Лекція

Основи програмування –

Мови програмування

ПЛАН

1. Програми та програмування
2. Мови програмування
3. Основні етапи виконання завдання на комп'ютері
4. Об'єктно-орієнтоване програмування
5. Інтегровані середовища розробки програмного забезпечення

Мова програмування

Мова програмування (англ. *Programming language*) — система позначень для опису алгоритмів та структур даних, певна штучна формальна система, засобами якої можна виражати алгоритми. Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми і дії, які виконує виконавець (комп'ютер) під її управлінням.

З часу створення перших програмованих машин було створено понад дві з половиною тисячі мов програмування. Щороку їх кількість поповнюється новими. Деякими мовами вміє користуватись тільки невелике число їх власних розробників

Елементи

Синтаксис

Синтаксис мови програмування визначає те, як буде виглядати програма на цій мові, зокрема, як пишуться оператори, оголошення і інші мовні конструкції. Наприклад, оголошення масиву V з десяти цілочислових елементів в на мові C буде виглядати так: int V[10];

На мові Pascal: V: array [0…9] of integer

Типи даних

Область зберігання даних в апаратній частині комп'ютера (пам'ять, регістри і зовнішні запам'ятовуючі пристрої) зазвичай мають доволі просту структуру в вигляді послідовності бітів, згрупованих в байти або слова.

Проте в віртуальному комп'ютері, як правило, організовано більш складним чином — в різні моменти виконання програми використовуються такі форми зберігання даних, як стеки, масиви, числа, символьні рядки та інші.

Один або декілька однотипних елементів даних, об'єднаних в одне ціле в віртуальному комп'ютері в певний момент виконання програми, прийнято називати *об'єктом даних*.

При виконанні програми існує багато об'єктів даних різних типів. *Тип даних* — це деякий клас об'єктів даних разом з набором операцій для створення і роботи з ним.

В кожній мові програмування є певний набір вбудованих примітивних типів даних. Додатково в мові можуть бути передбачені засоби, що дозволяють програмісту визначати нові типи даних.

Класифікація мов програмування

Мови класифікують за такими критеріями:

Рівень абстракції

Мови програмування високого рівня оперують сутностями ближчими людині, такими як об'єкти, змінні, функції. Мови програмування нижчого рівня оперують сутностями ближчими машині: байти, адреси, інструкції. Текст програми на мові високого рівня зазвичай набагато коротший ніж текст такої самої програми на мові низького рівня, проте програма має більший розмір.

Область застосування

Універсальні та спеціалізовані. Спеціалізовані мови теж бувають Тьюрінг-повні, та все ж їх область застосування обмежена, як наприклад у мови shell.

Підтримувані парадигми програмування

Об'єктно-орієнтовані, логічні, функційні, структурні…

Імперативні мови базуються на ідеї змінної, значення якої змінюється присвоєнням. Вони називаються імперативними (лат. *imperative* - наказовий), оскільки складаються із послідовностей команд, які звичайно містять присвоєння змінних <назва_змінної> = <вираз>, де вираз може посилатися на значення змінних присвоєних попередніми командами.

Мови програмування низького рівня

Перші комп'ютери доводилось програмувати двійковими машинними кодами. Проте програмувати таким чином — доволі трудомістка і важка задача. Для спрощення цієї задачі почали з'являтися мови програмування низького рівня, які дозволяли задавати машинні команди в більш зрозумілому для людини вигляді. Для перетворення їх у двійковий код були створені спеціальні програми — **транслятори**.

Машинний код		Асемблер	
0005	B4 09	7	mov AH, 09h
0007	BA0000r	8	mov DX, offset msg
00A	CD 21	9	int 21 h

Приклад машинного коду і представлення його на асемблері

Транслятори

Транслятори поділяються на:

компілятори — перетворюють текст програми в машинний код, який можна зберегти і після цього використовувати уже без компілятора (прикладом є виконувальні файли з розширенням *.exe).

інтерпретатори — перетворюють частину програми в машинний код, виконують і після цього переходять до наступної частини. При цьому щоразу при виконанні програми використовується інтерпретатор.

Мова низького рівня

Прикладом мови низького рівня є **асемблер**. Мови низького рівня орієнтовані на конкретний тип процесора і враховують його особливості, тому для перенесення програми на асемблері на іншу апаратну платформу її потрібно майже цілком переписати. Певні відмінності є і в синтаксисі програм під різні компілятори. Щоправда, центральні процесори для комп'ютерів фірм AMD та Intel практично сумісні і відрізняються лише деякими специфічними командами. А ось спеціалізовані процесори для інших пристроїв, наприклад, відеокарт, телефонів містять суттєві відмінності.

Переваги

За допомогою мов низького рівня створюються

Недоліки

1). Програміст, що працює з мовами низького рівня, має бути високої кваліфікації, добре розуміти будову мікропроцесорної системи, для якої створюється програма. Так, якщо програма створюється для комп'ютера, потрібно знати будову комп'ютера і, особливо, влаштування і особливості роботи його процесора.

2). Результуюча програма не може бути перенесена на комп’ютер або пристрій з іншим типом процесора.

3). Значний час розробки великих і складних програм.

Мови низького рівня, як правило, використовують для написання невеликих системних програм, драйверів пристроїв, модулів стиків з нестандартним обладнанням, програмування спеціалізованих мікропроцесорів

Мови програмування високого рівня

Мови програмування високого рівня можна сказати є більш зрозумілими людині, ніж комп'ютеру. Особливості конкретних комп'ютерних архітектур в них не враховуються, тому створені програми легко переносяться з комп'ютера на комп'ютер. Здебільшого достатньо просто перекомпілювати програму під певну комп'ютерну архітектурну та операційну систему. Розробляти програми на таких мовах значно простіше і помилок допускається менше. Значно скорочується час розробки програми, що особливо важливо при роботі над великими програмними проектами.

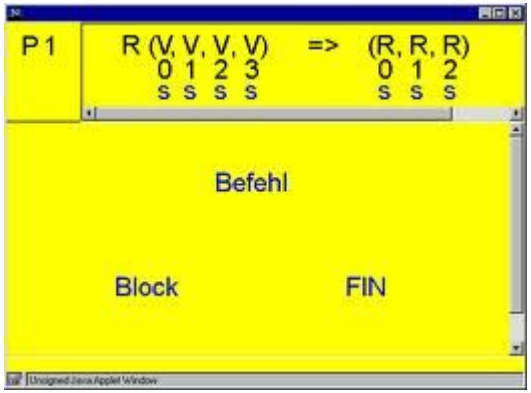
Недоліком мов високого рівня є більший розмір програм порівняно з програмами на мові низького рівня. Тому в основному мови високого рівня використовуються для розробок програмного забезпечення комп'ютерів, і пристроїв, які мають

рівня

- Адресна мова програмування
- Фортран
- Кобол
- Алгол
- Pascal
- Java
- C
- C++
- C#
- Objective C
- Smalltalk
- Delphi

Історія

Першу мову програмування високого рівня — Планкалькюль (нім. *Plankalkül*) спроектував німець Конрад Цузе у 1945 році, але вона не мала комп'ютерної реалізації і не одержала уваги, хоча мала дуже потужні на



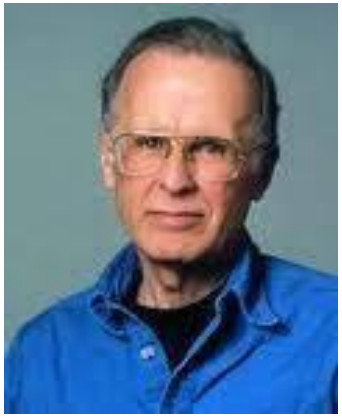
Конрад Цузе

Історія

UNIVAC I

У кінці 40-их — початку 50-их застосовувалися інтерпретовані системи кодування, коли певні команди мови програмування кодувалися числами, які уже інтерпретувалися машинним кодом. Ці системи називалися «автоматичним програмуванням», були простішими для програмування, ніж машинні коди, але могли мати значно меншу (до 50 разів) швидкодію, через що часто надавали перевагу машинним кодам. До таких систем належали — Short Code для BINAC (1949) і UNIVAC I (1952), Speedcoding для IBM 701, розроблена Джоном

IBM 701

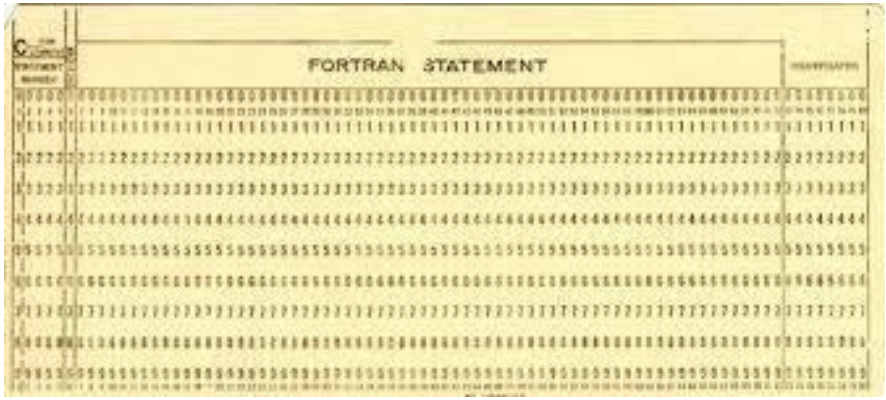


Джон Бекус

Першою широковживаною компільованою мовою став розроблений групою Джона Бекуса Фортран, анонсований у 1954 році і випущений у 1957 для IBM 704.

