Date.....

* Language

It can be defined as means of communication.
Language is of 2 types:
(i) Natural language
(ii) Programming language — (a) Algorithm — definiteness, Finiteness, I/P, O/P, Efficiencies
(b) Data structure

Complexity — Time → No. of steps in a algorithm
Space → No of variables used in a program the memory used by them.

* Why do we study programming languages?

Here are some reasons to study the programming languages

1. To improve your ability to develop the effective algorithm.
   - → OOP
   - → Structured programming
   - → Concurrent programming

   } understanding the language that implement these concepts.

2. To improve the use of your existing programming languages.
   - By understanding how features in language are implemented. For ex. how data such as array, string, list are created and manipulated by your programming language

Spiral

Date.....

3. To allow a better choice of programming languages
→ Knowledge of variety of programming languages may allow the choice of just the right languages for a particular project.

Eg. (i) AI Application
These would be in LISP, ML or Prolog.
(ii) Internet Applications :
using Java, Perl, C, C++ etc.

4. To make it easy to learn a new language.
→ A knowledge of PL constructs and implement the techniques allow the programmer to learn a new PL more easily when the need arises.

5. To make easy to design a new language.
→ most of the application are really in the form of PL and a designer of user interface for a large program such as O.S, graphic packages and text editor.

6. To increase your vocabulary of useful programming construct.
→ Human use the language to express the thought, ideas or concepts but language also serve the structure that allow the programming language to understand the implementation of the problems.

Spiral

Date.....

Programming Domains

1. Scientific Applications
- Computer invented in 40's.
- FORTRAN (Formula Translation)
- ALGOL (Algorithmic language)

2. Bussiness Applications
- produces the reports etc.
→ COBOL

3. AI applications
- Speech recognition, image processing & robotics invention
→ LISP, PROLOG, PERL etc.

4. System programming Applications
- OS & programming tools
→ C, C++, Java etc.

5. Scripting language Applications
- Web design & implementation
→ PHP, HTML, JSP, ASP etc.

* Role of PL

Spiral

Date.....

- The prominent purpose of PL is to provide instructions to a computers.
→ Many languages have been designed that can be alter to meet the new needs and combine with other languages.
→ The development of PL has been to add the more ability to solve the problem using a higher level of abstraction.

(i) Computer capibilities

The PL used to write the computer program which instruct the computer to perform some kind of computation, and organise the flow of control blw external devices.

(ii) Programming methods

Every PL must used their own methods that can be used as a feature or characteristics to solve a particular problem.

✱ Role of Programming language

1. Computer capabilities
2. Programming methods
3. Applications
4. Implementation methods
5. Theoratical & standardiration

Spiral

Date.....

(iii) Applications

The requirement of these new applications affect the design of new languages and the revision and the extension of older one.

(iv) Implementation methods

The development of better implementation method has affected the choice of features to improve include in the new language design.

(v) Theoratical & Standardiration

The research into the conceptual formation for language design and implementation using the formal mathematical methods as needed for understanding the strength and weakness of the language features.

The need for standard language that can be implemented easily on the variety of computer system which allow the programs that to be transported from one computer to another and as provided are strong conservative influence of the evolution of language design.

✱ What makes a Good language?

OR

Attributes & characteristics of a good language?

Spiral

Date

1. Simplicity, Clarity and Unity

- A programming language should be very clear and simple so that a user can understand the language easily. The syntax of the language should be close to natural language.

2. Orthogonality

It refer to the attribute of being able to combine various features of a language in all possible combinations with every combination being meaningful.
- For ex- a suppose a language provide an expression that can produce a value and to also provide conditional statement that evaluate an expression to get a True or False. These two features of the language expression and conditional statement are orthogonal. If any expression can be used with in the conditional statement

3. Naturalness

A language need a syntax that allow the program structure to reflect the logical structure of a program. It should be possible to translate such a program designed directly into the appropriate program statement that reflect the structure of the algorithm. It basically work on abstraction means hiding the complexity of the program.

Spiral

Date

4. Support for abstraction

Abstraction hides the complexity and keep only essential information. Detailed working of any entity is not required.

5. Program verification

The program written in a language is always a centered concern. There are so many techniques for verifying that a program correctly its required function. It may be verify by executing with input data and checking the output result against the specifications.

A language which make a program verification more complicated are not good.

6. Programming Environment

The technical structure of a programming language, is only one aspect that affecting its integrity. The special editor and testing packages are required to the language that may speed the creation and testing of the program.

7. Portability of program

Portability means architecture independence. A language should be portable if we run one program to first machine and

Spiral

Date.....

same program can be run on different kind of machine without changing the program the language is called portable.

8. Cost of use

Cost is a major issue to be take care in program design for large production program that will execute repeatedly.

→ Cost of program execution → It is imp.

→ Cost of program creation and testing → Program should be easy to create, easy to test and easy to use.

9. Robustness and Reliable

A programming language should have good exception handling techniques. Exception means run time error.

A program should give same result for same input under different circumstances then that programming language is reliable

X Evolution of programming languages (Towards Higher Level language)

→ HLL (High level language)
- AL (Assembly language
- LLL (Low level language)

Spiral

Date.....

1. High level language (HLL)

A HLL is a computer programming language that support the system development at a high level of abstraction. Different languages are designed to meet different programming needs that is scientific application by FORTRAN (Formula Translation), bussiness application by using (COBOL) [Common Bussiness oriented language), Artificial intelligence by using LISP processing, prolog and system programming as C, C++ and web programming as PHP, java script etc.

2. Assembly language

In this language the coding is done in the form of alphanumeric symbols known as mnemonics.

