$(A$, 1)
120 IF A$ = "Y" OR A$ = "y" THEN GOTO 30
130 PRINT "Goodbye "; U$
140 END
```

[8] Image source: https://en.wikipedia.org/wiki/BASIC

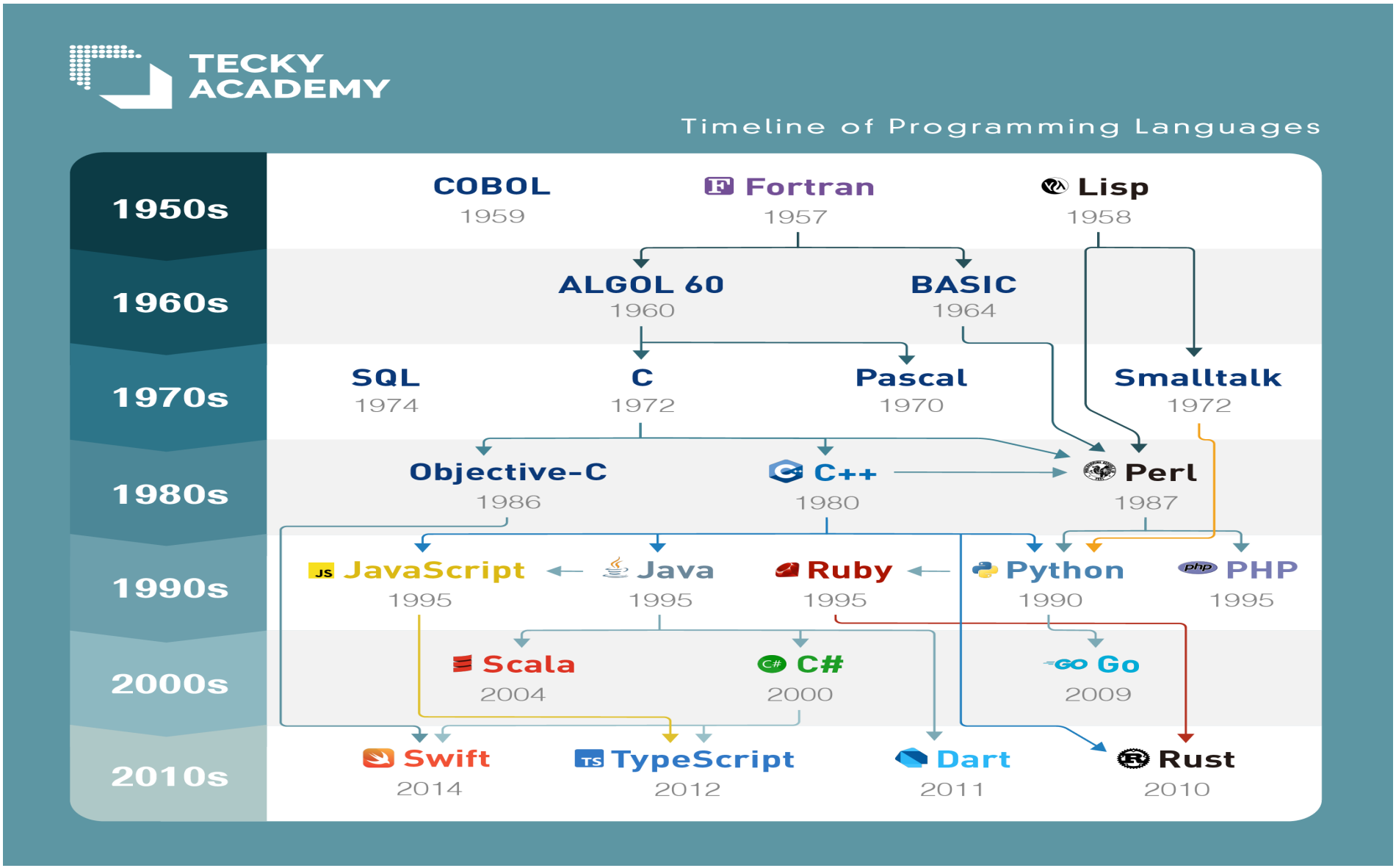
C Programming Language

History[9]

Developed at Bell Labs by **Dennis Ritchie** between 1972 and 1973

[9] Image source: `https://en.wikipedia.org/wiki/Dennis_Ritchie`