Основним призначенням Фортрану були швидкі наукові обчислення, оголошувалося що швидкодія згенерованого компілятором коду майже не відрізянтиметься від машинного коду написаного вручну.

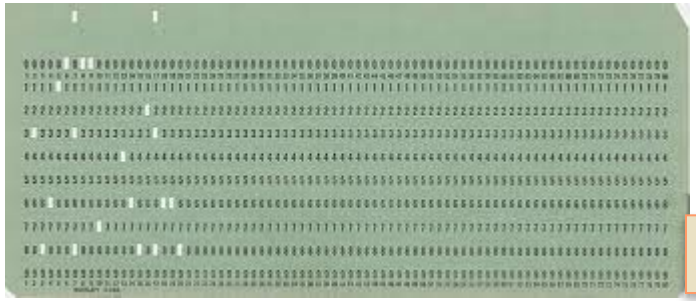
Уже у квітні 1958 близько половини програм для IBM 704 були написані на Фортрані. Випущений у 1958 році Фортран II дозволяв незалежну компіляцію підпрограм, що дозволило створювати більші програми, оскількі низька надійність IBM



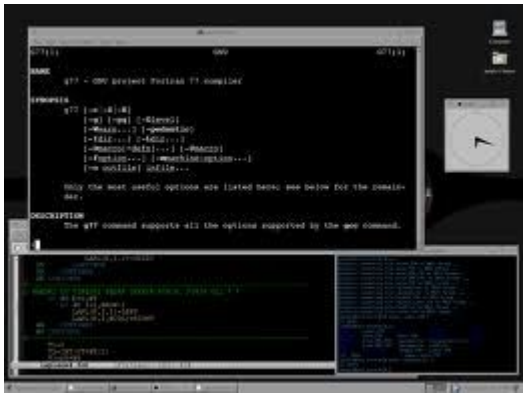
IBM 704

Розроблений у 1960—1962 роках Фортран IV був однією з найпоширеніших мов того часу і лишався стандартною версією Фортрану до появи у 1978 році Фортрану 77.

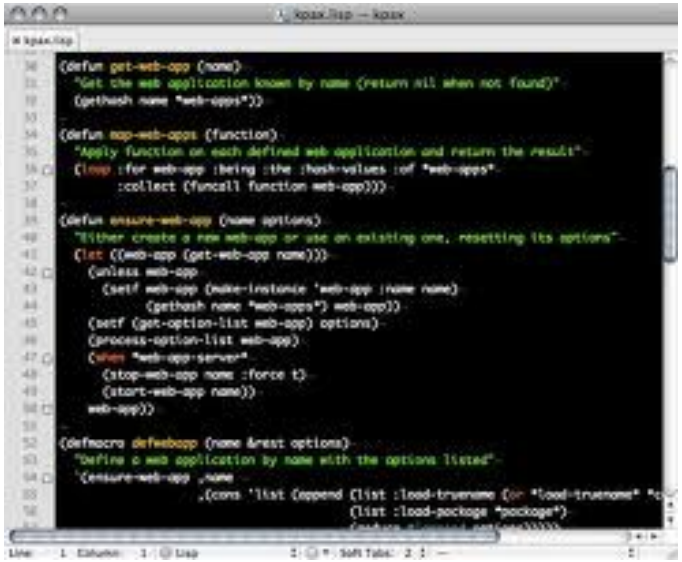
У 1958 році у MIT розробили LISP — першу функційну мову, яка понад чверть століття домінувала у програмуванні задач



Фортран IV



Фортрану 77



LISP

У кінці 1950-их почали розроблятися різні мови програмування.

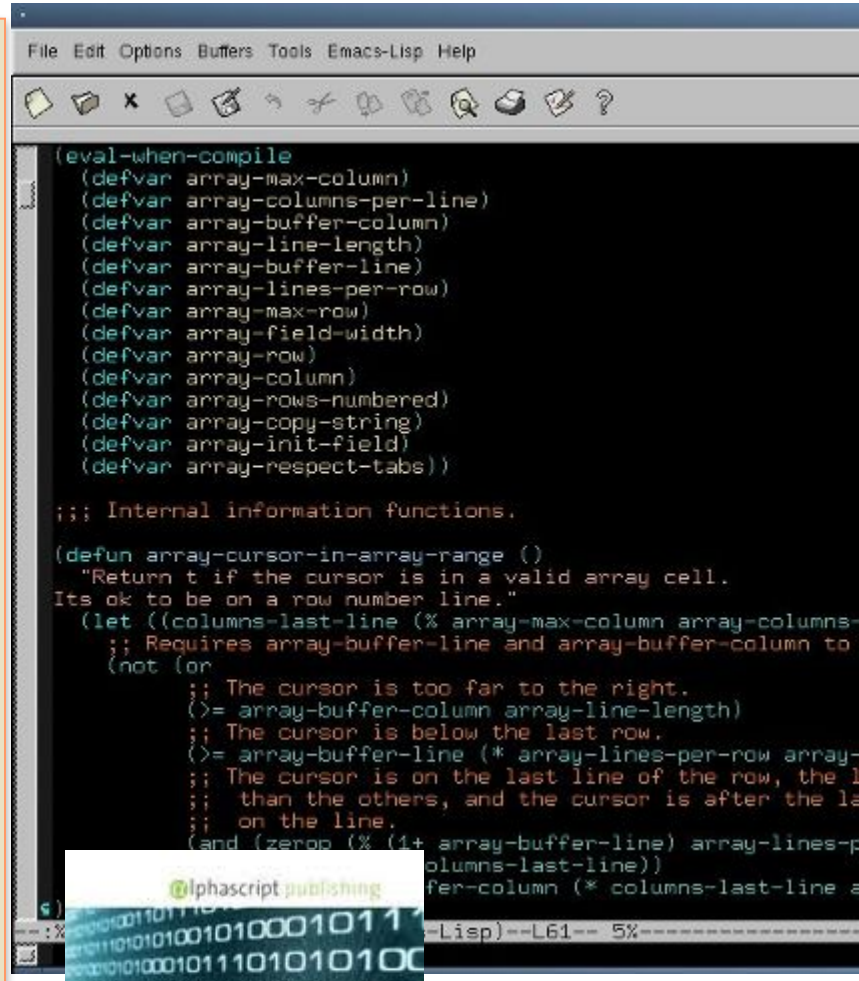
У 1958 році декілька значних груп комп'ютерних користувачів у США, включаючи SHARE — групу науковців-користувачів IBM і USE (UNIVAC Scientific Exchange, група науковців-користувачів UNIVAC) запропонували ACM заснувати робочу групу зі створення універсальної мови програмування.

Також ще у 1955 році німецьке Товариство прикладної математики і механіки (GAMM) заснувало комітет зі створення універсальної мови програмування.

У кінці травня 1958 року було проведено зустріч у Цюриху між ACM і GAMM, на матеріалах якої у грудні опубліковано ALGOL 58 Report.

На його основі було створено 3 значні реалізації — MAD (1961), NELIAC (1963), JOVIAL (1963). З них лише JOVIAL отримав поширення, ставши на чверть століття офіційною мовою програмування у Військово-морських силах США.

SHARE і IBM почали створення власної реалізації ALGOL, але припинили, врахувавши витрати на створення і просування Фортрану.



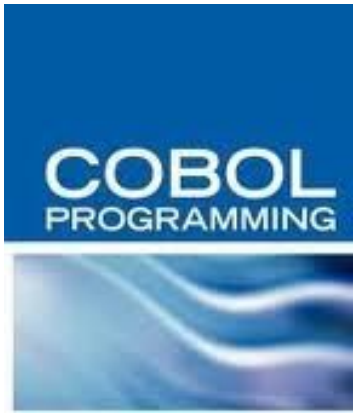
Впродовж 1959 року ALGOL 58 широко обговорювався, була запропонована нотація для опису синтаксису мов програмування — форма Бекуса-Наура. У 1960 проведено чергову зустріч і опубліковано ALGOL 60 Report.

ALGOL вплинув на багато мов програмування і став стандарною мовою для публікації алгоритмів, але через ряд причин не одержав широкого поширення — він був заскладним, і не було реалізацій, які підтримували його повністю, відсутність стандартного введення-виведення привела до появи різних несумісних реалізацій, деякі неоднозначності опису мови так і не були розв'язані. Також широкого вжитку уже набув Фортран, і IBM не підтримала ALGOL.