3. Low level language

In this language coding is done in the form of binary sequence 0 and 1 that is understand by a machine i.e. computer.

Spiral

Date.....

* Programming Paradigm

It means the style of programming for different languages.

(i) Imperative language or Procedural language

A program consist of a sequence of statements and the execution of each statement that cause the computer to change the value of one or more location in its memory. The syntax of such languages generally has the form like Statement 1;
Statement 2;
= = = ;

Eg- C, C++, Pascal, FORTRAN.

(ii) Declarative language or Functional or Application

In this language the execution of program functions wise. A programming language is to look at the function that the program represent rather than just the state changes as the program executes. and also control the flow of the program.
For ex- LISP, Ruby, Python, SQL.

(iii) Rule based language

It is similar to imperative language except that statements are not in sequences.

Spiral

Date.....

It is basically based on conditional evaluation.
For eg: If cond^{n}1 ; expression 1;
If cond^{n}2 ; expression 2;
ALGOL, PROLOG

(iv) Object - oriented programming

It is based on object and classes by building the concrete data object. It gains the efficiency build the flexibility of the application models

Ex, C++, Java, C#

(v) Structured programming or Modular programming

* Programming Environment

It is in which programs are created and tested. It's consist a set of tools that used in software development such as editor, debugger, verifier, test data generators and O.S.
We also used a command language for invoking them. It consist the following -

(i) Effect on language design

The program testing and debugging has the phase of separate compilation. and the compiler may need information about the sub-programs or share data objects.

Spiral

Date.....

Testing and debugging contain the features such as break points and execution trace features.

(ii) Environment Framework

A support environment consist of infrastructure services hold the environment framework that manage the development of a program. The framework supply the services such as data repository, GUI, security and communication services.

Programs are written to use these services and the programmer need to use the infrastructure services as part of their program.

(iii) Job control and process language

To control the work or task to the environment we have to follow the procedure. So that if you want to execute a program such as the C compiler you move the mouse to the appropriate picture or icon on the screen and click the mouse. Appropriate program start to execute. Before the window base system you would type in the the name of the program to run and compute the data as required, and user could invoke a model and execute the program.

Spiral

Date.....

The General category of process for scripting language that is interpreted and view the program and files as the primitive data to manipulate and control the entire programs which runs on the programming environment.

* Language description

-) Compiler / Interpreter
-) Assemblers
-) Linker / Loader

(A) –
(B) –

(A) General syntactic Criteria (Syntactic structure)

The primary purpose of syntax is to provide mean for communication blw programmer and programming language processor. We consider some of the ways that language syntactic structure may be designed to satisfy the conflicting goals.

(i) Readability

A program is readable if the structure of algorithm and data represented by the program. It is enhanced by a program syntax in which the differences reflect the structure of the original program.

Spiral

(ii) Writability

The syntactic feature that make a program easy to write and enhanced by the use of regular syntactic structure.

A syntax is redudant if it communicate the same item of information to more than one way.

(iii) Ease of verifiability

A programming language statements are relatively easy but the overall process of creating the correct program that is extremely difficult. so we need some techniques that enable the program to be mathematically proved.

(iv) Ease of translation

Making the program easy to translate into an suitable form. It relate to the need of translator that process the written program.

(v) Lack of Ambiguity

Ambiguity is a central problem in every language design. A language definition ideally provide a unique meaning for every syntactic construct that a programmer may write. An ambiguous construction allow two or more different interpretation. This problem usually arise not in the structure of an individual program.

Spiral

Date

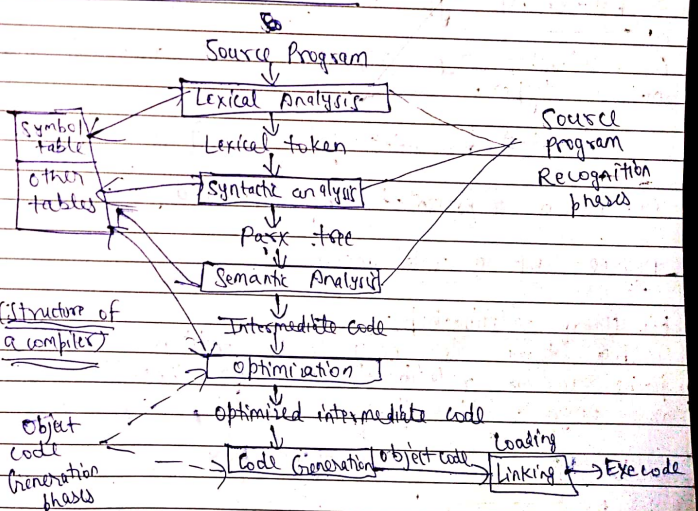
element but in blw different structure.

(B) Syntactic Elements of a language

The general syntactic style of a language is said by the choice of various basic syntactic elements. We consider the following

(i) Character set
(ii) Identifier
(iii) Operator symbols
(iv) keywords
(v) Noise words — goto label
(vi) Expressions
(vii) Statements
(viii) Comment.

