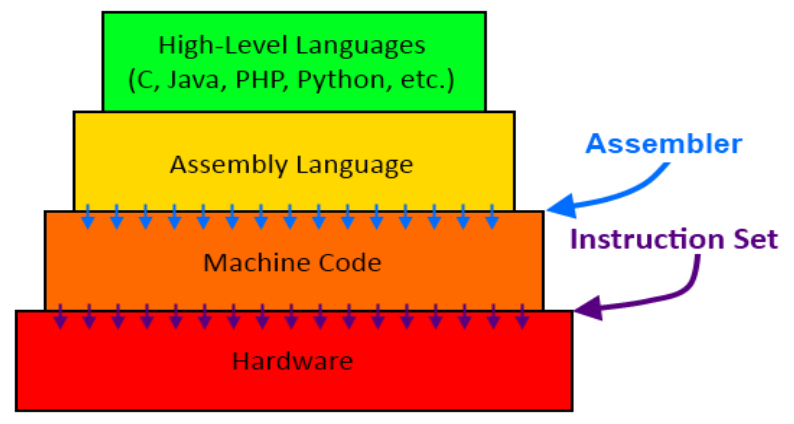
МОВИ ПРОГРАМУВАННЯ. ТРАНСЛЯЦІЯ ПРОГРАМНОГО КОДУ

Питання 1.3

Види мов програмування



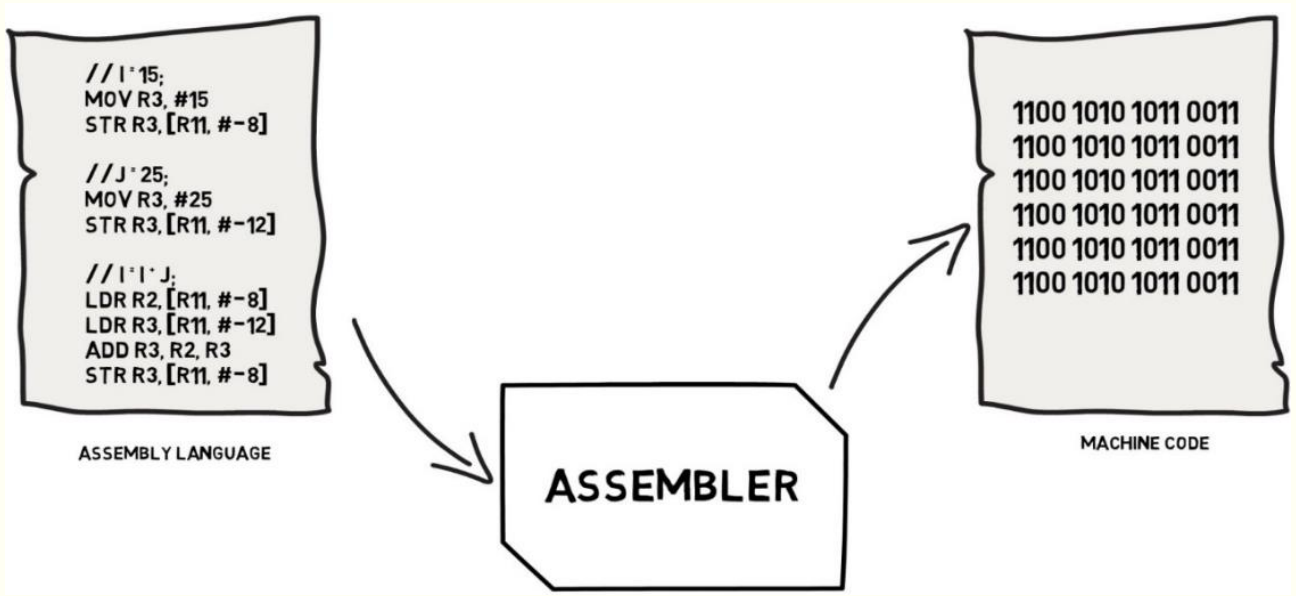
- ***Машинний код*** – набір команд (інструкцій), які виконуються безпосередньо центральним процесором без транслятора.
- ***Машинно-орієнтовані (низькорівневі) мови програмування*** – кожна команда відповідає одній команді процесора

Machine code	Instruction	Addressing mode	Example
0000	STORE	Address	STO 12
0001	LOAD	Number	LDA #12
0010	LOAD	Address	LDA 12
0100	ADD	Number	ADD #12
1000	ADD	Address	ADD 12
1111	HALT	None	HALT

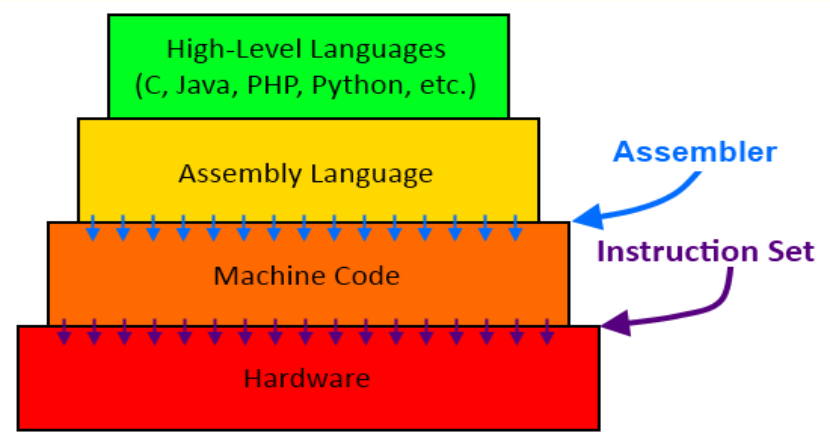
C code: d[3] = d[2] + a;

Assembly: # addi instructions as before

```
lw    $t0, 8($s4)   #  d[2] is brought into $t0
lw    $t1, 0($s1)   #   a  is brought into $t1
add   $t0, $t0, $t1   #  the sum is in $t0
sw    $t0, 12($s4)  #  $t0 is stored into d[3]
```



Мови високого рівня



- Наближені до англійської мови, легше сприймаються людиною
- Не залежать від комп'ютера

C

```c
#include <stdio.h>
int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```

D

```d
module helloworld;
import std.stdio;
void main() {
    writeln("Hello, world!");
}
```

Delphi

```
Program Hello_World;
{$APPTYPE CONSOLE}
Begin
 WriteLn('Hello, world!');
End.
```

Hello