У 1959 році була проведена зустріч у Пентагоні для створення мови CBL (Common Business Language), засновано комітет з його створення, і у 1960 опубліковано початкову специфікацію COBOL 60, який невдовзі

```
IDENTIFICATION DIVISION.

PROGRAM-ID. COBOL-DEMO.
AUTHOR.     M. A. COVINGTON.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-PC.
OBJECT-COMPUTER. IBM-PC.

DATA DIVISION.

WORKING-STORAGE SECTION.
77  SUM    PICTURE IS 9999999, USAGE IS COMPUTATIONAL.
77  X      PICTURE IS 9999999, USAGE IS COMPUTATIONAL.

PROCEDURE DIVISION.

START-UP.
    MOVE 0 TO SUM.
GET-A-NUMBER.
    DISPLAY "TYPE A NUMBER:" UPON CONSOLE.
    ACCEPT X FROM CONSOLE.
    IF X IS EQUAL TO 0 GO TO FINISH.
    ADD X TO SUM.
    GO TO GET-A-NUMBER.
FINISH.
    DISPLAY SUM UPON CONSOLE.
    STOP RUN.
```

FIGURE 60. COBOL program

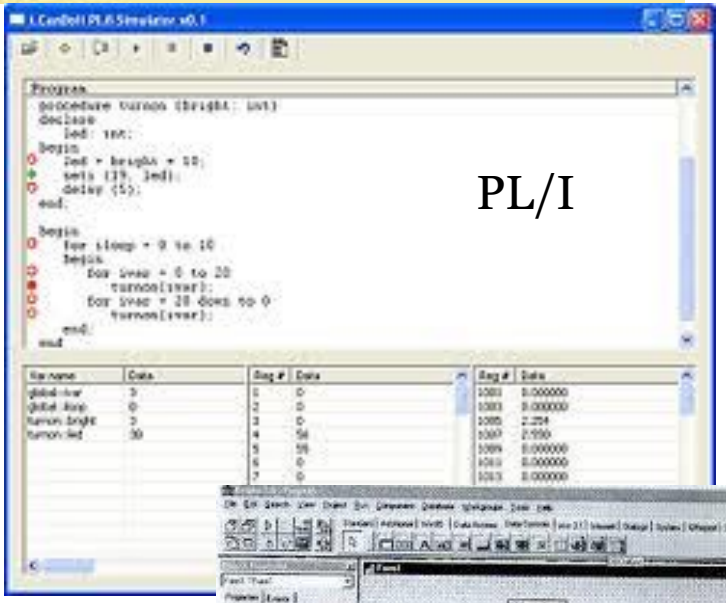


У 1964 році було створено спрощену мову BASIC (Beginners All-purpose Symbolic Instruction Code) для навчання програмуванню студентів, які переважно спеціалізувалися у вільних мистецтвах, а не технічних науках.

Тоді як науковці переважно використовували Фортран, а бізнес — COBOL, у 1963 році в IBM вирішили створити універсальні платформу IBM/360 і мову програмування. У стислі терміни до 1965 року було розроблено мову PL/I, яка поєднувала можливості Фортран, ALGOL і COBOL, і виявилась заскладною, хоча і була у широкому вжитку у 1970-их у наукових і бізнес задачах, також її підмножини (PL/C, PL/CS) використовувалися для навчання програмуванню.

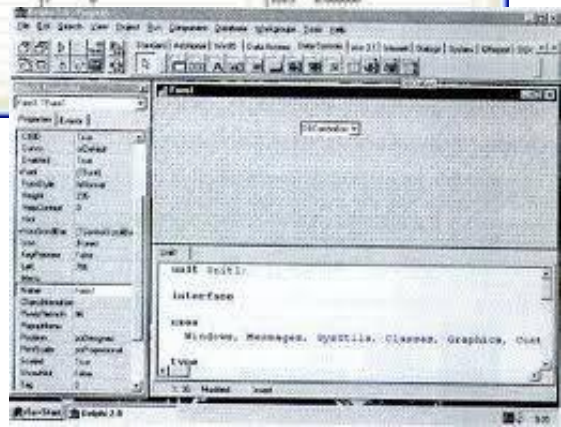
```
 10 INPUT "What is your name: "; U$
 20 PRINT "Hello "; U$
 25 REM
 30 INPUT "How many stars do you want: "; N
 35 S$ = ""
 40 FOR I = 1 TO N
 50 S$ = S$ + "*"
 55 NEXT I
 60 PRINT S$
 65 REM
 70 INPUT "Do you want more stars? "; A$
 80 IF LEN(A$) = 0 THEN GOTO 70
 90 A$ = LEFT$(A$, 1)
100 IF (A$ = "Y") OR (A$ = "y") THEN GOTO 30
110 PRINT "Goodbye ";
120 FOR I = 1 TO 200
130 PRINT U$; " ";
140 NEXT I
150 PRINT
```

BASIC



PL/I

SIMULA 67

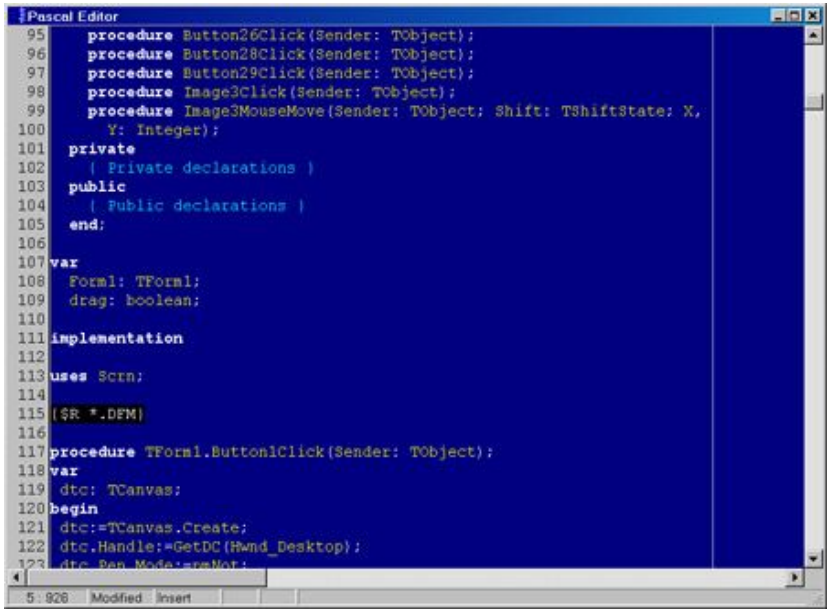


У 1965 році Ніклаус Вірт і Тоні Хоар запропонували комітету з розвитку мови ALGOL свою версію, яку згодом назвали ALGOL-W і застосовували для навчання в деяких університетах. Пропозиція була відхилена через незначну кількість змін, на користь значно складнішого ALGOL 68.

У ALGOL 68 з'явилися визначення структур данних і динамічні масиви. ALGOL 68 став першою мовою із формальною специфікацією, яка однак була складною для