X Stages in Translation

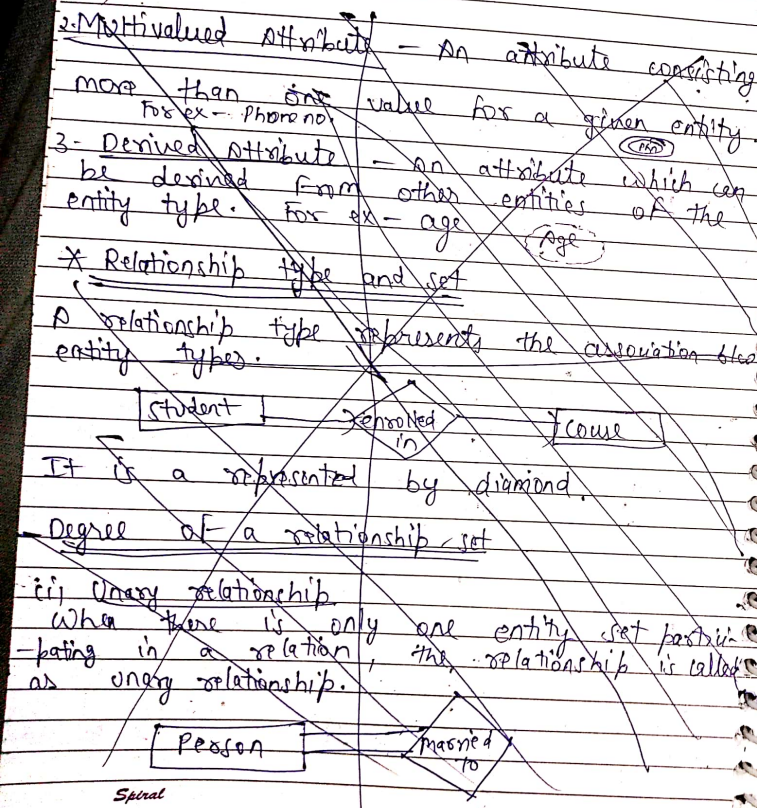


Spiral

Source code → Translation → Object code

2. Multivalued Attribute – An attribute consisting more than one value for a given entity. For ex – Phone no.

3- Derived Attribute – An attribute which can be derived from other entities of the entity type. For ex – age



* Relationship type and set

A relationship type represents the association between entity types.

It is a represented by diamond.

Degree of a relationship set

(i) Unary relationship

When there is only one entity set participating in a relation, this relationship is called as unary relationship.

Spiral

Date

* Stages in translation

The process of converting the source code into object code is known as Translation.

Logical translation process is divided into two major parts as analysis of source code and synthesis of executable code.

(i) Analysis of source code

In this phase the translator analyse the source code and consist the following three subphases

(a) Lexical Analysis or Scanning

In this the sequence of characters are group into its elementary like identifier, symbols, keywords, blank, comment, noise words. Basically Identifier are used to identify a particular variable that store in symbol table. The former model is used to design a lexical analyser that is FSA (Finite state Automata).

(b) Syntactic Analysis or Parsing

In this phase the syntax of the statement is being analysed. tokens are passed and examing for correct syntactic structure that is based on rules used for the language. The programme syntax expressions are detected and token organised into a statement and generate a

Spiral

a parse tree and table and produce the error message for invalid strings. Technique used for syntactic analysis are based on formal grammar.

(c) Semantic Analysis

This phase is intermediate b/w analysis and synthesis part of transmission. The various functions in this phase are like symbol table maintenance, error detection and expression of macros.

The optimization can be done and unnecessary statements are eliminated, statements move out of loops if possible and recursion can be removed as required.

Produce the error message for invalid constraint. For ex - identifier are not declared and the information must be stored together with identifier.

(ii) Synthesis of object code

In this output produced by semantic analyser has converted into object code. It consist of the following subphases and the final stage of translation are concerned with construction of the executable program from the output produced by the semantic analyser.

(a) Optimization

The semantic analyses produce the output in some intermediate code and the internal

Spiral

representation of such code is not properly formatted. The code generator produce the properly formatted code with the help of code optimization. The compiler can optimise code to obtain the result as efficient as assembly code.

For ex - save the intermediate result in registers. remove the constant operation from loops. and the use of register is optimised and the calculation of assay accessing formulas to be done on the basis of internal structure of the compiles.

(b) Code generation

After the translated program in internal representation has been optimized and it must be form into assembly language statements, machine code or other object program i.e. to be output of the translation.

This process involve formatting the output properly from the information contain in the internal program representation.

The output code may be directly executable or there may be other translation steps to follow and code is also linked with libraries if necessary.

(c) Linking and Loading

In the optional final stage of translation, the pieces of code resulting from separated translation

Spiral

Date.....

of subprogram are linked together with the help of linker to produce the final object code. The linking loader or link editor load the various segment of translated code into main memory or to be run by CPU.

* Boot strapping (Self compiling compiler)

The translator for new language is written in that language or we can say the compiling compiler.

For ex - the initial pascal compiler was written in pascal and designed to execute on a P-code virtual machine.

One problem with this is how to get started if the original pascal compiler was written in Pascal and how did this first compiler get compiled. One way is to compile the compiler by hand into P-code. The Pcode interpreter can execute this hand translated compiler that is not difficult for some of the compiler.

* Types of compiler

1. Single pass compiler

It is a compiler that passes through the source code of each compilation you need only once. A pass can have more than one phase. Passes are the no. of times the compiler traverse through the source code. whereas the multipass compiler processes the source code or

Spiral

Date.....

abstract syntax tree of a program several times.

* Formal Translation Models

A grammer consist of a set of rules i.e. Production rules that specify the sequence of characters. A formal grammer is just a grammer specified using a strictly defined notation. There are two classes of grammer useful in compiler construction -

1. BNF Grammer (Backus - Naur form)
2. ~~Regular Grammer~~ FSM (Finite state M/C)
3. PDA (Push down automata)

1. BNF Grammer

A BNF grammer is composed of finite set of BNF grammer rules which define a language such as programming language. We consider the syntactically a set of correct programs each of which is simply a sequence of characters. It is formal representation of CFGs (context Free grammer).