```
h
```

Lisp

```
"Hello, world!"
```

Boo

```
import System.Drawing
import System.Windows.Forms
f = Form()
f.Controls.Add(Label(Text: "Hello, world!", Location: Point(40,30)))
f.Controls.Add(Button(Text: "Ok", Location: Point(50, 55), Click: {Application.Exit()}))
Application.Run(f)
```

Groovy

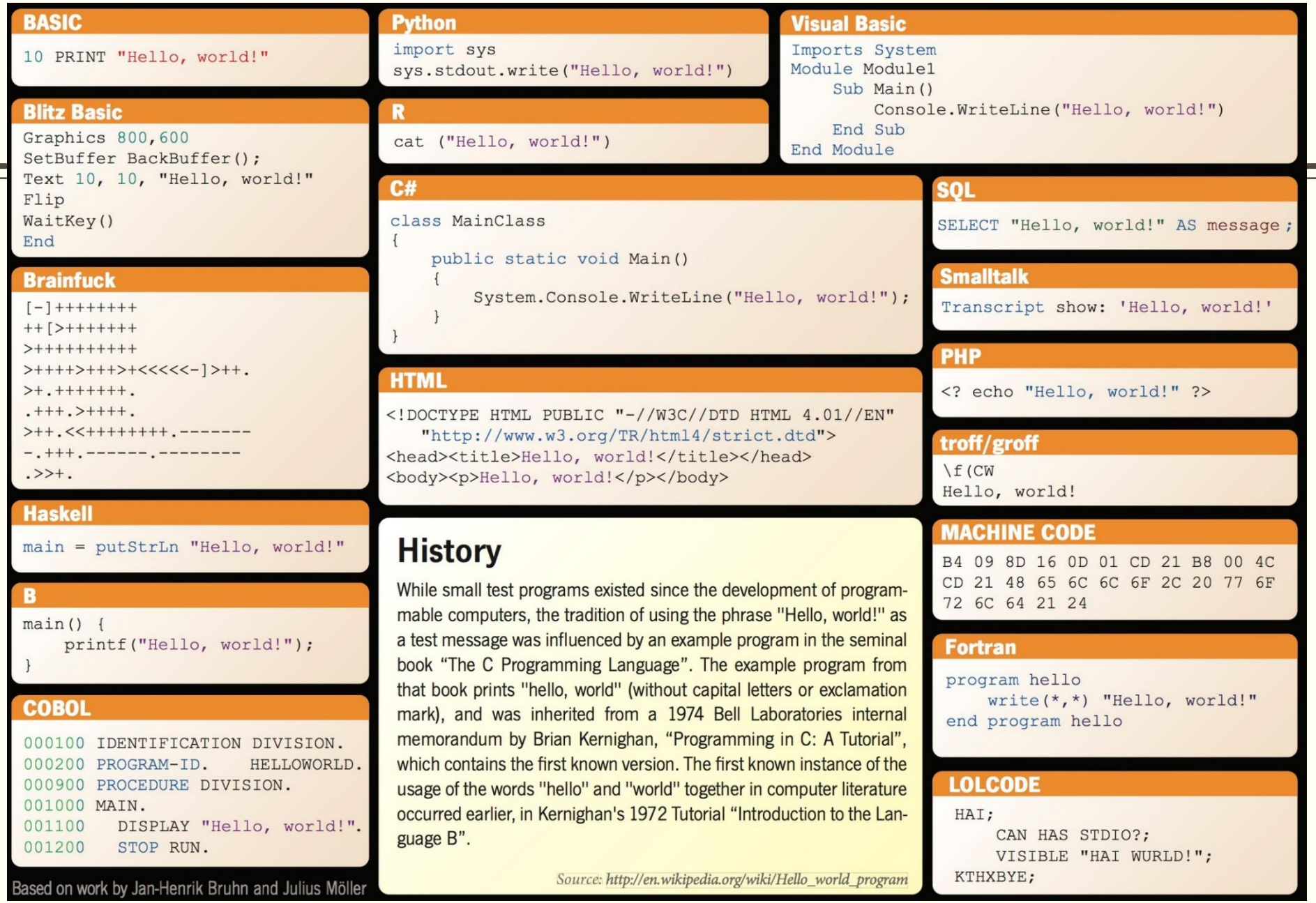
```
println "Hello, world!"
```

Perl

```
print "Hello, world!\n";
```

Мови високого рівня

- Дуже часто для виводу використовуються ***вбудовані функції*** – іменовані блоки коду, які дозволяють вивести на екран текст, що передається (тут – Hello, world!)



BASIC

```
10 PRINT "Hello, world!"
```

Blitz Basic

```
Graphics 800,600
SetBuffer BackBuffer();
Text 10, 10, "Hello, world!"
Flip
WaitKey()
End
```

Brainfuck

```
[-]++++++++
++[>+++++++
>++++++++++
>++++>+++>+<<<<<-]>++.
>+.+++++++.
.+++.>++++.
>++.<<++++++++.-------
-.+++.------.--------
.>>+.
```

Haskell

```
main = putStrLn "Hello, world!"
```

B

```
main() {
    printf("Hello, world!");
}
```

COBOL

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    HELLOWORLD.
000900 PROCEDURE DIVISION.
001000 MAIN.
001100   DISPLAY "Hello, world!".
001200   STOP RUN.
```

Based on work by Jan-Henrik Bruhn and Julius Möller

Python

```
import sys
sys.stdout.write("Hello, world!")
```

R

```
cat ("Hello, world!")
```

Visual Basic

```
Imports System
Module Module1
    Sub Main()
        Console.WriteLine("Hello, world!")
    End Sub
End Module
```

C#

```
class MainClass
{
    public static void Main()
    {
        System.Console.WriteLine("Hello, world!");
    }
}
```

HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<head><title>Hello, world!</title></head>
<body><p>Hello, world!</p></body>
```

History

While small test programs existed since the development of programmable computers, the tradition of using the phrase "Hello, world!" as a test message was influenced by an example program in the seminal book "The C Programming Language". The example program from that book prints "hello, world" (without capital letters or exclamation mark), and was inherited from a 1974 Bell Laboratories internal memorandum by Brian Kernighan, "Programming in C: A Tutorial", which contains the first known version. The first known instance of the usage of the words "hello" and "world" together in computer literature occurred earlier, in Kernighan's 1972 Tutorial "Introduction to the Language B".

Source: http://en.wikipedia.org/wiki/Hello_world_program

SQL