History[10]



[10]Image source: https://tecky.io/en/blog/evolution-of-programming-languages/

About C

Background

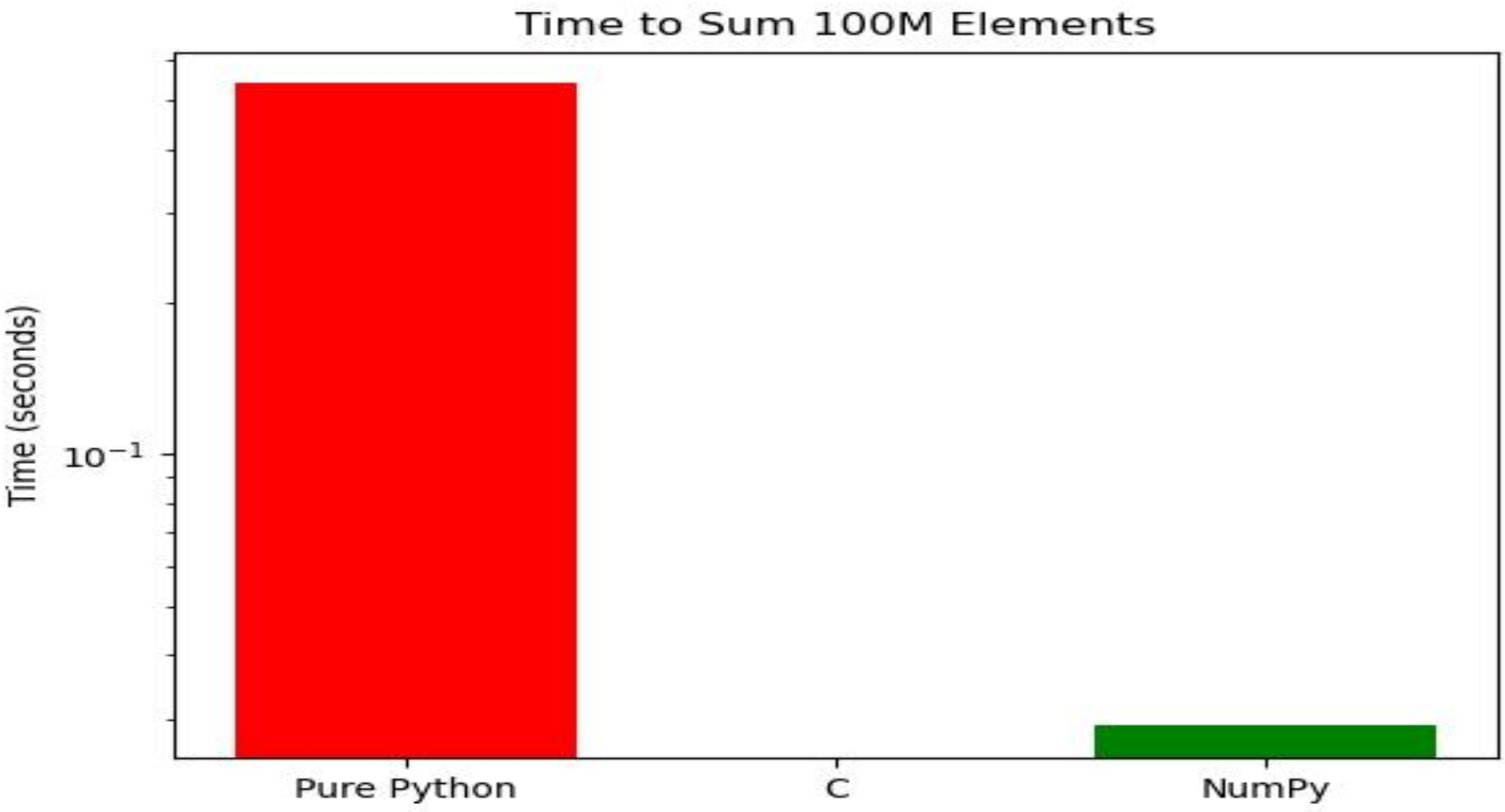
- C was the successor of the 'Basic Combined Programming Language' (BCPL) called B developed at Cambridge University in 1960's
- C was developed along with the UNIX operating system