<digits> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<unsigned integer> ::= <digit> | <unsigned integer> <digit>

Terminals : E, T, F etc.
Non-terminals : +, *, -, /, id etc.

Spiral

LL → (L to R using Left most) :: - assignment

Parsing - grammer - string (I/P)

Date

Eg: Grammer for simple Assignment statement

<assignment statement> :: = <variable> = <arithmatic expression>

<arithmatic expression> :: = <terms> | <arithmatic expression> + <term> |

<term> :: = <primary> | <term> * <Primary> |

<primary> :: = <variable> | <Number> |

<variable> :: = <identifier> | <identifier> [<subscript list>]

<subscript list> :: = <arithmatic expression> | <subscript list>

Parse Tree

It represents the syntactic structure of a string according to some CFG.

Passing methods

1. Top - down passing

- Build parse tree from root to leaves using LMD (Left most derivation)
- Recursive Descendant, LL Parser.

2. Bottom - up passing

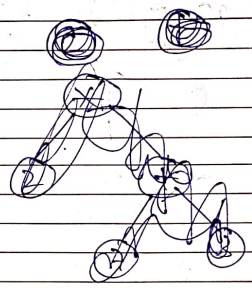
- From leaves to root
- operator precedance - , LR parser,
- LALR (Look Ahead LR)

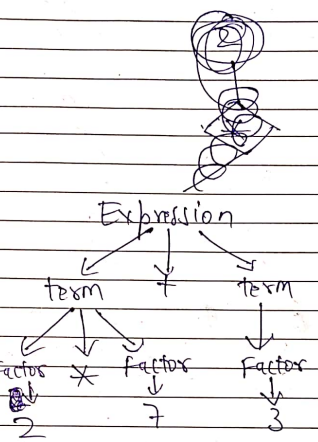
Spiral

If there is more than one parse tree then it is ambiguity.

Date

2 * 7 + 3





* Ambiguity

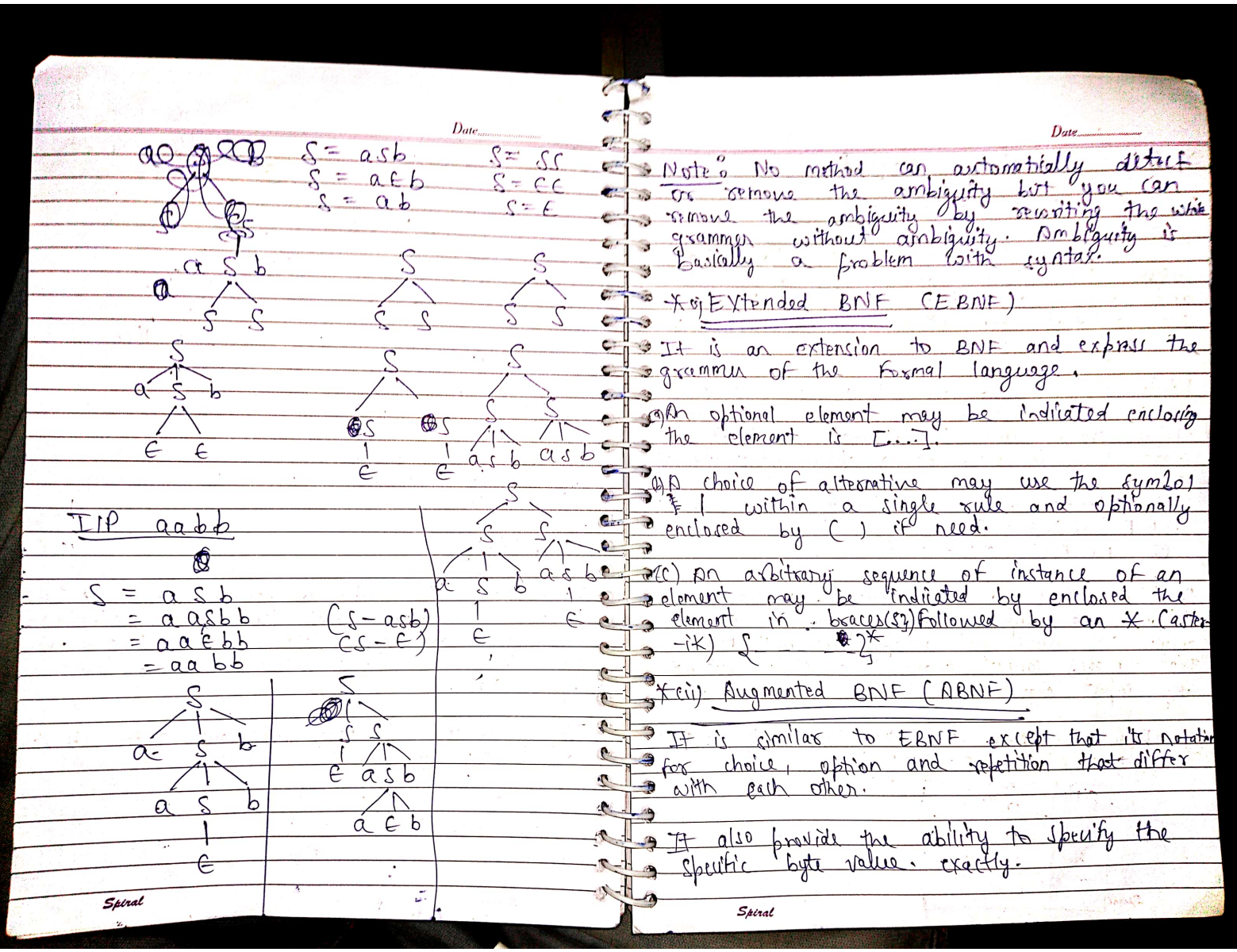
A grammer is said to be ambiguous if there exist more than one left most derivative or more than one right most derivative or more than one parse tree for the given input string.