```
SELECT "Hello, world!" AS message;
```

Smalltalk

```
Transcript show: 'Hello, world!'
```

PHP

```
<? echo "Hello, world!" ?>
```

troff/groff

```
\f(CW
Hello, world!
```

MACHINE CODE

```
B4 09 8D 16 0D 01 CD 21 B8 00 4C
CD 21 48 65 6C 6C 6F 2C 20 77 6F
72 6C 64 21 24
```

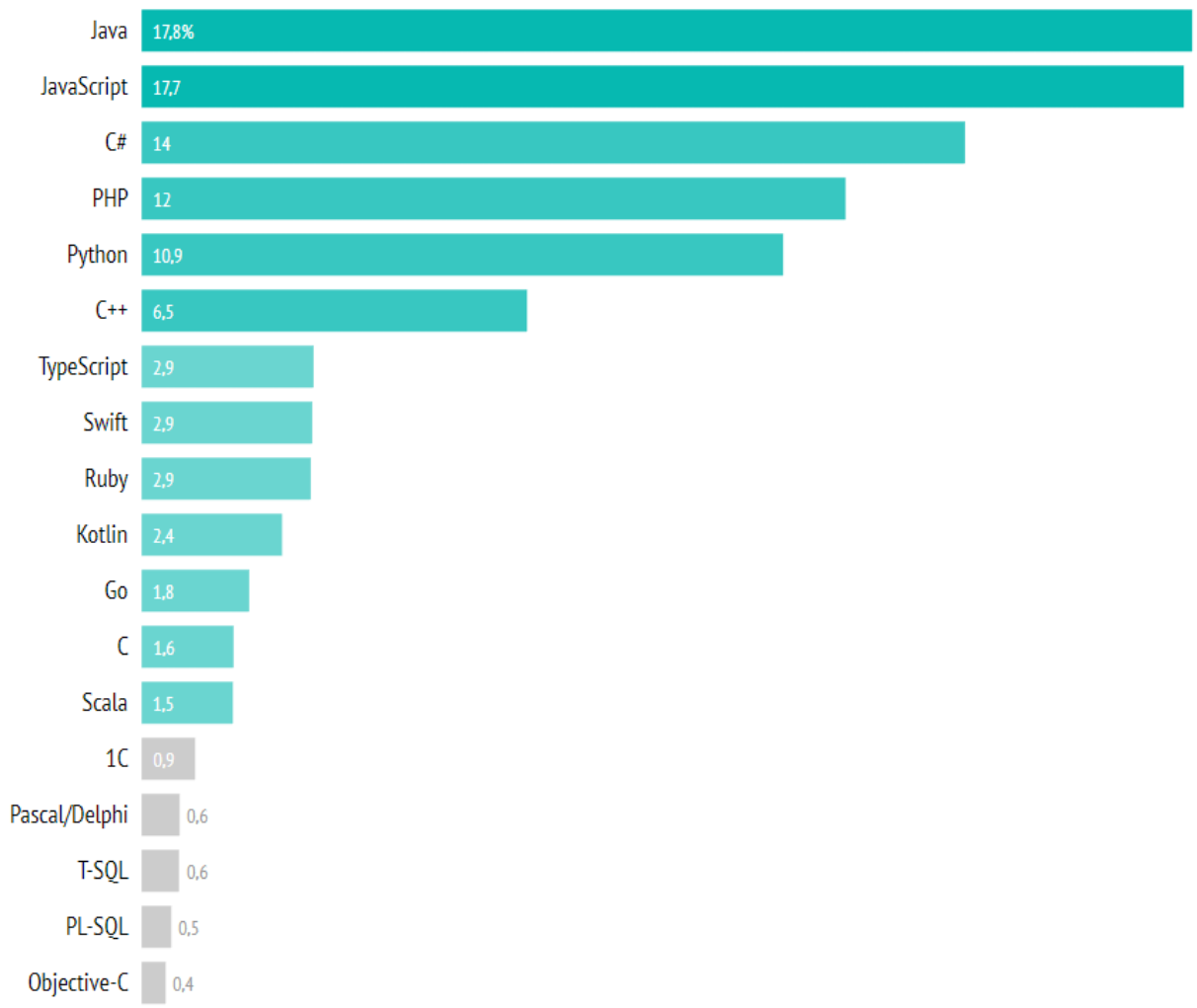
Fortran