Ніклаус Вірт

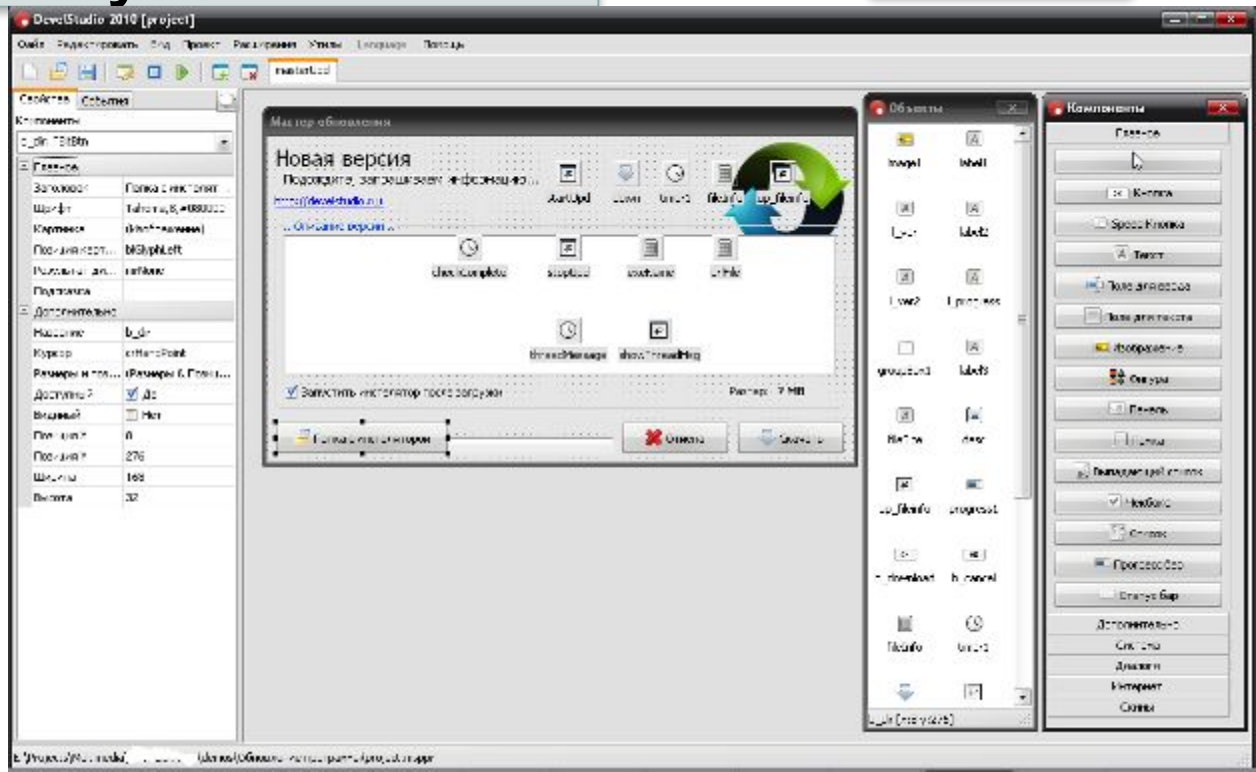
Тоні Хоар



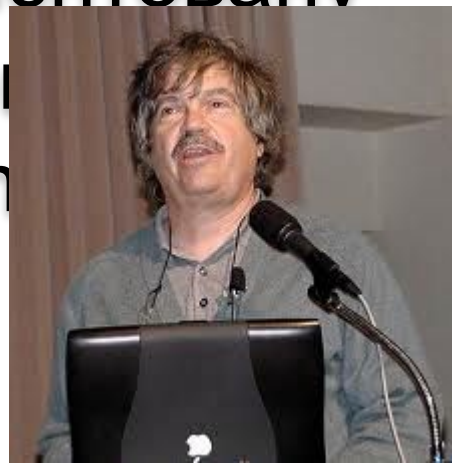
У 1972 році Деніс Річі розробив у Bell Labs мову С.

Тоді ж у Марселі створено інтерпретатор мови Пролог - першої і найвідомішої мови логічного програмування.

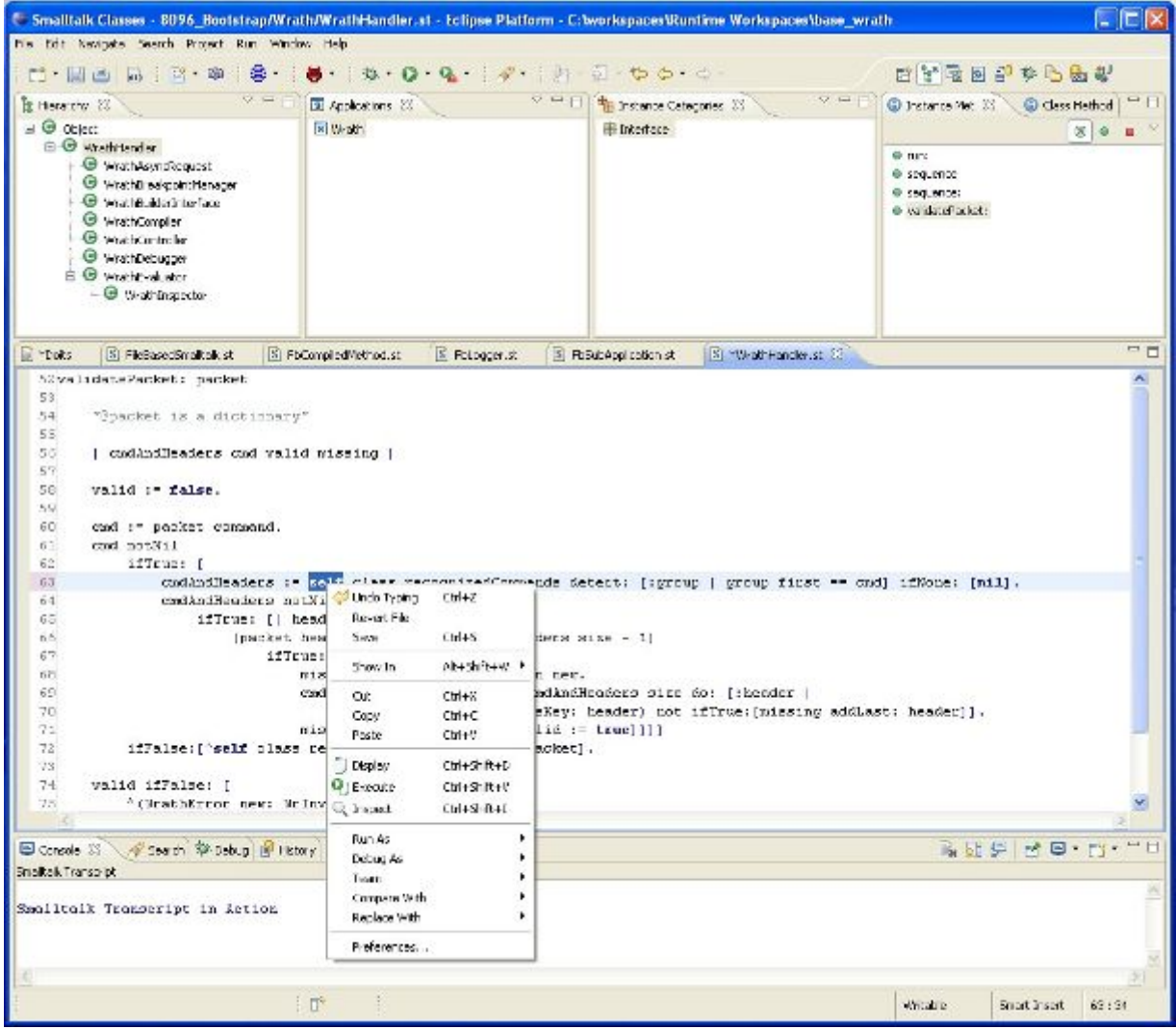
Деніс Річі



Алан Кей у Xerox PARC розробив першу широко вживану об'єктно-орієнтовану мо
Sm



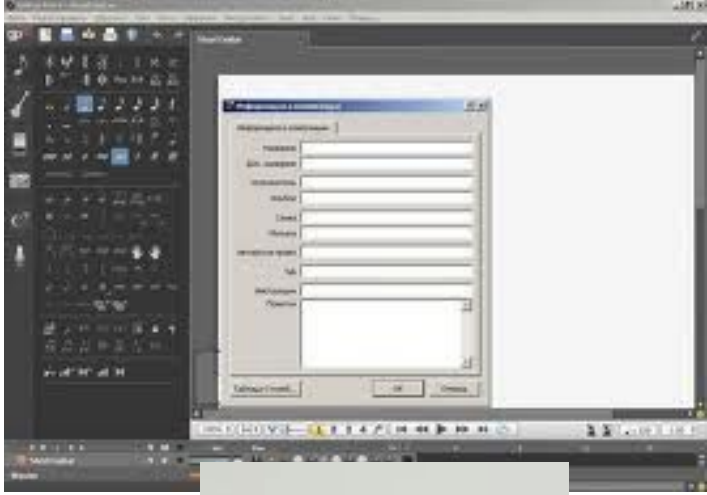
Алан Кей



У 1973 Робін Мілнер в Единбурзькому університеті створив ML.

У 1975 році в Массачусетському технологічному інституті описано спрощений діалект мови Лісп — Scheme.

У 1976 випущено мову для статистичного програмування S, на базі якої в 1993 році створено R.