If a grammer is not ambiguous then it is called unambiguous.

S = asb | ss

S = ∈

Spiral



Date.....

* Formal Translation Model

2. FSM (Finite state Machine)

FSM is used to recognise the pattern that takes the stream of symbols as input and change its state accordingly. In transition the automata can either move to next state or stay in the same state.

Finite Automata has two states such as acceptance or rejection. When input string successfully crossed and the automata reached its final state then it will be accepted otherwise rejected.

Finite Automata consist of following $(Q, \Sigma, q_0, F, \delta)$

DFA $\delta : Q \times \Sigma \to Q$

NDFA $\to$ $Q \times \Sigma \to 2^Q$

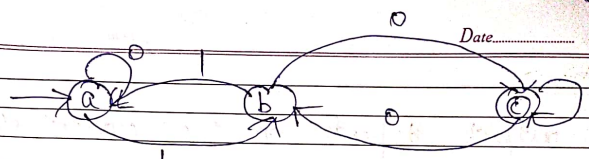
* DFA refers to the uniqueness of the computation and input character goes to one state only. It does not accept the null move that means cannot change the state without any input characters

Ex:- Draw the DFA for $Q = \{a, b, c\}$, $\Sigma = \{0, 1\}$, $q_0 = \{a\}$, $F = \{c\}$

State	Input 0	Input 1
a	a	b
b	c	a
c	b	c

Spiral

Date.....



NDFA

It can move to any combination of the states in the machine. It consist the multiple next state for each input symbol as DFA consist single next state for each input symbol.

DFA	NDFA
– single state	– multiple space
– more space	– less space
– Backtracking is always allowed.	– Backtracking is not allowed

* Tranducers

An automata that produce output based on current input or previous state is called Tranducer. It has two types.

1. Mealey machine
2. Moore machine

1. Mealy

Output depends both on the current state and current input

Spiral

Scanned by CamScanner

Date.....

2- Moore

The output depends on the current state.

Conversion do it yourself

* Regular Grammer

A right or left linear grammer is called a regular grammer. Every regular expression can be represented by Regular grammer.

- We can generate finite automata for regular grammer.

$<NT> = <T><NT> | <T>$

Regular expression is used to generate pattern of strings.

3 * PDA

It is an abstract model machine similar to finite state automata. It can remember infinite amount of info. that is augmented with an external stack memory that is used to provide last in first out (LIFO) memory management capability to PDA.

A PDA can push an element into the top of the stack and pop up an element from the top of the stack. A PDA is more powerful than finite automata. It is defined

Spiral

Date.....

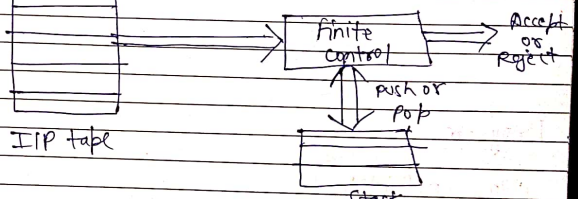
by the following tuples -

$PDA = \{Q, \Sigma, \Gamma, q_0, Z, F, \delta\}$

Stack symbol is in Γ

Stack symbol left to right

mapping function



Q Consider the PDA is given by M: ($Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $\Gamma = \{\delta, b\}$, δ, q_0, $z_0 = 0$, $F = \{q_3\}$ where $\delta(q_0, a, 0) = \{(q_1, 10), (q_3, \epsilon)\}$ $\delta(q_0, \epsilon, 0) = \{(q_3, \epsilon)\}$

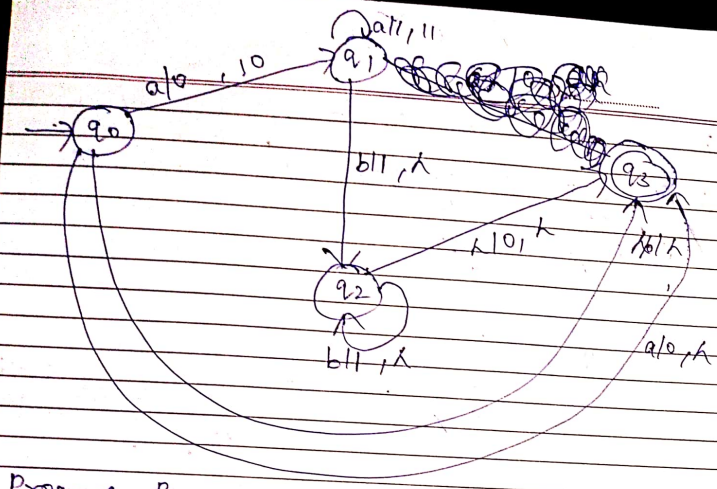
$\delta(q_1, a, 1) = \{(q_1, 11)\}$

$\delta(q_1, b, 1) = \{(q_2, \epsilon)\}$

~~$\delta(q_1, b, 1) = \{(q_2, \epsilon)\}$~~

$\delta(q_2, \epsilon, 0) = \{(q_3, \epsilon)\}$

Spiral



Program P_2

```
Var n :int;
procedure D (var n: int)
begin
    n=n+1;
    print n;
end
procedure D
begin
    Var n : int;
    n=3;
    w(n)
    end
begin // begin P2
  n=10;
  D;
  end
```

Spiral

Date.....

If language has dynamic scoping and parameters are passed by reference what will be printed by the program

The result of the above program is 4.