```
program hello
    write(*,*) "Hello, world!"
end program hello
```

LOLCODE

```
HAI;
    CAN HAS STDIO?;
    VISIBLE "HAI WURLD!";
KTHXBYE;
```

Популярні мови програмування (Україна, січень 2019)



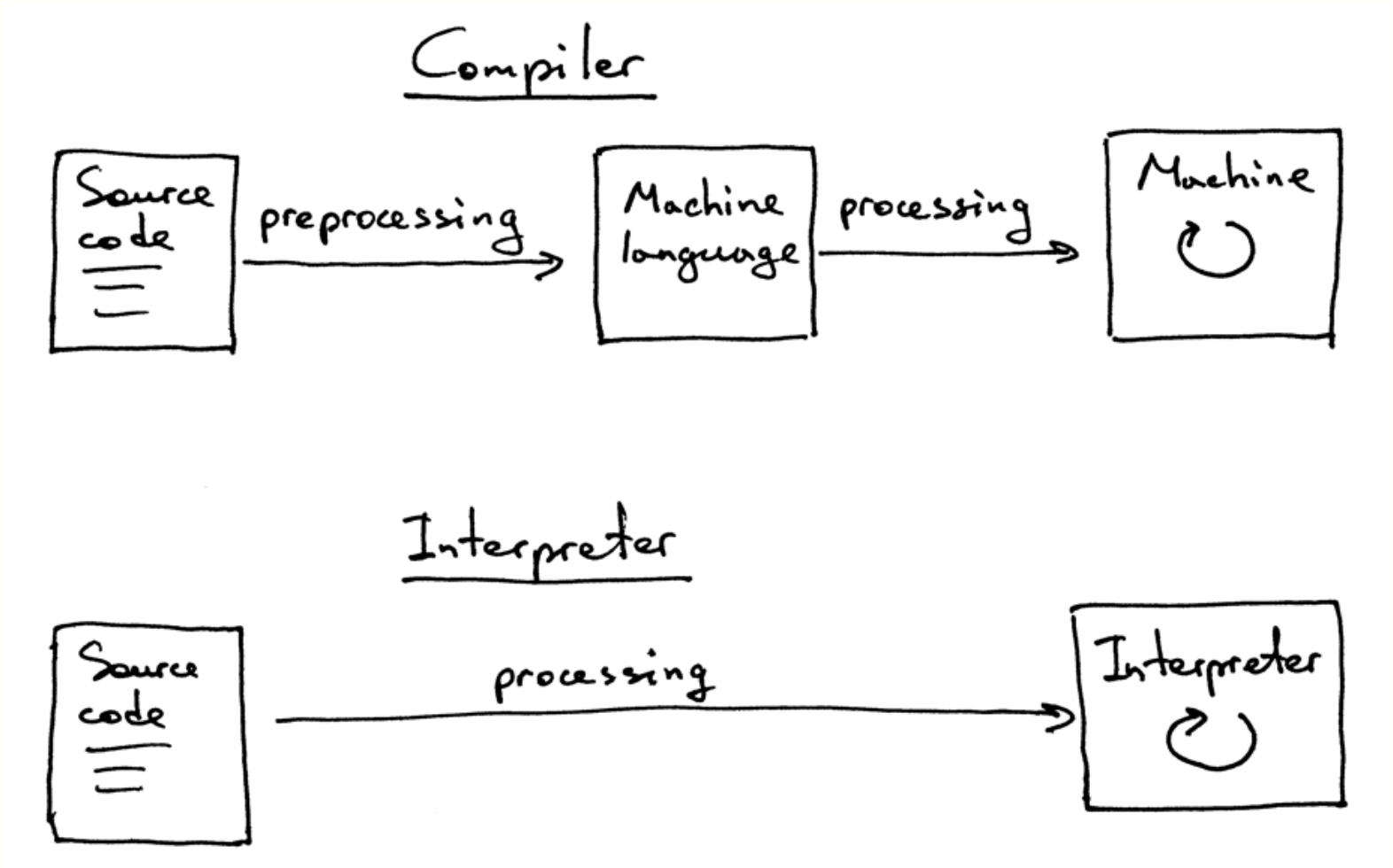
- Kotlin витисне Java у розробці під Android.
- Продовжує зростати TypeScript.
- JavaScript-екосистема поступово переймає першість у JVM.
- Частка C# лишається майже незмінною.
- Прогнозовано росте використання Python.

Програмування мовою високого рівня

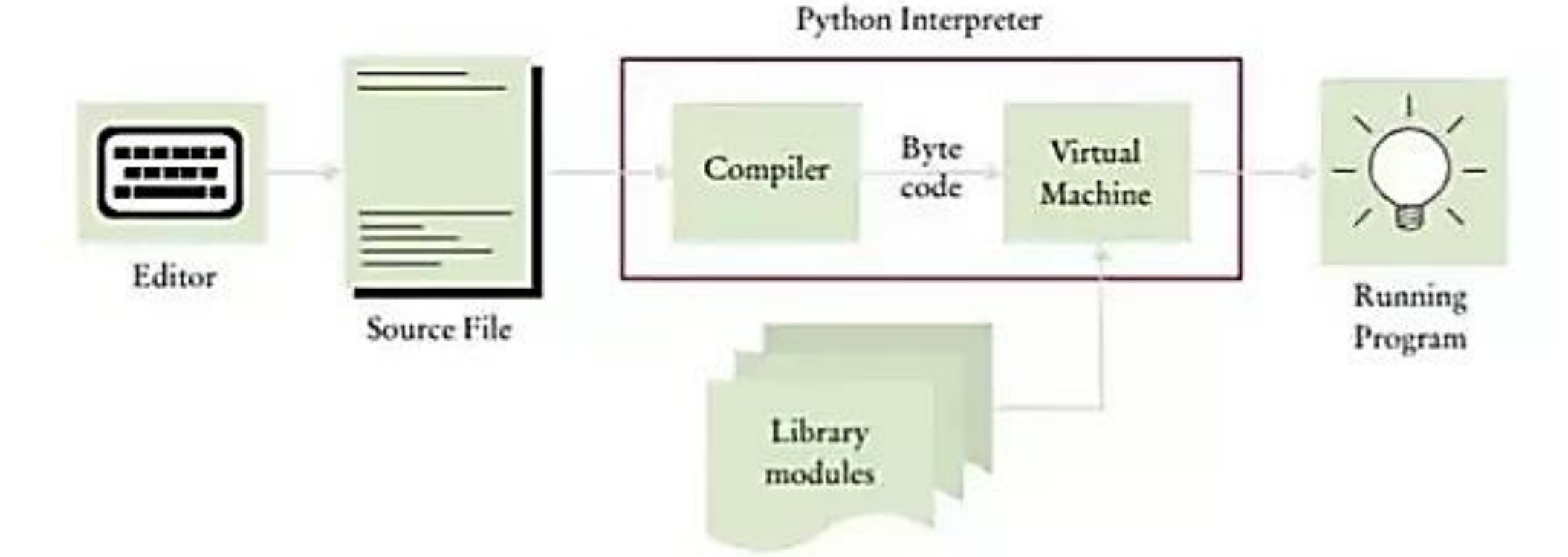
- ***Мова програмування*** визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми і дії, які виконає комп'ютер під її управлінням.
  - ***Лексика*** – сукупність слів у мові.
  - ***Синтаксис*** вивчає граматичну будову словосполучень та речень у мові.
  - ***Семантика*** – смислове значення слів.

- Мова програмування призначена для передачі команд і даних від людини до комп'ютера.
  - Природні мови використовуються для спілкування людей між собою.