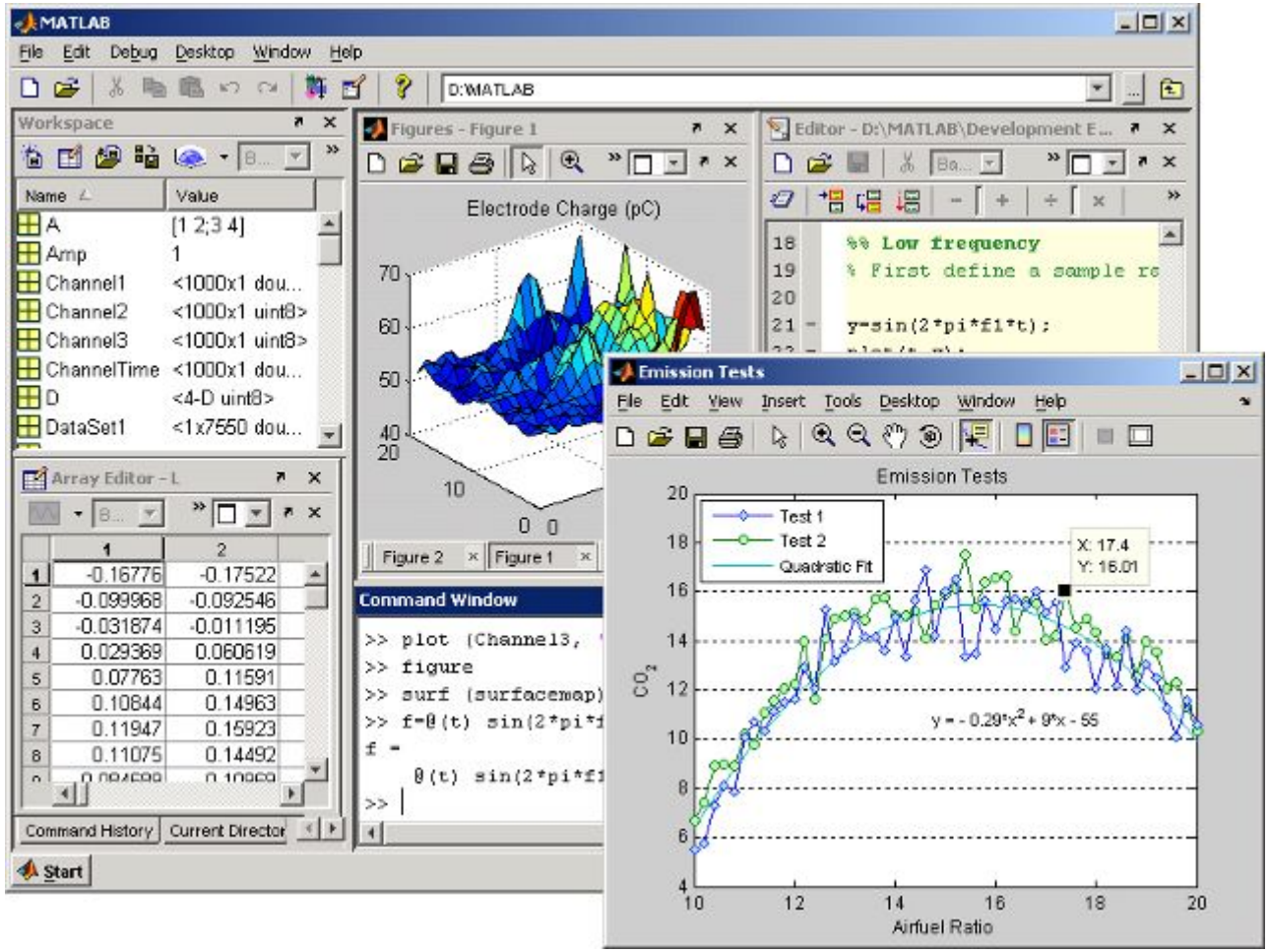
Професор Робін Мілнер

У 1975 Міністерство оборони США утворило міжнародну групу для створення нової мови програмування для власних потреб, конкурс у 1979 виграла мова Ада.

Ада

dBASE II

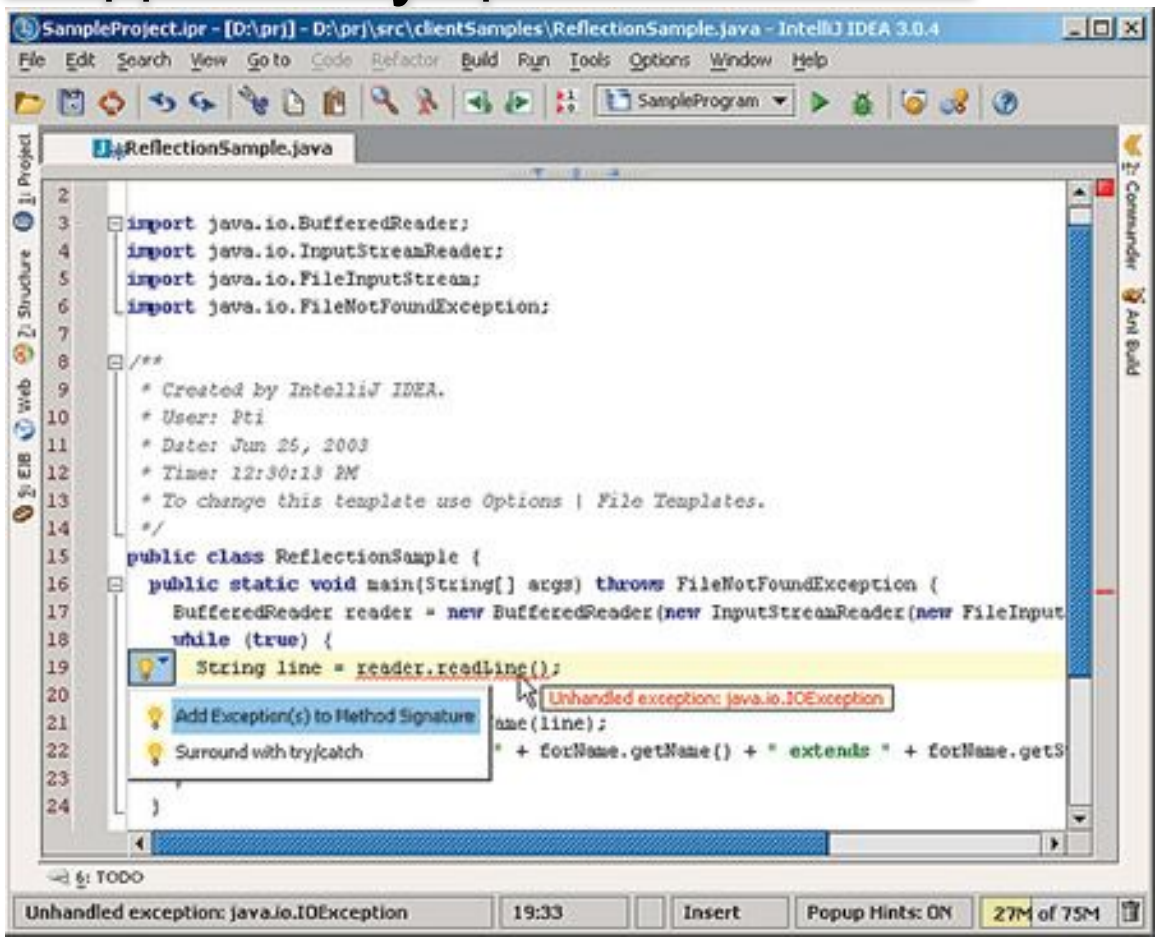
діалектів Ліспу створено Common Lisp. Випущено MATLAB



MATLAB

У 1985 році Б'ярн Страуструп опублікував реалізацію мови C++. Тоді ж випущено AWK.

Б'ярн Страуструп

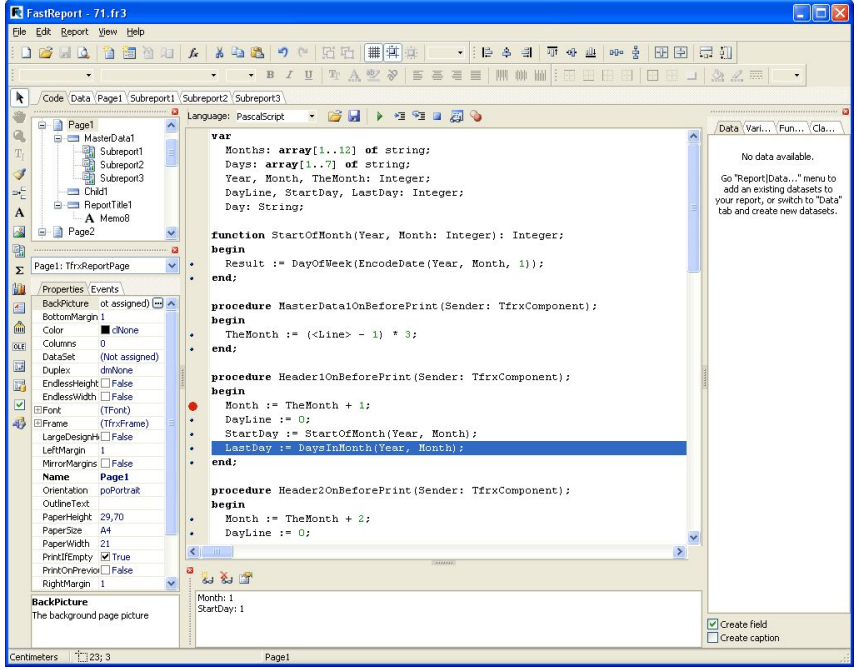


Середовище мови C++.