* swap program with the help of bitwise operator

```
@ Swap ( int *n , int *y )
{
    *n = *n ^ *y ;   // ^ is XOR
    *y = *n ^ *y ;
    *n = *n ^ *y ;
}
```

Bitwise operator

swap(3, 4)

3 → 011

4 → 100

011 ^ 100 = 111 = *n

111 ^ 100 = 011 = *y

111 ^ 011 = 100 = *n

```
Void main()
{ int num1, num2;
  printf("Enter two nos");
  scanf("%d%d", &num1, &num2)
printf("\n The no. before swapping are
        number = %d number 2 = %d", num1, num2);
printf("\n the no. after swapping are no1 =
        %d number 2 = %d", num1, num2);
}
```

Spiral

Unit-2

Date.....

1. Name

Name is basically a string of character used to identify some entity in a program.

2. Object

Object is a variable having the memory location.

3. Data

Data is a particular value assigned to object of particular type.

4. Data object

It act as a containes which can store any data value. Data value may be a no, string, boolean etc.

5. Variables

Variables are data object which are provided with the name.

6. Datatypes

It is a class of data object together with a set of operations for creating and manipulating them. Datatypes are always specified by following three things.

Spiral

Date.....

(a) Attribute : Attribute are particular datatype for any data object.

(b) Values : Attributes of every data type associated with a particular value which can be assigned to them at runtime or compiletime. These values are not fix.
In the computer representation they are having the particular bit sequence depending upon the architecture of computer.

(c) Operation : operation are the entities with the help of which we can do the actual implementation. It may be in the form of some inbuilt function or user defined function. It is also possible through assignment and various mathematical operation.

* Implementation of data type

The following are the basic elements of the implementation of the datatype -

(a) Storage implementation

It is used to represent the data object of the datatype in the storage of the computer during program execution.

(b) Algorithm or procedure

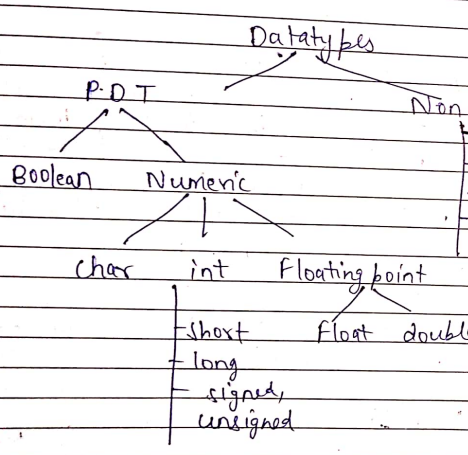
Spiral

Date

Manipulates the storage representation of data object for the defined operations.

(c) Syntactic representation

Attributes of data objects are represented by declaration or type definition value by internal & operations by using inbuilt function or user defined functions.



Declaration	Definition
Datatyp along with giving its name for memory allocation	giving value to that variable function.

Spiral

Date

eg. function declaration
- name of func
- return type

int add (int, int);

eg. int add (int a, int b) { return (a + b); }

* scope of variable

The region of the program over which a binding is maintained - It has two types:

(i) Static scoping

It is an association determined at compilation time. It use the sequential processing of the program.

(ii) Dynamic scoping

It is an association that determined at run time and program statements follow the execution order of different statement.

* Lifetime

The lifetime of a binding is the interval during which it is effective.

Spiral

Date.....

eg: int fact(int n)
{
int n;
if(n==0)
return 1;
else
{ n= fact(n-1);
return n*n;
}}

* Type checking

It means checking that each operation executed by a program receive the proper no. of arguement of the proper datatype.
To implement the statement n= a+b*c in this the compiler check the two arguement of proper datatype for each operation like +, *.
Type checking can be done at following.
Static - Information is provided at compile time.
Dynamic - The information is provided at run time.
The major advantage is greater flexibility but disadvantage is the program is difficult to debug and extra storage is required.

Spiral

Memory leak - Failure of any program

Date.....

* Binding

It is an association b/w two entities. For ex name & location, Name of function & type.
1. Static (Early) 2. Dynamic (late binding)

* Garbage collection in C++

It is a dynamic approach to automatic memory management and heap allocation that recovers and identify that memory blocks and reallocate the storage for reuse.
- The primary purpose of garbage collection is to reduce the memory leaks.
- It is opp. of manual memory management which requires the programmer to specify which object to deallocate and return to memory system.
- It is basically a memory occupied by object that are no longer in use by the programmer.

* Literals

The constant value used within a program is known as literal. These constant value occupy the memory but do not have any reference like variable.

Spiral

Date

There are four types of literals -

1. Integral literal

- Decimal : 2016U
- octal : 03740U , 020000061L
- Hexadecimal : 0X7F0U

2. Character Literal

It is a single character constant enclosed within single quotes
eg: 'a', '0', '\n', 'a' ..

3. Floating literals

1.2 , -0.4, 0.3 - -

4. String Literals

Sequence of characters enclosed within double quotes.
Eg: "A"

'A' or "A"
Character (1 byte) String (2 byte) (bcz it is followed by null (\0))

* Enumerated data types

enum colors {W, B, R, G, C}
0 4 5 10 11

Spiral

Date

It will assign value from 0 if value is not given

* Typedef

It is a keyword that is used to give a new symbolic name for the existing name in the program. This is like defining alias for the command.

```
int main()
{
typedef long long int LLI;
printf("Storage size for long long int %d \n",
                                    sizeof(LLI));
return 0;
}
```

u-unsigned

Output → 8 bytes ld = 4x2 = 8

* Pointer

It stores the memory address of another variable

Advantages

1. It reduces the complexity of the program.
2. They increase execution speed.

Types of Pointer

1. Dangling Pointer

A pointer pointing to a memory location that has been deleted is called dangling pointer

