Programming languages

Structured programming, Imperative programming, Procedural programming

- Separated code & data scopes
- Global functions
- Assignment operator
- Execution: set of commands are changing global program state

Functional programming

- Lambda calculus
- Everything is a function
- Lists, trees
- No explicit assignment operator (explicit variables)
- No explicit program state

```
(defun factorial (n)
        (if (= n 0)
                1
                (* n (factorial (- n 1))) ) )

(loop for i from 0 to 16
        do (format t "~D! = ~D~%" i (factorial i)) )
```

Logic programming

- Facts & rules
- Theorems
- Mathematical Logic

```
man(Vlad).
woman(Tanya).
family(X, Y):-man(x),woman(Y).

?- family(Vlad, Y)
Y = Tanya
```

Object-oriented programming

- Encapsulation
- Polymorphism (overloading, virtual methods, static & dynamic type)
- Inheritance (multiple inheritance, interfaces, abstract classes)
- Aggregation & Composition
- Generic programming/Meta-programming
- Meta-classes

```
template <int N> struct Factorial {
        enum { value = N * Factorial<N - 1>::value };
 };
template <> struct Factorial<0> {
        enum { value = 1 };
};
// Factorial<4>::value == 24
// Factorial<0>::value == 1
void foo() {
        int x = Factorial<4>::value; // == 24
        int y = Factorial<0>::value; // == 1
}
```

Context-free grammar

- Set of production rules that describe all possible strings in a given formal language

Digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Digits → Digit | Digit Digits
Number → Digits | (-Digits)
Exp → Number
Exp → Exp + Exp
Exp → Exp – Exp
Exp → Exp * Exp
Exp → Exp / Exp
Exp → (Exp)

Programming languages/software development evolution

- Key paradigms will stay
- Distributed environments
- Functional trend
- Dynamic resource management
- Open source
- Libraries, Frameworks, Tools, IDEs
- Hybrid clouds
- ML, Big Data, Blockchain