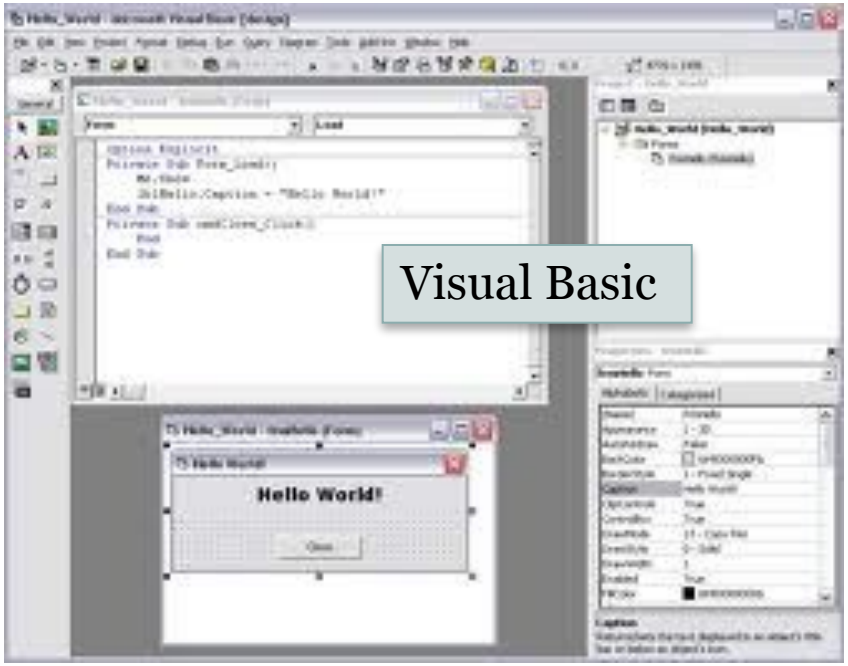
У 1986 році опублікована мова Objective-C і створено Erlang. Тоді ж Borland і Apple незалежно створили об'єктно-орієнтоване розширення мови Pascal - Object Pascal.

У 1987 році створено Perl.

У 1990 році опубліковано Standard ML і Haskell.



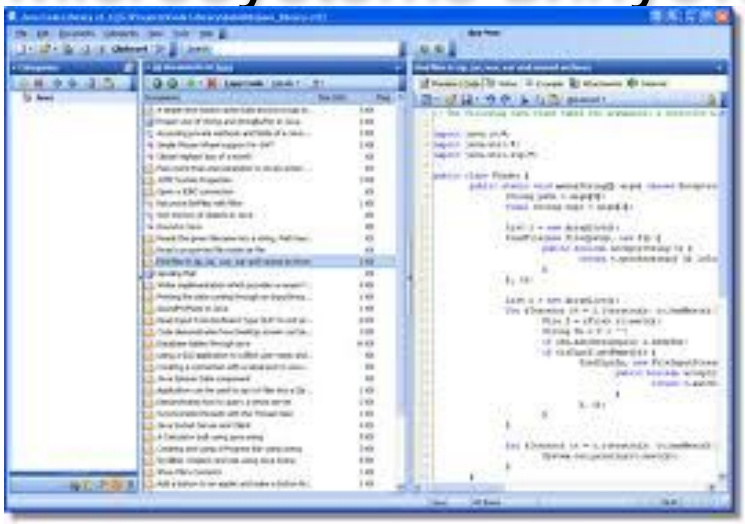
Object Pascal



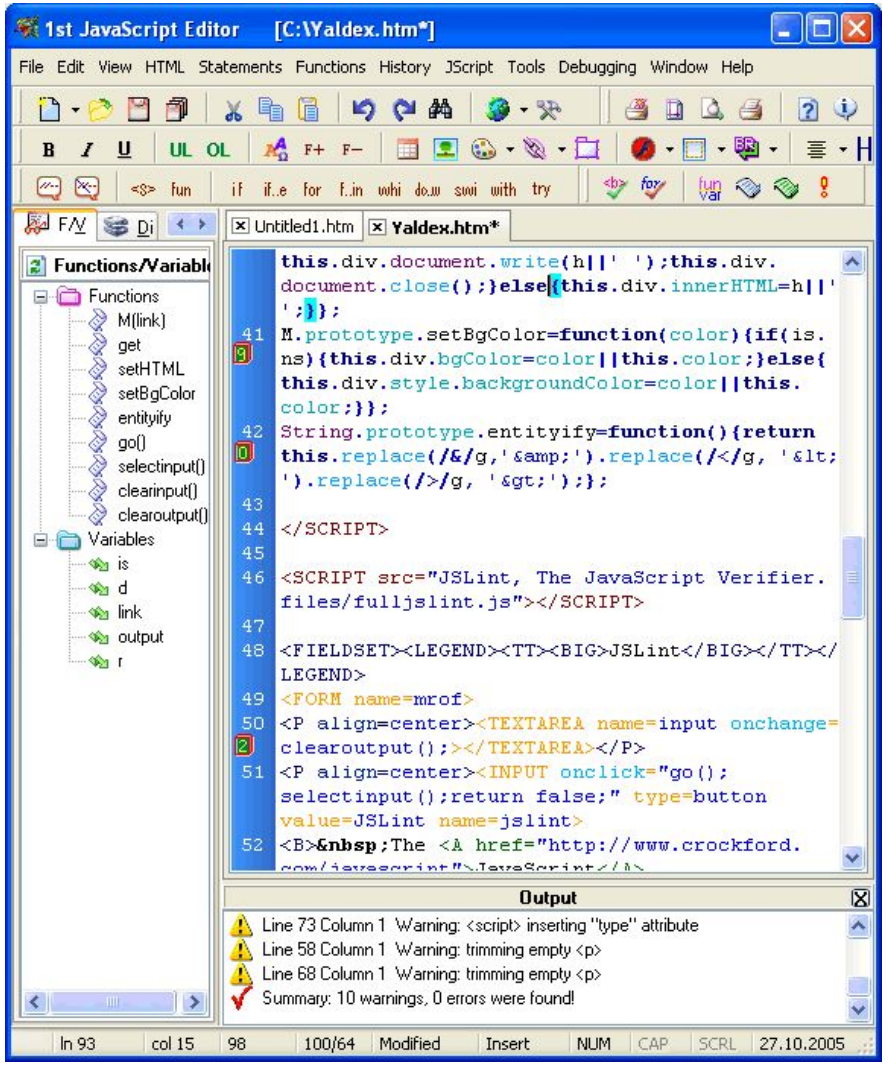
У 1992 випущено Oracle 7 з підтримкою PL/SQL

У 1993 році створено Lua.

У 1995 році Sun Microsystems випустила



Java

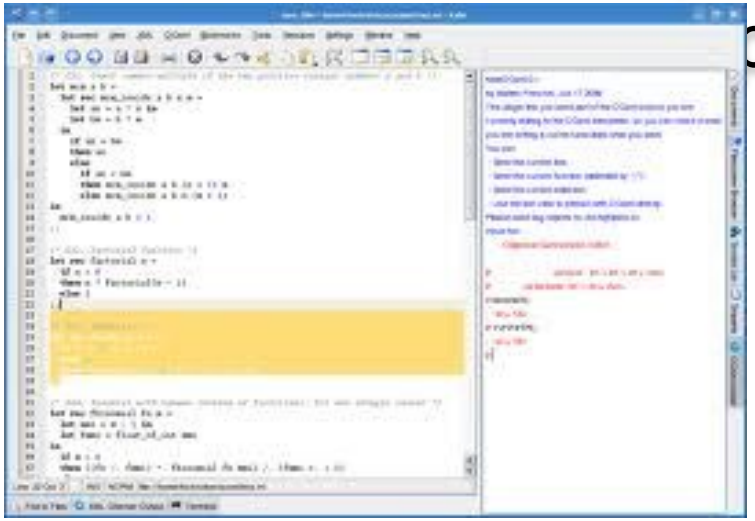


JavaScript

У 1996 році створено OCaml.

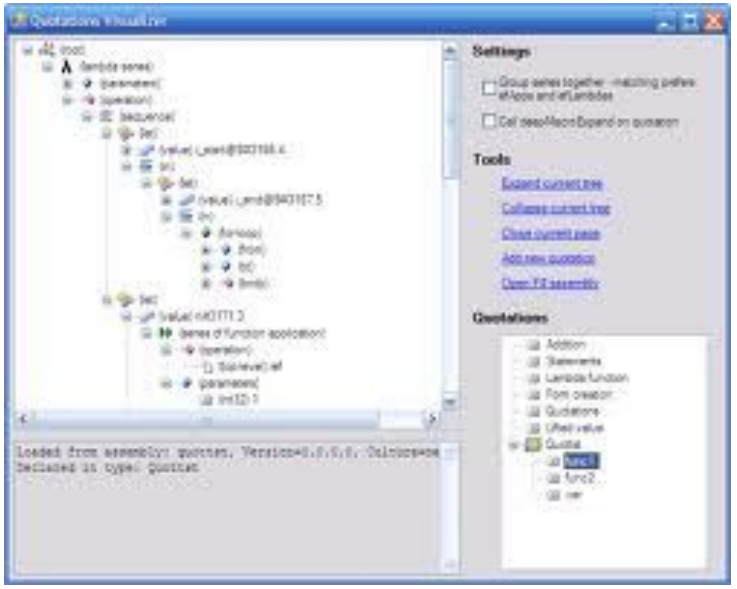
У 2001 році створено C#.