Features

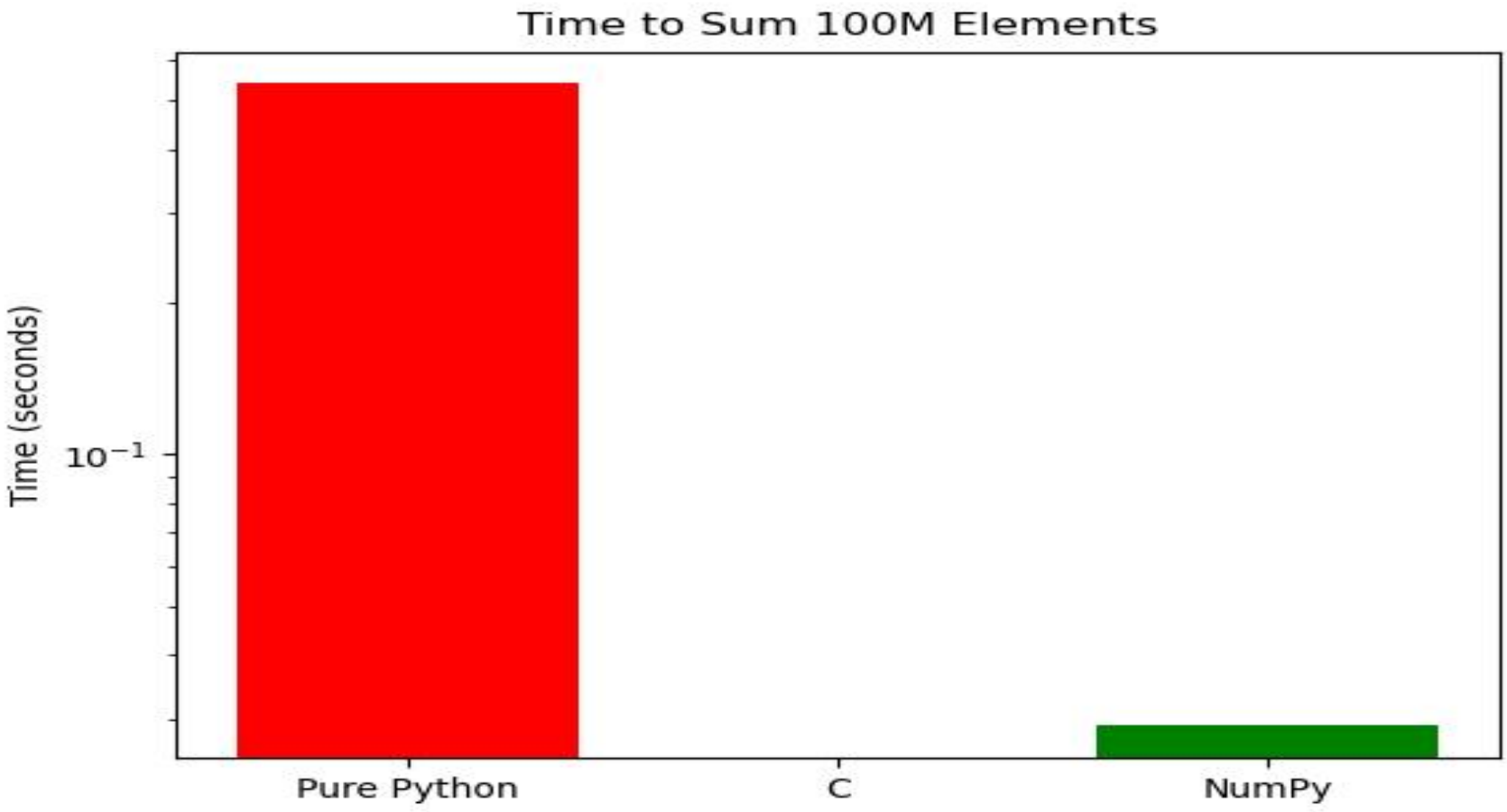
- C compiler integrates capabilities of an assembly-level language with those of a high-level language – suitable for both system software and business purposes
- High speed (faster than BASIC, much faster than Python[a])
- Portable
- Suitable for structured programming (provision for modules, functions, and blocks)