Spiral

```
int main()
{
    int *ptr = (int *)malloc(sizeof(int));
    free(ptr);
    ptr = NULL;
}
return 0;
```

2. Void Pointer

Void pointer is a specific pointer type - void* - a pointer that points to some data location in storage, which doesn't have any specific type.

```
void main()
{
    int n = 4;
    float y = 5.5;
    void *ptr;
    ptr = &n;
printf("Integer variable is = %d", *((int*)ptr));
    ptr = &y;
printf("Float variable is = %f", *((float*)ptr));
}
```

3. NULL Pointer

Null pointer is a pointer which is pointing to nothing - In case, if we don't have address to be assigned to a pointer, then we can simply use NULL.

```
int main()
{
    int *ptr = NULL;
    printf("Value of ptr is %d", ptr);
    return 0;
}
```

4. Wild pointer

A pointer which has not been initialised to anything (not even NULL) is known as wild pointer. The pointer may be initialized to a non-NULL garbage value that may not be a valid address.

```
int main()
{
    int *ptr;
    int n=10;
    ptr = &n;
    return 0;
}
```

Q. Explain the diff b/w structure and union with example

* Array types

An array is a homogeneous aggregate of data elements in which an individual element is identified by its position in the aggregate relative to the 1st element. A reference to an array element in a program includes one or more non-constant subscript and such references require

Spiral

a run time calculation to determine the memory location being as reference.

* Subscript binding and Array categories

The binding of the subscript type to an array variable is usually static but the subscript value range are sometimes dynamically bound. In some languages the 'lower bound of the subscript range is implicit'. For ex - In C, C++, Java. The lower bound of all the index range is fixed at 0, In FORTRAN I, II, IV. It was fixed at 1.

In most other languages the subscript range must be completely specified by the programmer.

The category definition are based on the binding to subscript value range and binding to storage. The category name indicate where and when storage is allocated.

(i) Static Array

It is one in which the subscript range are statically bound and storage allocation is static

(ii) Fixed stack dynamic Array

It is one in which the subscript range are statically bound but the allocation is done at declaration time during execution.

Spiral

The advantage of this array is space efficiency

(iii) Stack dynamic array

In which the subscript range are dynamically bound and storage allocation is dynamic. The advantage of the array over the static and fixed stack dynamic array is flexibility.

The size of an array need not be known until the array is about to be used.

(iv) Heap dynamic array

In which the binding of subscript range and storage allocation is dynamic and can change any no. of times during the lifetime of an array.

The advantage of this array over the other array is flexibility - Array can grow and shrink during the program execution as the need for space changes.

C and C++ also provide dynamic array call the standard library functions which are generally heap allocation and deallocation operations.

C → malloc, free
Allocation deallocation
C++ → new, delete

Spiral

Date.....

* Associative Array

It is an unordered collection of data elements that are indexed by an equal no. of value called key. In Java language a set of pair is defined in Java.util class. The design issue for associative array like the form of keys, elements and the size of associative array to be static or dynamic.

For implementation of associative array in PERL implemented by providing some fixed amount of space initially. when the structure reach some predetermine level of fullness that can also be expanded but might be costly as required at a new function to be used and all existing elements to be rehashed into the structure.

* Record type

It is possibly heterogeneous aggregate of data elements in which the individual element are identified by their name.
There is a need in program to model the collection of data - Record have been part of all of the most popular programming language.

In object oriented language the class construct support the record and C++ still improved the structure of the record although it is redudant

Spiral

Date.....

The design issue that are specific to the record has the syntactic form of reference to the field and reference environment to be allow

A report type is a datatype that describe such values and variables and computer languages allow the programmer to define the new record type. A record may have zero or more keys and a key is a field that has a record and serve as an identifier and a unique key is called primary key or record key.

* Operators

1. Arithmatic
2. Assignment - =, +=, -=
3. Relational - <=, >=, ==
4. Logical - &&, ||, !
5. Bitwise - <<, >>, ~, &, |, sizeof()

* Overload operator

It is used for more than one purpose in the programming languages. The multiple use of operator in programming language to be acceptable. The diff. operators can be used in different languages for specific purpose.

* Type conversion and Type casting

Spiral

Type conversion refer to changing an entity of one datatype into another.
Type conversions are either narrowing and widening. A narrowing conversion convert the value to a type that cannot store given approximation of all the values of the original type. While widening conversion change a value to a datatype that can allow for any possible value of the original data.

We consider the two subsection as two kinds of type conversion.

1. Implicit conversion
It is an automatic conversion done by a compiler. i.e. already defined in the functions as stored in a header file or packages.

Program to convert char to int.

```
#include<stdio.h>
void main()
{ int a; char b;
printf("Enter one no.");
scanf("%d", &a);
printf("Enter another no.");
scanf("%c", &b);
c = a + b;
printf("The sum is %d", c);
getch();
}
```

Spiral

```
int a=1;
char x='k';
sum = a+x;
```

double → int
1.000000000

2. Explicit conversion

```
double a;
a = 912.0;
int b;
b = a;
```

```
main()
{ double n = 1.2;
  int sum = (int)n + 1;
  printf("Sum = %d", sum);
}
```

* Operator Evaluation order

The operator evaluation order investigate the language rules that specify the order of the evaluation of the statement.

1. Operator precedence
It is the order of evaluation of operators in expressions. It basically determine the grouping of terms in an expression and decide how an expression is evaluated. Certain operators have higher precedence than others.