У 2002 році створено



OCaml

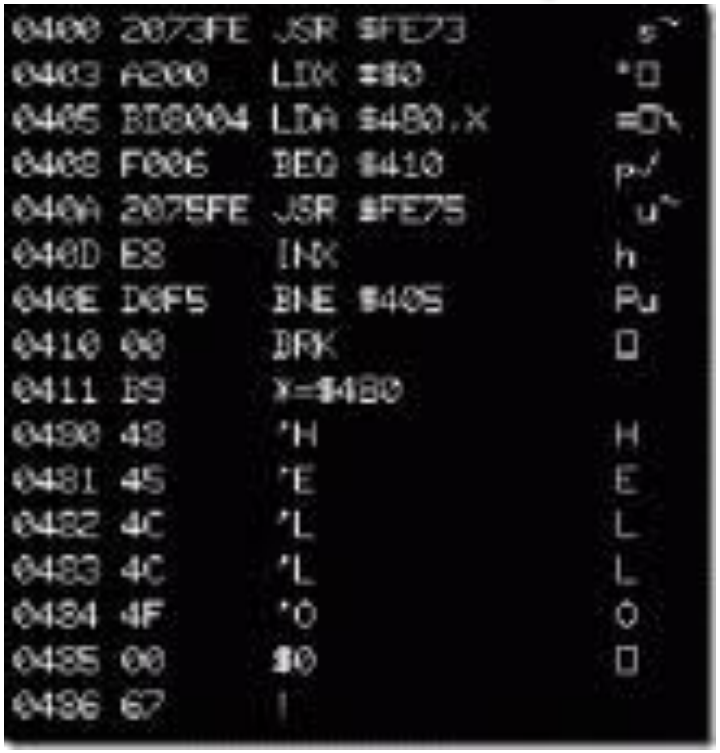
C#



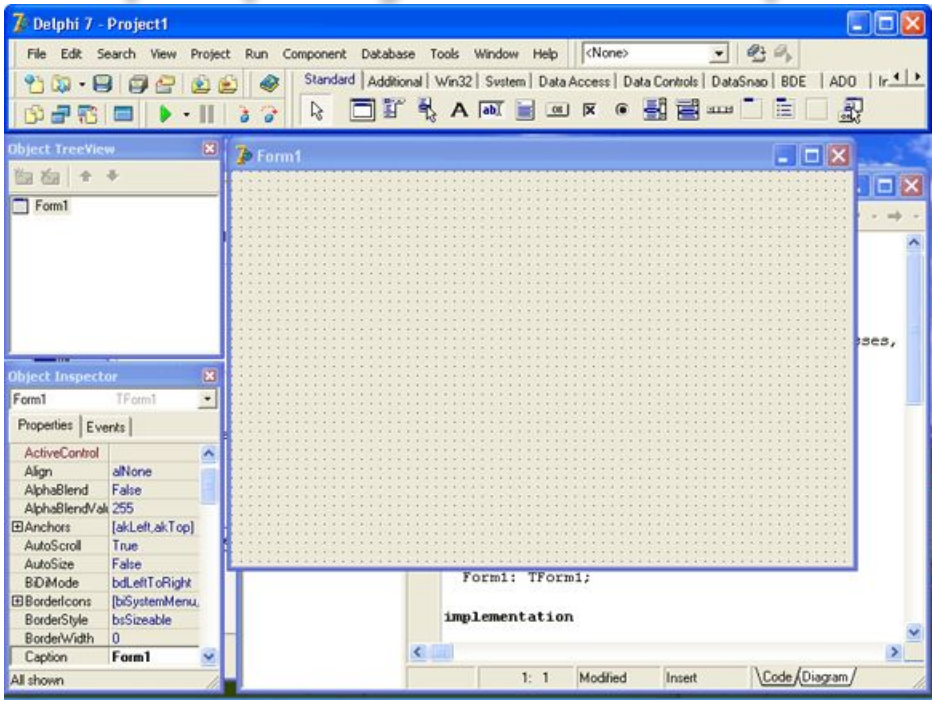
F#

Scala

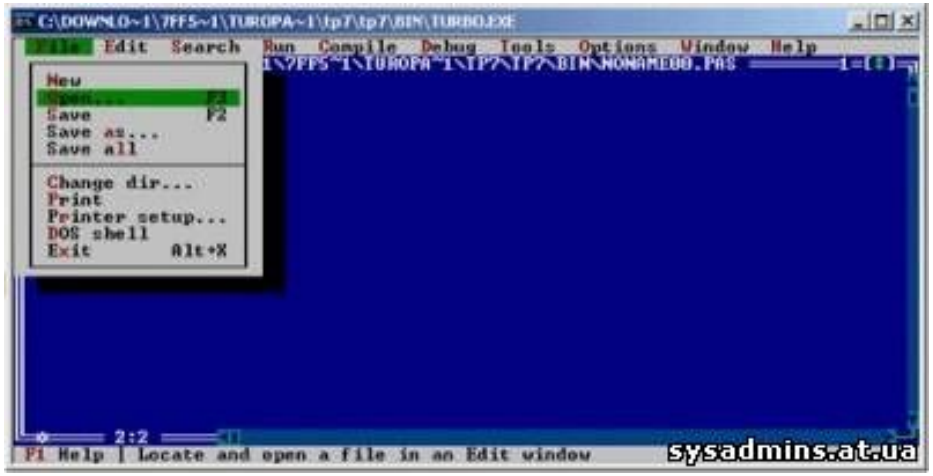
Фрагмент програми на Асемблері



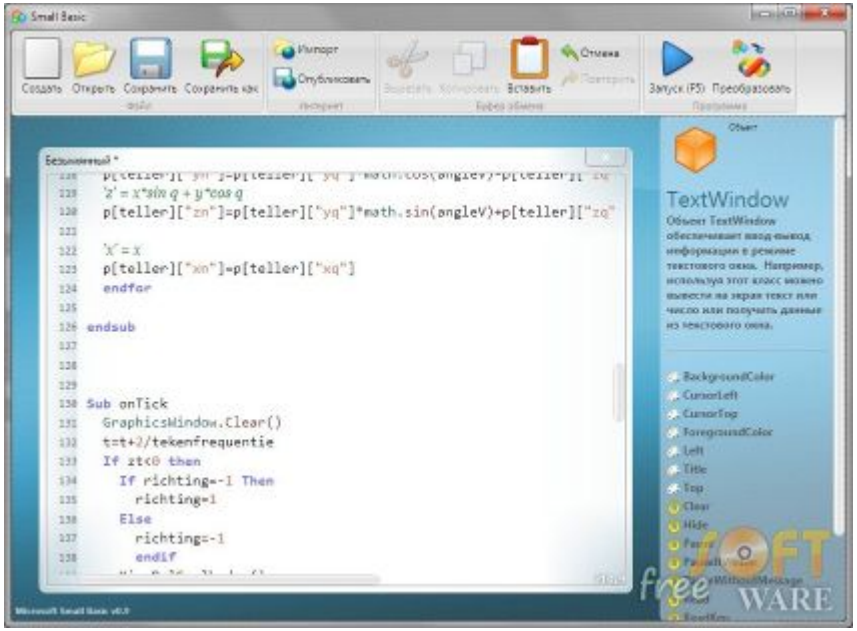
Середовище програмування Delphi



Borland Turbo Pascal 7.0



Small Basic — програмування для початківців



П'ять поколінь мов програмування

Перше покоління

Початок 1950-х років — мови перших комп'ютерів. Перша мова асемблера, створена за принципом «одна інструкція — один рядок».

Основна відмінна риса: орієнтування на конкретний комп’ютер.

Друге покоління

Кінець 1950-х — початок 1960-х р.р. Розроблено символьний асемблер, в якому з'явилося поняття змінної. Це перша повноцінна мова програмування.

Основна відмінна риса: орієнтування на абстрактний комп'ютер

П'ять поколінь мов програмування

Третє покоління

1960-ті р.р. — мови програмування високого рівня. Їхні характеристики:

відносна простота;

незалежність від конкретного комп'ютера;

можливість використання потужних синтаксичних конструкцій.

Простота мов дає змогу писати невеликі програми і людям, які не є професійними програмістами.

Основна відмінна риса мови третього покоління: орієнтування на алгоритм (алгоритмічні мови).

Приклади: Сі, Паскаль, Джава, Бейсік, та багато інших.

Всього у світі існує близько 200 популярних мов програмування третього рівня.

Четверте покоління

Початок 1970-х р.р. до сьогоднішнього часу. Створюються мови, призначені для реалізації великих проектів. Проблемно-орієнтовані мови, що оперують конкретними поняттями вузької галузі. Як правило, в такі мови вбудовують потужні оператори, що дозволяють одним рядком описувати функції, для опису яких мовами молодших поколінь потрібно було б сотні чи навіть тисячі рядків початкового коду.

Часто відносять: SQL, SGML (HTML, XML), Prolog, та багато інших вузькоспеціалізованих мов. Щоправда ряд мов, які відносять до четвертого покоління, не є мовами програмування як такими. Наприклад SQL є мовою запитів до баз даних, HTML є мовою розмітки гіпертексту, а не повноцінними мовами програмування, скоріше вони виступають своєрідними спеціалізованими доповненнями до мов програмування. Теж саме стосується XML.