- Free-form (no requirements of starting from specific line numbers, indentation, etc.,) unlike COBOL, FORTRAN, Python
- Good precursor of OOP (e.g., C++, Java, Python)

[a] `https://medium.com/codex/how-slow-is-python-compared-to-c-3795071ce82a`

C, Python and NumPy



C, Python and NumPy

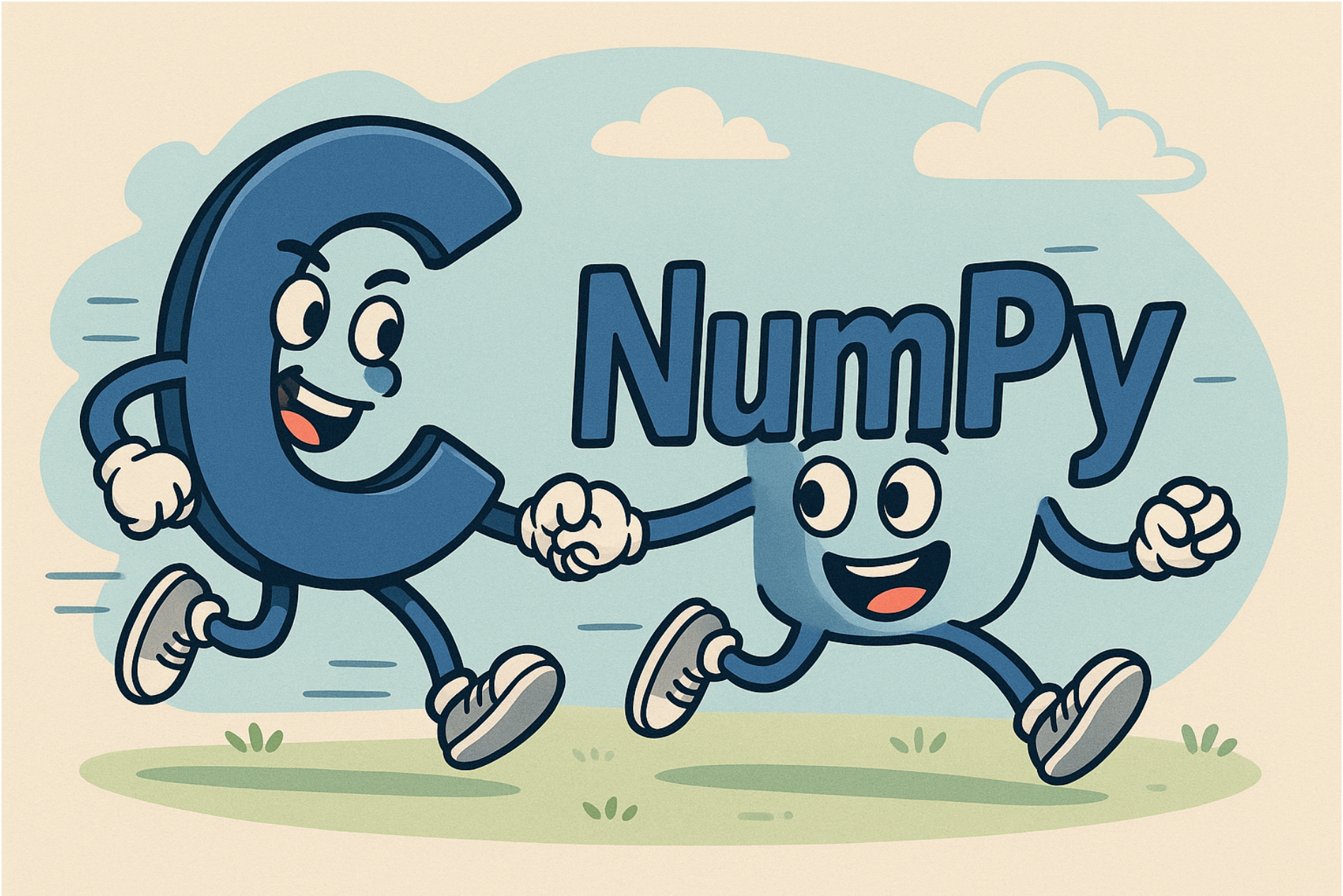


NumPy largely (e.g. **sum()** function) is implemented in C[a]

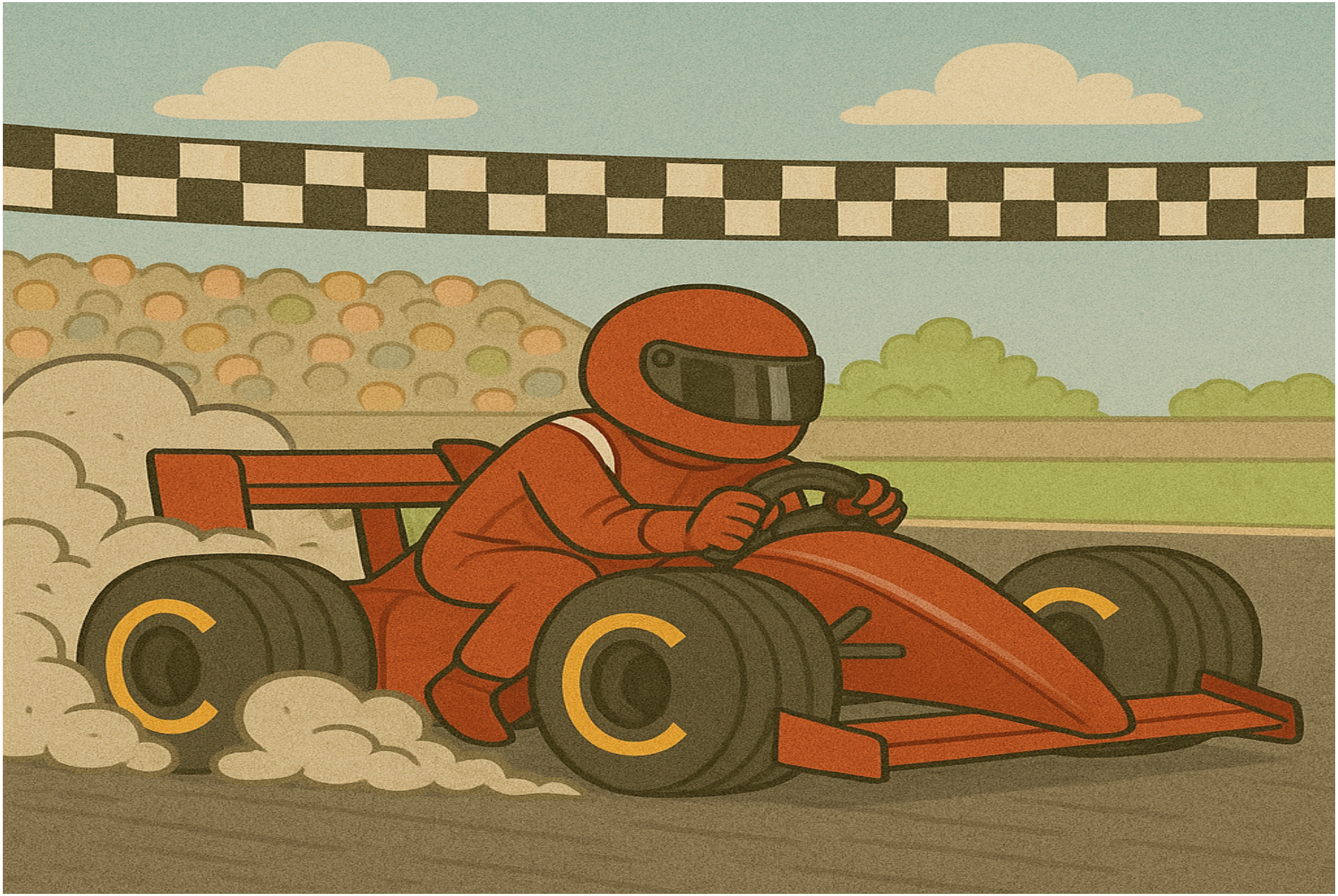
[a] Harris, C.R. et al. Array programming with NumPy. Nature 585, 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2

NumPy and C

NumPy and C



Getting started with C



Getting started with C