Компілятор vs інтерпретатор



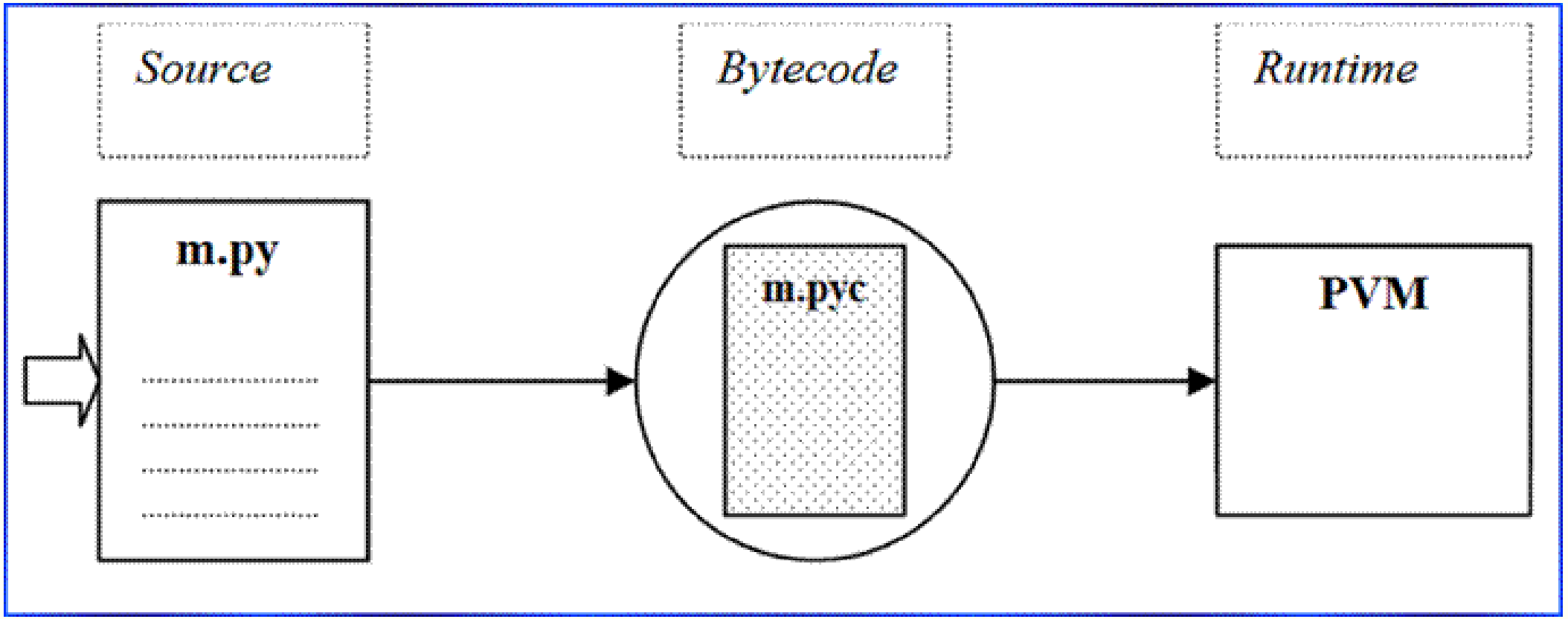
Як первинний код Python запускається на виконання?



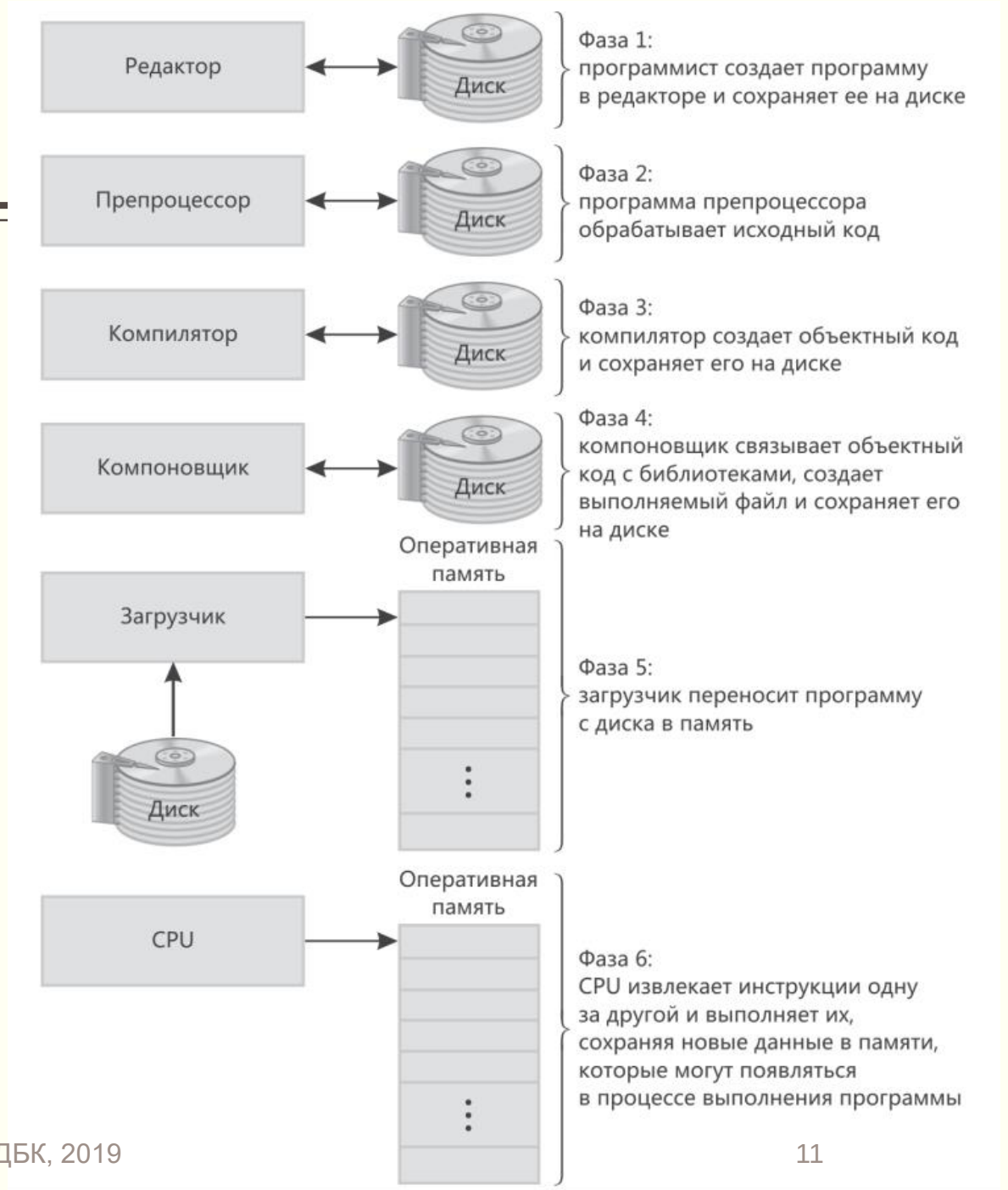
Програма мовою Python працює напряму з первинного коду.

- Він конвертується в проміжну мову, яка потім транслюється в машинну мову.

Компіляція Python-програми

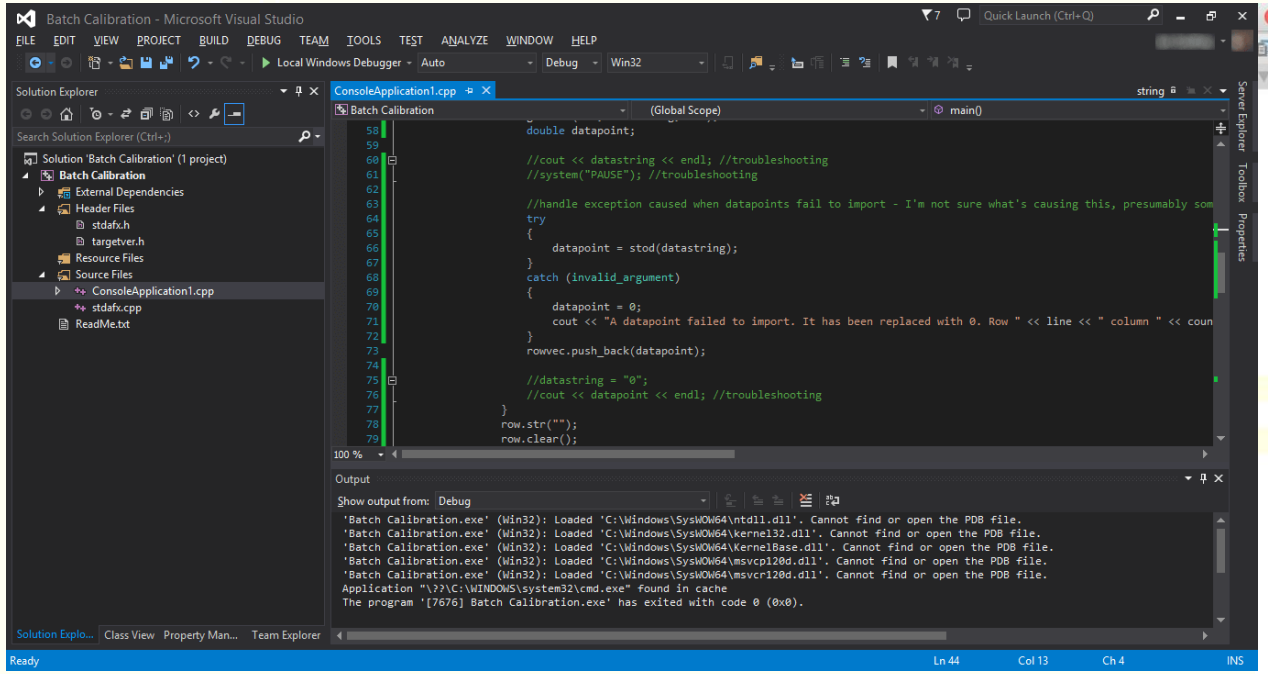


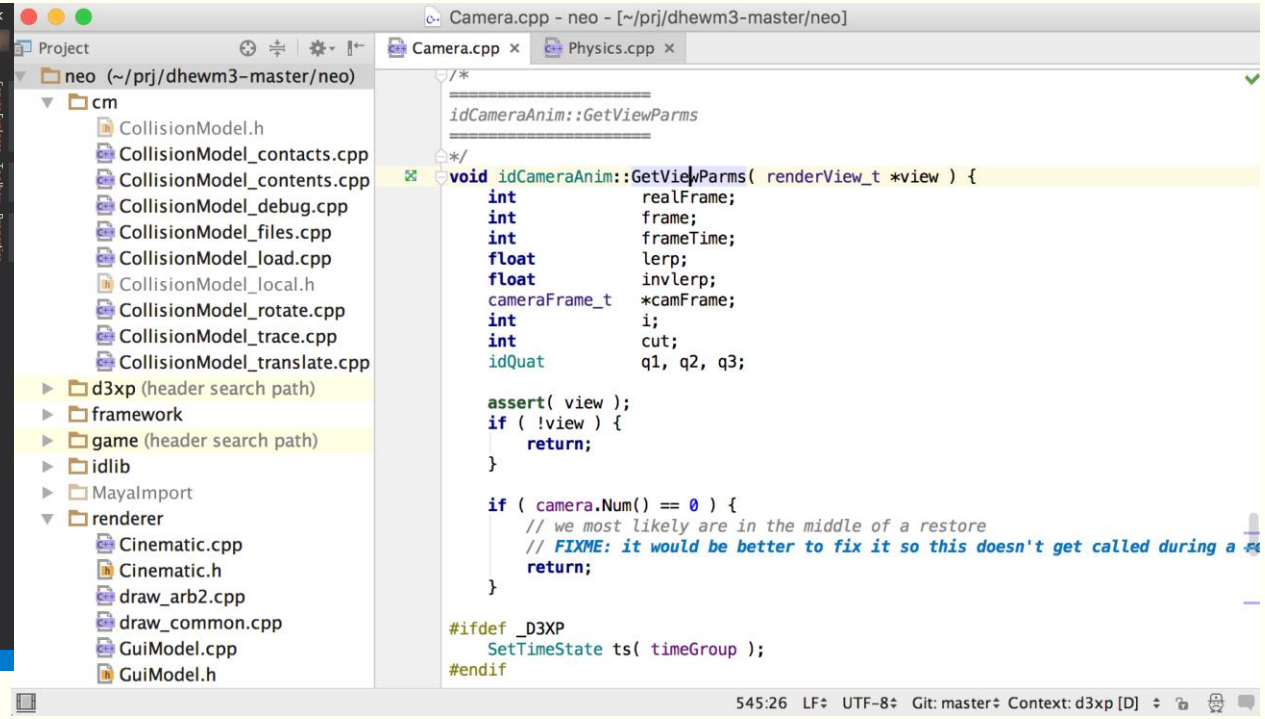
Етапи, які проходять програми, що написані мовою С



Фаза 1: створення програми

- Visual Studio (Windows, Mac OS X)
- Clion (Windows, Mac OS X, Linux)
- Eclipse (Windows, Mac OS X, Linux)





Фаза 2: препроцесинг

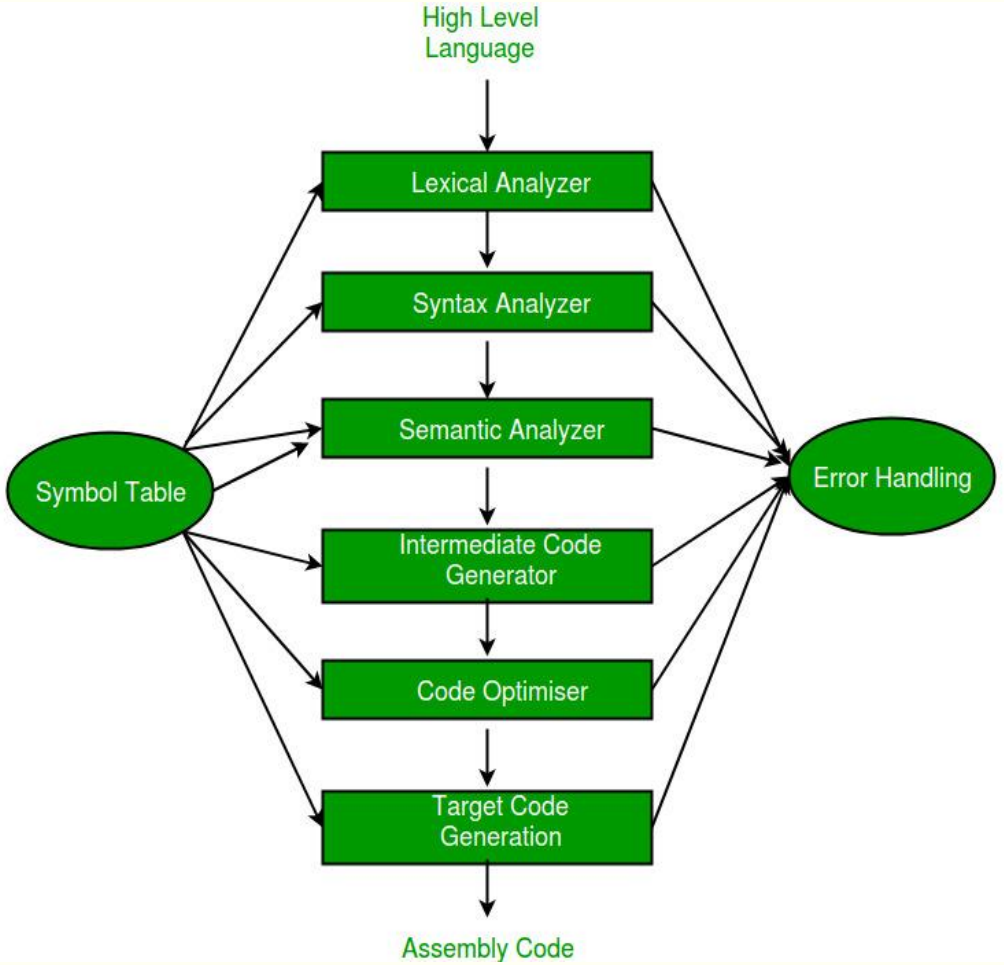
- Ви даєте команду ***скомпілювати*** програму.
  - Компілятор транслює первинний код програми в машинний (об'єктний) код.
- У системах на мові С перед фазою трансляції автоматично запускається ***програма-препроцесор***.
  - Препроцесор мови С виконує спеціальні команди, які називають ***директивами препроцесора***.