Основна відмінна риса мови четвертого

П'ять поколінь мов програмування

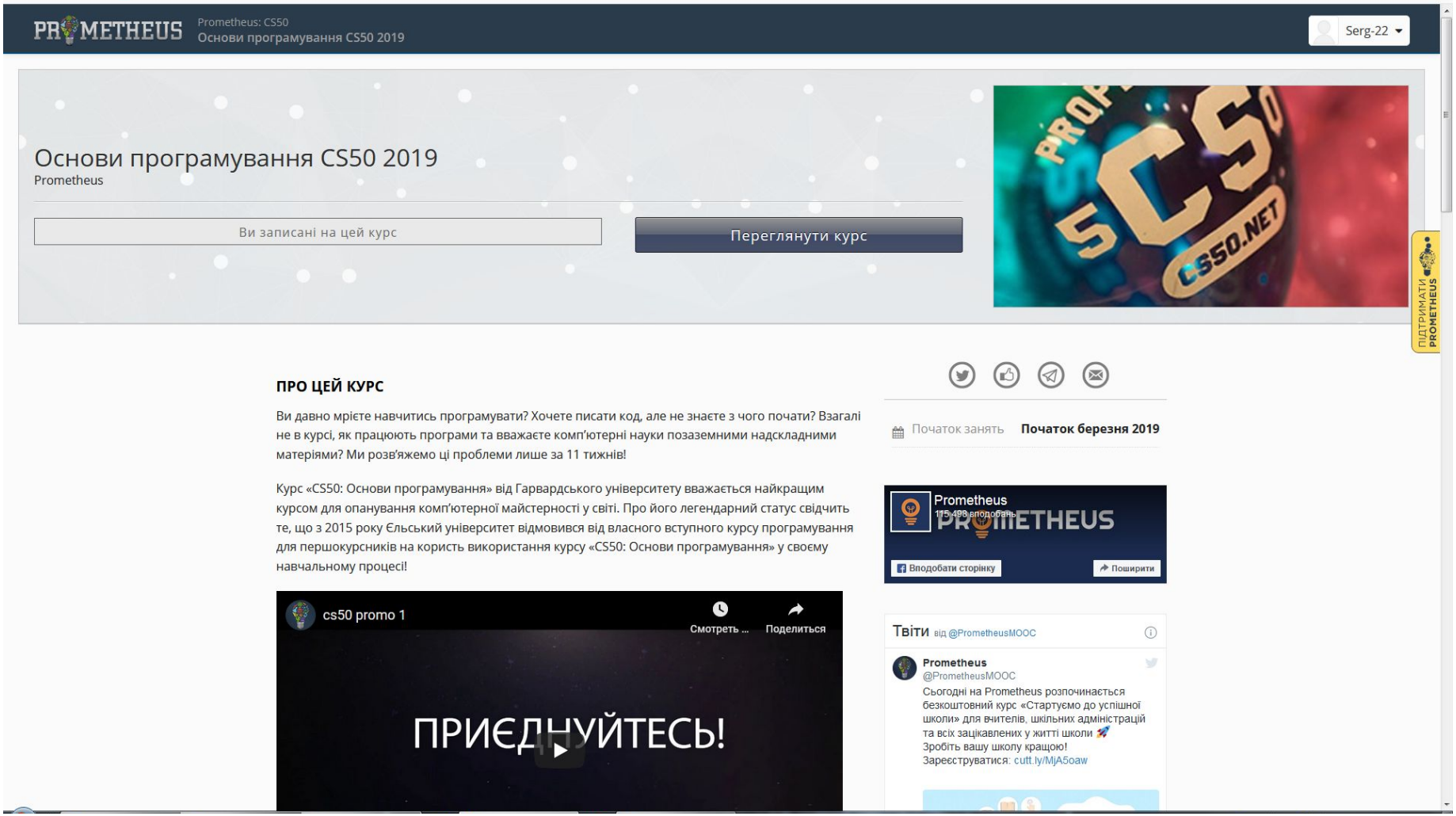
П'яте покоління

П'ятого покоління мов програмування поки що **не існує**.

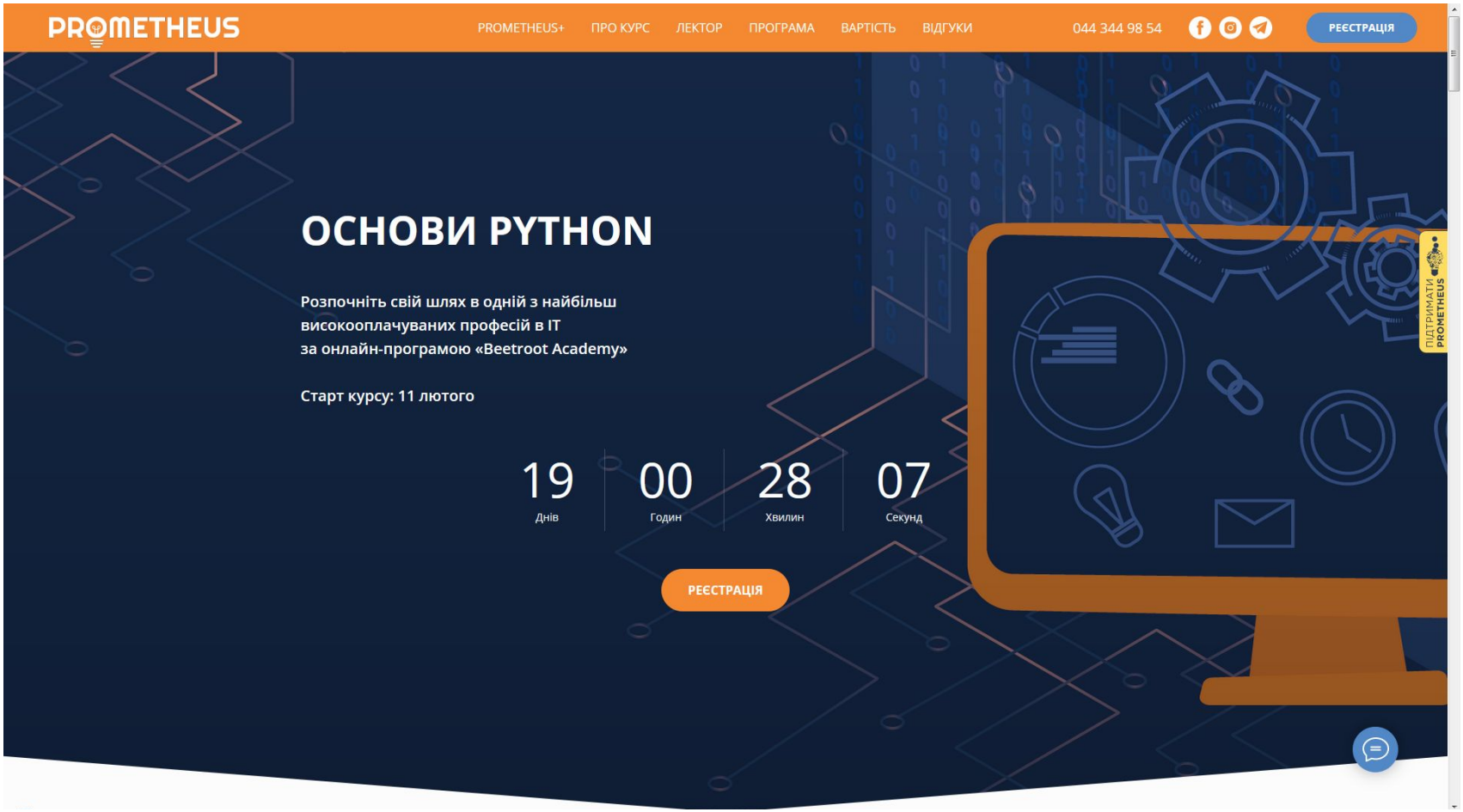
Виробники пропроієтарних програмних продуктів часто намагаються приписати своїм продуктам якісь маркетингові особливості, і деколи вказують що їхній продукт — це «мова п'ятого покоління». Насправді, всі ці продукти — це просто середовища для прискореного створення продуктів (Rapid Application Development — RAD), і використовують мови третього та четвертого поколінь.

Мова п'ятого покоління витіснить чи суттєво потіснить мови третього (напр. Java) і четвертого покоління (напр. SQL) за рахунок значно збільшеної продуктивності праці програміста — в 10-1000 раз. За прогнозами, **5GL** буде оперувати мета-мета-даними.

Наразі існує єдина мова, яка працює з мета-мета-даними, — це мова команд менеджерів пакетів чи менеджерів залежностей, таких як **apt, yum, smart, maven, cpan** та інші. Вони оперують над метаданими про метадані про дані у пакетах. Використання **apt-get, yum та smart** дійсно надзвичайно підвищило продуктивність системних адміністраторів — приблизно в 1000-у раз. Використання менеджерів залежностей, таких як **maven, cpan, rakudo, pim, easy_install** дійсно значно підвищило продуктивність програмістів



https://courses.prometheus.org.ua/courses/course-v1:Prometheus+CS50+2019_T1/about



https://prometheus.org.ua/prometheus-plus/python-beetroot-course/