2. Associativity rules

It define the order in which the operators of the same precedence are evaluated in an expression. When an expression contain two

Spiral

adjacent occurence of operators with the same level of precedence. Now the question is which operator is evaluated first i.e. answer by the associativity rules of the language. An operator can either have left or right associativity.

Consider the expression A − B + C − D

In 'C' language

Left : postfix ++, postfix −−, *, /, %
Right : prefix ++, prefix −−, unary +, unary −

3- Paranthesis

The programmer can alter the precedence and associativity rule by placing the paranthesis in the expressions. The language that allow paranthesis in arithmatic expression could dispense with all the precedence rules and simply associate all the operator L to R or R to L.

The programmer would specify the desire order of evaluation with paranthesis.

4. Conditional expression

Some times if-then-else statements are used to perform a conditional expression assignment. For ex- if (count = 0)
then average = 0

Spiral

else average = Sum/count

In C, C++ and Java this can be specified more conveniently in assignment statement using a conditional expression which has the form :
expression 1 ? expression 2 : expression 3

* Mixed - Mode Assignment

An expression in which the two operand are not the same type is called mixed mode expression. It generate a value whose type is equal to the more capable of the two operand.

One of the design decision concerning about arithmatic operation is whether an operator can have operand of different type. Language that do allow such expressions which are called mixed mode expression must define the convention for implicit operand type conversion.

In FORTRAN, C, C++ use the coercion rule for mixed mode assignment that are similar to those as they used for mixed mode expression that is many of the possible type mix are legal.

The Pascal language include some assignment coercion rule. For ex- integer value can be assigned to real variables but not vice-versa

Spiral

Date.....

C, C++ and Java allow the mixed mode assignment only if the required coercion is widening. So that an int value can be assigned to an float variable but not vice-versa.

In all the language that allows mixed mode assignment the coercion takes place only after the right side expression has been evaluated.

One alternative would be coercion all the operand in the right side to the type of target before evaluation.

* Control statement and control structure

The statement that provide the capability such as assign the value to a variable. is called control statement. Control statement of the first successful programming language FORTRAN designed by the architect of IBM. All were directly related to machine language instructions so that their capabilities had more to do with instruction design than language design.

Control structure is a control statement and the collection of statement whose execution it controls. Control structure should have single entry and single exit.

Spiral

Date.....

* Compound statement

It is a statement which result from the app. of one or more logical connectives to a collection of simple statements. In C language, For ex- {n=1; y=0;}

In ALGOL

```
begin
    statement 1;
    :
    statement n;
end;
```

Compound statement allow the collection of statement that to be abstracted to a single statement. This is powerful concept which can be used to great advantage in control statement design. The data declaration can be added to the beginning of a compound statement in several language which making a block.

* Selection statement

It provides the means of choosing blw two or more execution path in a program.

It falls into two general categories -

(i) Two-way selection statement

Spiral

Date.....

It is based on the collection of design consi-deration. It is the type of expression that control the selector.
All imperative languages include a single-way selector - in most cases as a subform of two way selector. for ex- if-else its a two-way selection bcz it select b/w two diff. action.

(ii) Multiple - selection statement or n-way selection statement

It is used to execute atmost one of the ch-oice of a set of statements presented.

```
switch (expression)
{
  case CONSTANT 1:
    statement 1
    statement 2
    ---
    break;
  case CONSTANT 2:
    statement 1
    statement 2
    ---
}
```

* Iterative Statement

It is the process where a set of inst. or statements are executed repeatedly for a

Spiral

Date.....

specified no. of times or until a condition is met.

(i) counter controlled loop

The type of loop where the no. of execution is known in advance - The value of vari-able which control the execution of the loop i.e. previously known. It also include some means of specifying the initial and terminal variable of the loop variable. It is supported by machine instructions. unfortuna-tely machine architecture approach to progra-mming at a time of architecture design.

(ii) Logically controlled loop

It repeat a collection of statement until some boolean condition changes the value of loop.

For ex-

```
while
loop:
if expression = 0 goto out
[loop body]
goto loop
out
```

Counter controlled loop

For ex-

```
do
{
  ---
}while(condition);
```

* Unconditional branching statement

It is the transfer of the program execution to the labeled program.

Spiral

In C programming goto statement is used to transfer the control from one point to other.

```
goto label
- - -
- - -
label: ←
statement;
```
(Forward jump)

```
label: ←
Statements;
. . .
goto label;
```
(Backward jump)

```
main()
{ int sum = 0;
for(int i=0; i<=10; i++)
{
   sum = sum + i;
   if(i==5)
   {
   goto addition;
   }}
addition:
printf("%d", sum);
}
```

* Guarded Commands

The selection is a list of guarded commands of which one is chosen to execute if more than guard is true. One statement is non-deterministically choosen to be executed in the program. It is an alternative and different form of selection and loop structure that is suggested by Dijkstra.

Suppose we are writing a program that the services interrupt and the interrupt have the same priority. For this we need a construct that choosen among current interrupt in some random ways.

```
if <Boolean Expression> → <Statement>
[ ] <Boolean Expression> → <Statement>
[ ] - - -
Fi
```

All of the boolean expression are evaluated each time the construct is ~~reached~~ during the execution. If more than one expression is true one of the corresponding statement is non-deterministically is choosen for execution. If none is true, a run time error occur that pause the program termination. This force the programmer to consider and list all possibilities as with Ada's Case statement. Consider the following example.

```
if i=0 → sum := sum + i
[ ] i>j → sum := sum + j
[ ] j>i → sum := sum + i
Fi
```

Date

if i=0 & j>1. this construct choose non-deterministically b/w the 1st and 3rd assign-ment. if i=j and is not zero a run time error occur bcz none of the condition is true.

Spiral

Date

Spiral