Orthogonality – A relatively small # of constructs that can be combined into a small # of ways to build the control and data structures of a language Purity – A product that won’t let you check array bounds in C#. Assembly – Directly controls physical hardware, no compiler, one machine instruction High Level Language – More abstract, can be compiled/translated into machine language. Von Neumann: Serial Component. Multicore Processor: multiple execution time. Symettrical Multiprocessing: System with multiple identical processor that share memory and connect via a bus (Serial or parallel computation) Evaluation Criteria (3 Major Ones) -Variables, Assignment Statement, Iteration Language Categories: Imperative – Contains a component (statement) Functional – Updated imperative (no iteration, rely completely on recursive) Logical – Rule based, rules specified in no particular order, Data = fact or rule / Query. Control: interfering engine () – then use Object-Oriented – Closely related to imperative. 2nd type – procedural. Language Design Trade off: 1) readability vs cost of execution – index range check in ada & c. 2) Writability. Implementation: Compilation, one time translation, multiple use of the executable, suitable to any time sensitive application (DIAGRAM BELOW) lexical analyzer – sequence 🡪 lexical units, syntax analyzer lexical units 🡪 parse tree, Intermediate code 🡪 parse tree, code generator 🡪Intermediate code 🡪 Machine code. Interpretation: Slow execution time, suitable to small tasks. Hybrid: .NET languages: a “Common language runtime which runs bytecodes in an internal language format. Combination of the first two parts of the compiler. Trans compiler: a source to source compiler. EX. C# 🡪C.Syntax – Form or structure of the expression statement and programming units Semantics – Meaning of expression, statements, and programming units. The meaning of language is described by state changes of computer. SMALL DIAGRAM: {state of computer} statement I {stateof computer} – goes down to: different of states 🡪 meaning of the statement. Approaches: pure hardware – too expensive. Pure software – machine dependent. Virtual machine – machine independence JVM. Terminology: - Language: set of sentences. Sentence: string of characters over some alphabet. Lexemes – Lower level syntax unit language (smallest unit, cannot break down any further) (+,-,I,count). Tokens: Category name of lexemes (e.g Identifier). BNF – Abstractions are used to represent syntactic structures (act like syntactic variables (also called nonterminal symbols) 1) This rule contains 3 abstractions and one lexemes 2) A rule has a LHS & RHS, consist of terminals (lexemes and tokens) and nonterminal systems 3) A grammar is a finite nonempty set of rules BNF is the original, most simple, mostly used in academic papers of theoretical context, for communicating to humans. (as opposed to being used in compiler/parser.) There's no one exact specification of BNF. EBNF means Extended BNF. There's not one single EBNF, but each author or program define their own variant that's slightly different. Derivation - a process from start symbol and stops when we reach a terminal (Finish).

ambiguous grammar is a [context-free grammar](https://en.wikipedia.org/wiki/Context-free_grammar) for which there exists a [string](https://en.wikipedia.org/wiki/String_(computer_science)) that can have more than one [leftmost derivation](https://en.wikipedia.org/wiki/Leftmost_derivation) or [parse tree](https://en.wikipedia.org/wiki/Parse_tree), while an unambiguous grammar is a context-free grammar for which every valid string has a unique leftmost derivation or parse tree Operator precedence determines which operator is performed first in an expression with more than one operators with different precedence. For example 10 + 20 \* 30 is calculated as 10 + (20 \* 30) and not as (10 + 20) \* 30. Associativity is used when two operators of same precedence appear in an expression. Associativity can be either **L**eft **t**o **R**ight or **R**ight **t**o **L**eft. For example ‘\*’ and ‘/’ have same precedence and their associativity is **L**eft **t**o **R**ight, so the expression “100 / 10 \* 10” is treated as “(100 / 10) \* 10”.

Precedence and Associativity are two characteristics of operators that determine the evaluation order of subexpressions in absence of brackets.

CF-Grammar: 1) Set of Nonterminal symbols, N 2) Set of Terminal symbols, T 3) Set of Production rules, P. A🡪B A~~E~~(N U T)\* Repeated zero. B(N U T)\*

Terminal Symbols: The elementary set of the language, literal strings that CANNOT be changed in a production line. Nonterminal Symbols: The symbols that CAN be replaced in a production rule. Production Rule: A grammar is defined by its production rule CF Grammar: Every production is of the form where A~~E~~N, &~~E~~(NUT)\* Regular: Every production is one of the form A 🡪 B or A🡪aB or A 🡪 Lambda Context Sensitive: If every production is of the form S🡪aSBC, S🡪aBC, CB🡪HB,HP🡪HC, HC🡪BC, aB🡪ab, bB🡪bb, bC🡪bc, cC🡪cc. Rule for Recursion: you should always not use any left recursive rule to avoid an infinite loop. CODE== factor() { switch() {case condition1: return (expr()) case condition: return id() }}. Operational Semantics: The meaning of a program by executing its statements on a machine, either simulated or actual Axiomatic Semantics: Based on logical conditions. An assertion before a statement(precondition), the weakest precondition is the least restrictive (first question of HW) Denotational Semantics: Based on functions. Complete, but complex. Flex: an acronym that stands for "fast lexical analyzer generator. Lexical analysis is the processing of character sequences such as source code to produce symbol sequences called tokens for use as input to other programs such as parsers. Lexene – Regular expression. Input: A file Output: yylex()

Bison**:** is a program that converts the formal description of a **computer** language grammar into a C language program that can parse the syntax and symbols of that grammar into instructions that the **computer** can execute. Syntax analyzer Typedef: keyword used in C language to assign alternative names to existing datatypes. Its mostly used with user defined datatypes, when names of the datatypes become slightly complicated to use in programs. Variables: Linear DIAGRAM going DOWN: variable 🡪 expression 🡪 statement 🡪 program 🡪 software… Static Variables: **Static** Keyword in C++ **Static variables** in a **Function**: When a **variable** is declared as **static**, space for it gets allocated for the lifetime of the program. ... So, its value is carried through the **function** calls. The **variable** count is not getting initialized for every time the **function** is called. Static Function: only visible to other functions in the same file: SMALL CODE 🡪 static void func1(void) {} Void Func2(void){} Scoping – represents a range in which a variable is visible 1) Static – Physical Layout of program 2) Dynamic – dependent on Calling Sequence Reference Environment – All visible variables at a certain location in your code.

Python: kind of GME language. Advantages – readability much better than perl, developer productivity: 1/3 to 1/5 size of c# or Java code. Disadvantages: slower than c/c#. Run c:\python script.py. List \*associative array: A=[‘spam’,’egg’,100,4+3g]; A[0:2] 🡪 [‘spam’,’egg’]; A[:]=[]; -assignment statement #Fibonacci series A,b=0,1A=0, b=1. #RHS is evaluated from left to right. #RHS is evaluated before any. RHS of the assignment takes place. Pop: Removes he item at the given position in the list. If no index is provided, a pop() removes and returns the last emement added. Package: a self-contained unit of user-defined variables and subroutines. It prevents naming conflicts with no other packages. Constructor: you can use “new” as a function. It returns an object reference by blessing function, which blesses a reference to the package’s class. Inheritance: Perl has no special syntax for specifying the class to inherit from instead, its all strickly in the semantics. Each package can have a variable called @ISA to govern inheritance. Person 🡪Employee

 

 