  - Вони визначають операції, які повинні бути виконані до компіляції.
  - Зазвичай це підключення інших файлів та заміна деяких ділянок тексту.
- Директиви препроцесора в С починаються з символу #. Поширені:
  - #include – включення файлів у проект
  - #define/#undef – оголошення/відміна констант і макросів
  - #ifdef / #else / #endif / #ifndef / #if / #elif – умовна компіляція
  - #pragma – інструкції компілятору в коді

Фаза 3: компіляція програми

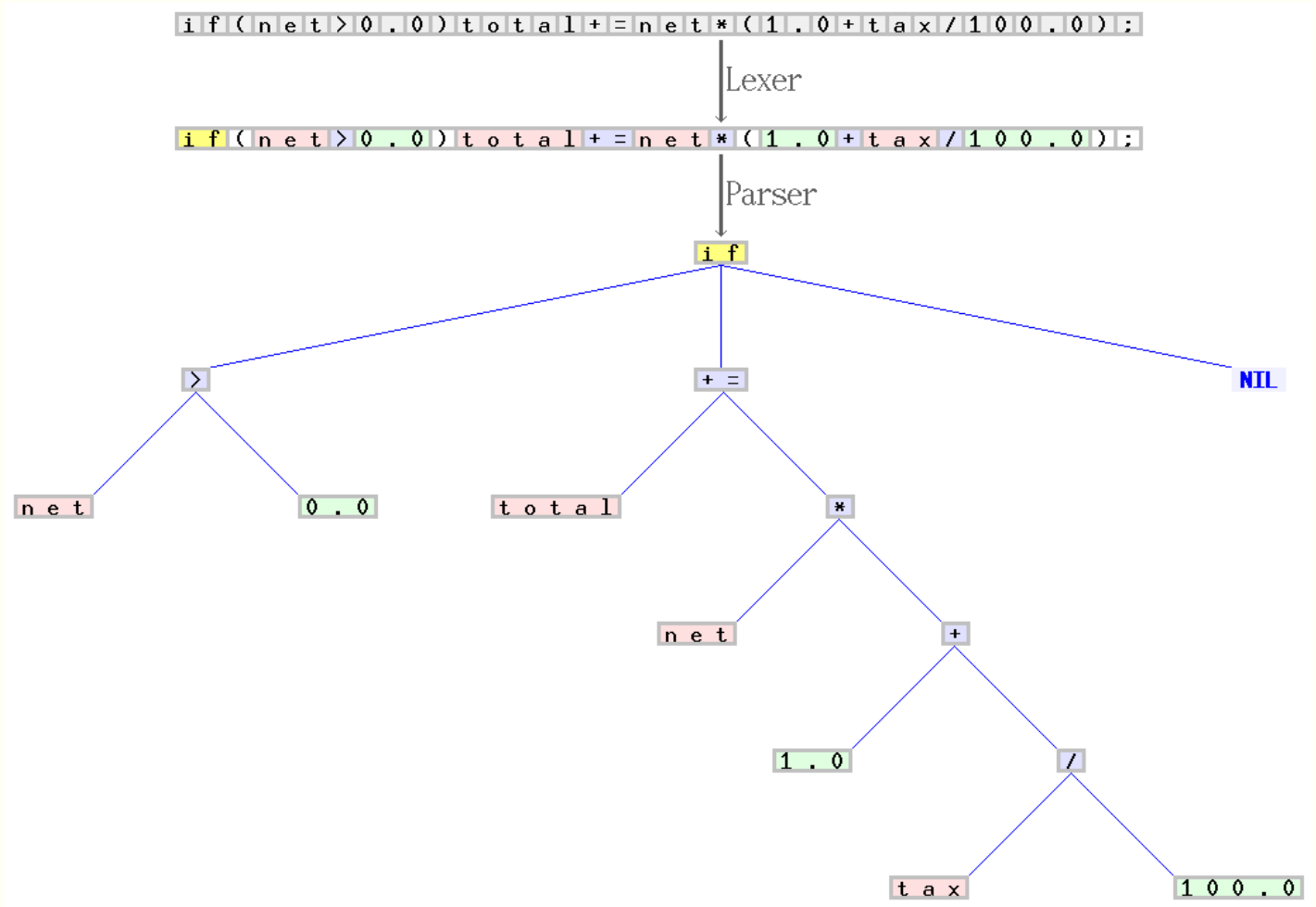
- У фазі 3 компілятор транслює програму мовою С в машинний код.
  - Якщо він не зможе розпізнати інструкцію, генерується синтаксична помилка, оскільки це вважається порушенням правил мови.
  - Щоб допомогти виправити помилкову інструкцію, компілятор виводить повідомлення.
  - Стандарт С не визначає конкретний текст повідомлень про помилки, тому в різних системах він може відрізнятись.
  - Синтаксичні помилки також називають помилками компіляції, або помилками часу компіляції.

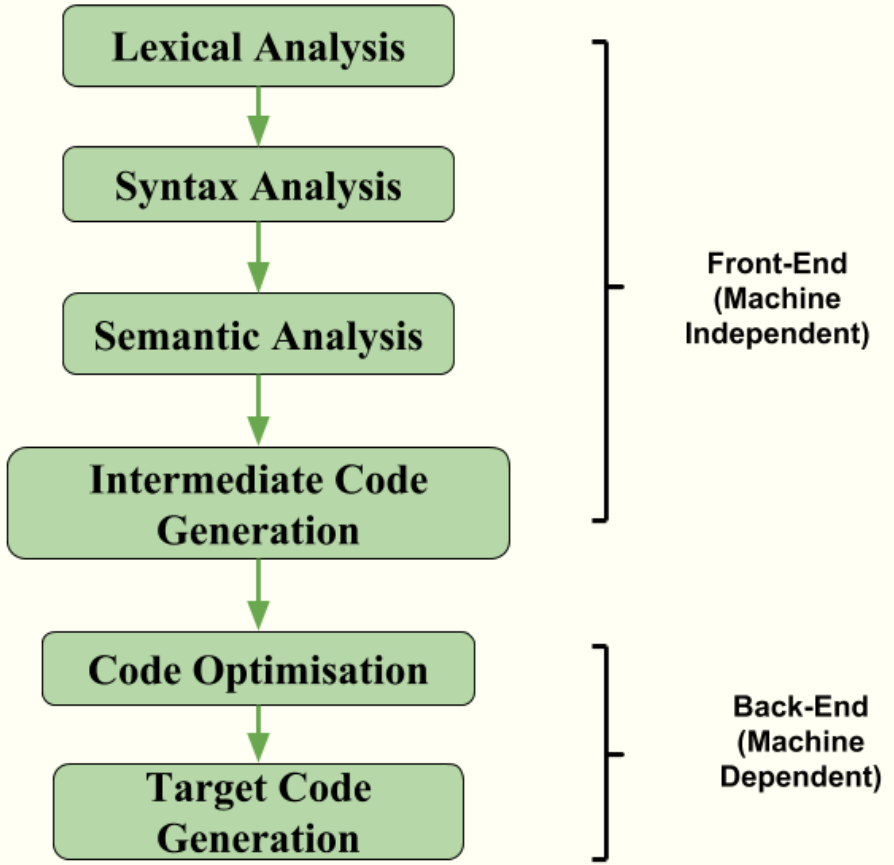