```c
#include<stdio.h> //Standard library


int main()
  {
      printf("Got started!!!");

  }
```

Getting started with C

```c
#include<stdio.h> //Standard library


int main()
  {
      printf("Got started!!!");

  }
```

Output

Got started!!!

Python version

```
print("Got started!!!")
```

Python version

```
print("Got started!!!")
```

Output

Got started!!!

C vs Python

C

```c
#include<stdio.h>

int main()

{

printf("Got started!!!");

}
```

Python

```python
print("Got started!!!")
```

C vs Python

C

```c
#include<stdio.h>

int main()

{

printf("Got started!!!");

}
```

Python

```python
print("Got started!!!")
```

- Semicolon in C
- No indentation in C
- Declaration of variables in C (to be discussed shortly)

The First Program (contd.)

- **main** is a special and mandatory function in C that indicates the start and end of the program
- There can be exactly one **main** function in a C program
- **main** may or may not take *arguments*
- Braces ({}) is the scope of a function
- Each statement is terminated by a semicolon (;)
- Usually lowercase is followed; uppercase is reserved for symbolic constants

INTERNATIONAL STANDARD ISO/IEC ISO/IEC 9899:2024

The function called at program startup is named **main**, with return type `int`.

- If the return type of the main function is a type compatible with `int`, a return from the initial call to the main function is equivalent to calling the `exit` function with the value returned by the main function as its argument reaching the } that terminates the main function returns a value of 0 (success; non-zero for error).
- If the return type is not compatible with `int`, the termination status returned to the host environment is unspecified.

An Input and Output Program

```c
#include<stdio.h> //Standard library

int main()
{
      int num1, num2, sum = 0;  //Variable declaration

      printf("\n Give two integers below:\n");
      scanf("%d %d", &num1, &num2); //Reading from the terminal

      sum = num1 + num2;

      printf("The sum = %d \n", sum); //Printing on the terminal

}
```

An Input and Output Program

```c
#include<stdio.h> //Standard library

int main()
{
      int num1, num2, sum = 0;  //Variable declaration

      printf("\n Give two integers below:\n");
      scanf("%d %d", &num1, &num2); //Reading from the terminal

      sum = num1 + num2;

      printf("The sum = %d \n", sum); //Printing on the terminal

}
```

Output

Give two integers below:
1
2
The sum = 3

Structure of a C Program

Documentation
Linking
Definition
Global Declaration
main()
User-defined Functions

Structure of a C Program

Documentation	✗
Linking	✓
Definition	✗
Global Declaration	✗
main()	✓
User-defined Functions	✗

Structure of a C Program: Example

Section	Program code
Documentation	/* Program to calculate the volume of a sphere Code written by Kripa */
Linking	#include<stdio.h> #include<math.h>
Definition	#define PI 3.14159 float calculate_volume();//Forward declaration
Global Declaration	float r = 5.0;

main()	int main()
	{
	float volume; //Variable declaration
	volume = calculate_volume(); //Function call
	printf("\nVolume = %f\n", volume);
	}

Structure of a C Program: Example (contd.)

Section	Program code

User-defined Functions	float calculate_volume()
	{
	float vol; //Variable declaration
	vol = (4.0/3.0)*PI*pow(r,3.0);
	return vol; //Returning the value to main()
	}

```c
 /* Program to calculate the volume of a sphere
 Code written by Kripa */
#include<stdio.h>
#include<math.h>

#define PI 3.14159

float calculate_volume(); //Forward declaration

float r = 5.0;

int main()
{
   float volume; //Variable declaration

   volume = calculate_volume(); //Function call

   printf("\nVolume = %f\n", volume);

}
```

Running the code (contd.)

```
float calculate_volume()
{
 float vol; //Variable declaration

 vol = (4.0/3.0)*PI*pow(r,3.0);

 return vol; //Returning the value to main()

}
```

Output

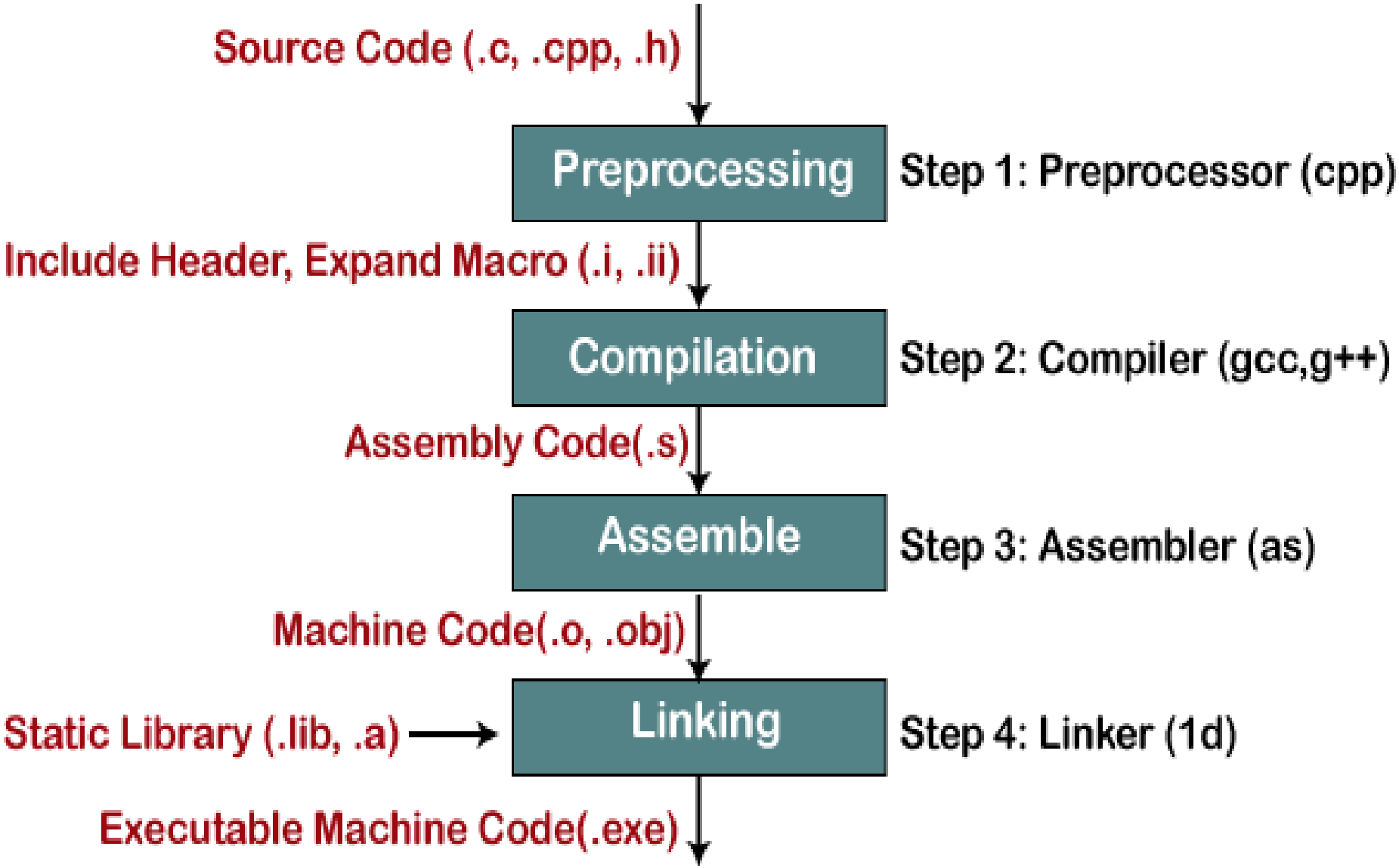
Volume = 523.598328

Execution of a C program

Steps

- Writing the program in C (volume.c)
- Compilation (Conversion from C to object code, i.e. machine readable code)
- Linking (Linking the program to the functions from C library)
- Execution

Compilation etc.[11]



[11] https://www.javatpoint.com/gcc-linux

Compilation and Linking: LINUX

`> gcc volume.c -lm` produces the object code `a.out`.

`> gcc -o volume volume.c -lm` produces the object code `volume`.

Execution: LINUX

```
> ./a.out
```
.

```
> ./volume
```
.

Execution: LINUX

```
> ./a.out > out
```
.

```
> ./volume > out
```
.

Execution: Editors and Integrated Development Environment (IDE)s (opensource)

- Vi/Vim (Editor) + GCC (Compiler) : Linux
- Dev C++ (Windows only)
- Cygwin (Linux-like feeling in Windows)
- Visual Studio Code (Windows/Linux)
- Eclipse (Windows/Linux)
- CodeLite (Windows/Linux)

```c
#include<stdio.h> //Standard library


int main()
  {
      printf("THANK YOU!!!");

  }
```

```c
#include<stdio.h> //Standard library


int main()
  {
      printf("THANK YOU!!!");

  }
```

Output

THANK YOU!!!


THANK
YOU

C you again...