Детальніше про процес компіляції

- Компіляція відбувається у кілька етапів:
  - Лексичний аналіз (токенізація) – розбиття програми на токени (послідовність символів, що є граматичними одиницями мови програмування).
  - Синтаксичний аналіз (парсинг) – перевіряє граматичну правильність тексту програми, будує дерево розбору.
  - Семантичний аналіз – перевіряє, чи має смисл утворене дерево розбору, виконує перевірку типів, управляючих операторів тощо.
  - Генератор проміжного коду – перетворює дерево розбору на набір команд проміжної мови.
  - Оптимізатор коду – перетворює утворений код так, щоб він споживав менше ресурсів та працював швидше. Зміст коду не змінюється.
  - Генератор цільового коду – формує зрозумілий для машини код, виділяє регістри пам'яті, відбирає інструкції на виконання тощо.



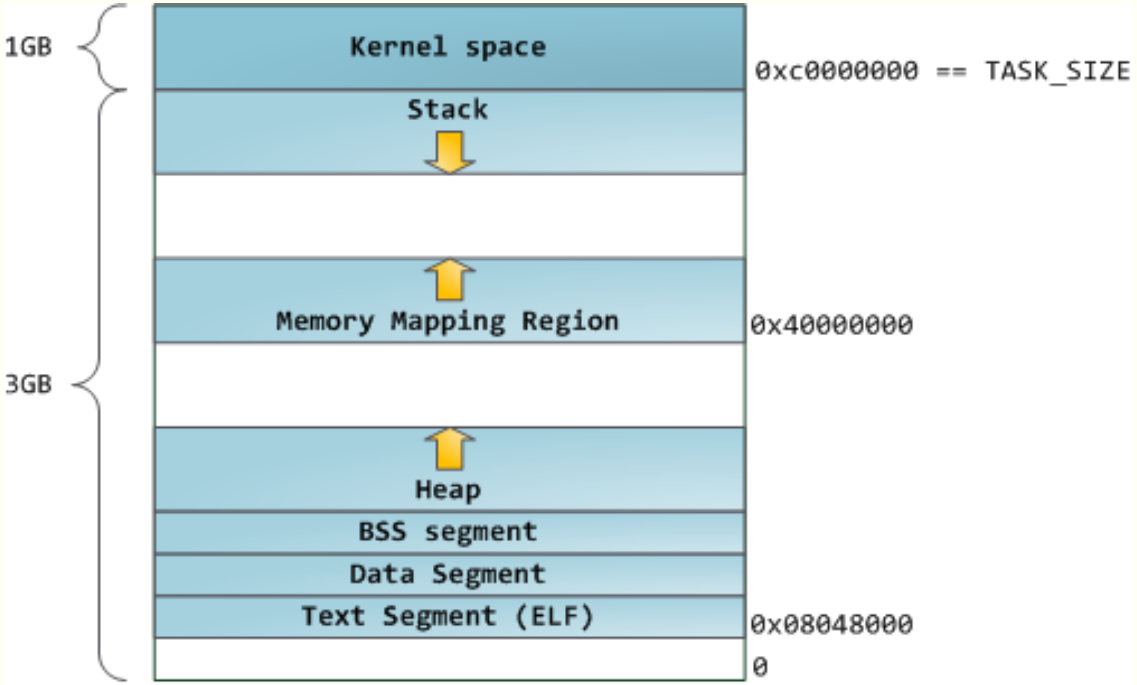
Етапи лексичного та синтаксичного аналізу





Фаза 4: компонування (linking)

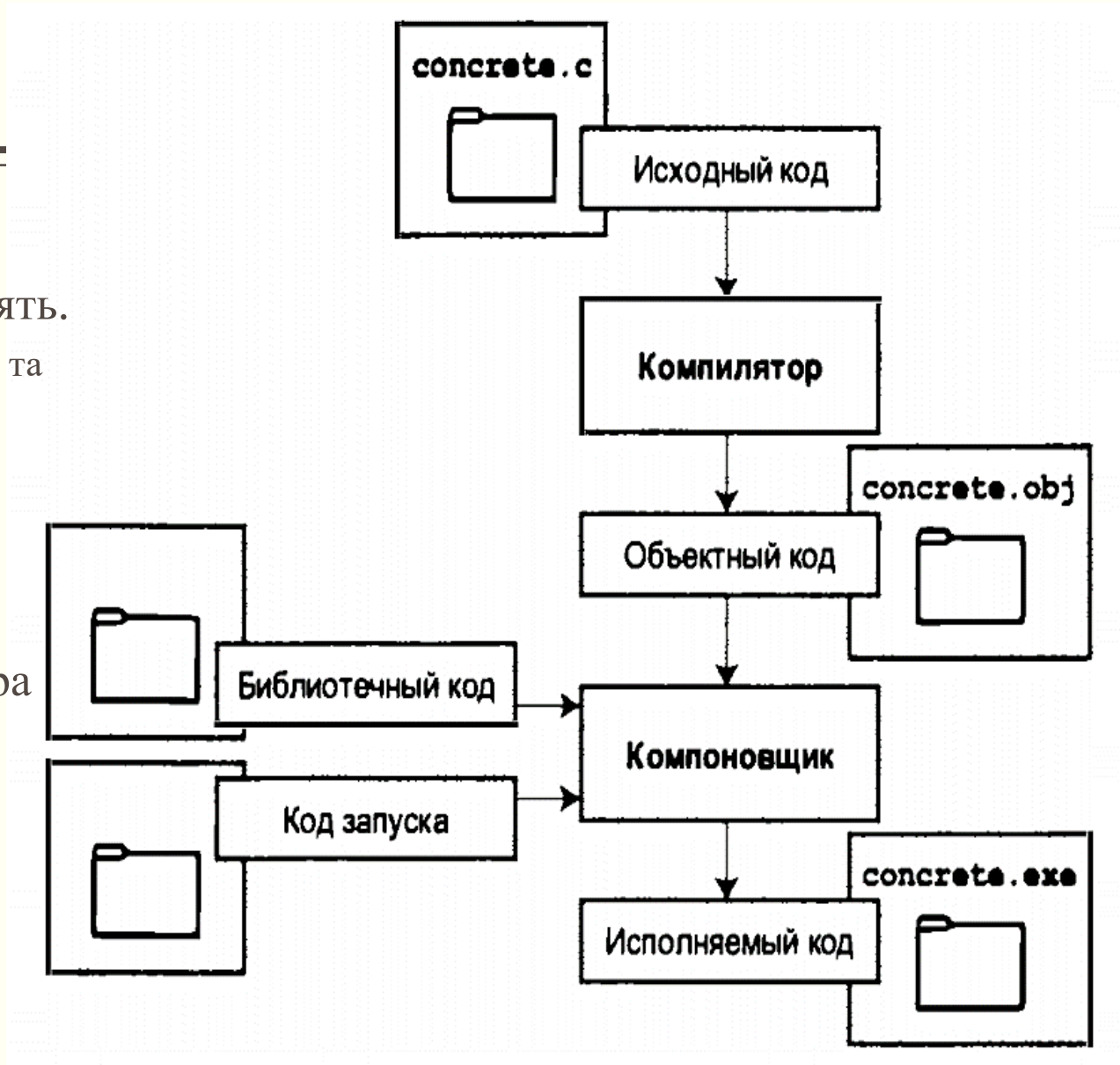
- Зазвичай С-програми містять посилання на функції, що знаходяться десь в іншому місці (стандартних бібліотеках або приватних бібліотеках групи програмістів, які працюють над спільним проектом).
  - Об'єктний код, створений компілятором, найчастіше містить «дірки», зарезервовані під відсутні частини.
  - Компонувальник зв'язує об'єктний код з кодом відсутніх функцій і генерує виконуваний образ (без «дірок»).



Фаза 5: завантаження
Фаза 6: виконання

- Перш ніж програма буде запущена, її необхідно завантажити в оперативну пам'ять.
  - Завантажувач читає виконуваний образ з диску та копіює його в оперативну пам’ять.
  - Також завантажувач робить завантаження необхідних спільних бібліотек.

- Комп'ютер під керуванням свого процесора (CPU) виконує програму інструкція за інструкцією.




Компилятор
Мой код
140 ошибок


Компилятор компилирует программу
Мой друг, который не программист и видит 64 ворнинга
Я, видящий 0 ошибок

Що вивчають у мовах програмування?

ДЯКУЮ ЗА УВАГУ!

Наступне запитання: