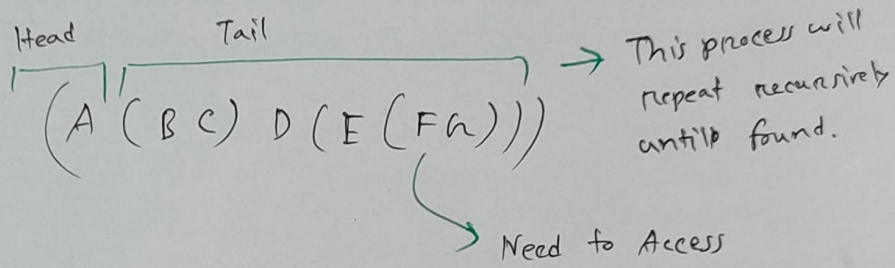
CSE 425 / L-10 / 30.09.2024

Quiz-1
07.10.2024
Tentative

- Linked list will be given
  - show the box pointer diagram
  - write a code in C++
  - how to access the list.



- More on LISP! → 1970 (MIT)
  - Purely functional programming language.
    - → Recursive calling of function
    - → Lambda expression (~~Anno~~ Anonymous)
      - → Lambda Calculus
    - → Not the traditional variable concept
    - → Introduce Symbolic Programming.
- Composite function

$\alpha(x)$, $g(x)$ } $\alpha(g(x))$ } fog, gof

Higher order filter design.

Lambda expression:

cube (x) = x * x * x

cube(2) = ?
= 8

$(\lambda(x)\, x * x * x)\ (2) = 8$

- will be given.
- what will be the output?

- Handware deprecated this language.

Scoping:

- visibility / accessibility of a variable.
- may be in the containing block
- may be from global definition
- sometimes dynamically resolved.
- Benefits: we can reuse a variable name.

Two type of scoping:

1. Static:
   - resolve from containing block then outer containing block, if not found, look for in the global.
2. Dynamic:
   - Local definition in the point of invocation.

containing block.

Scoping:

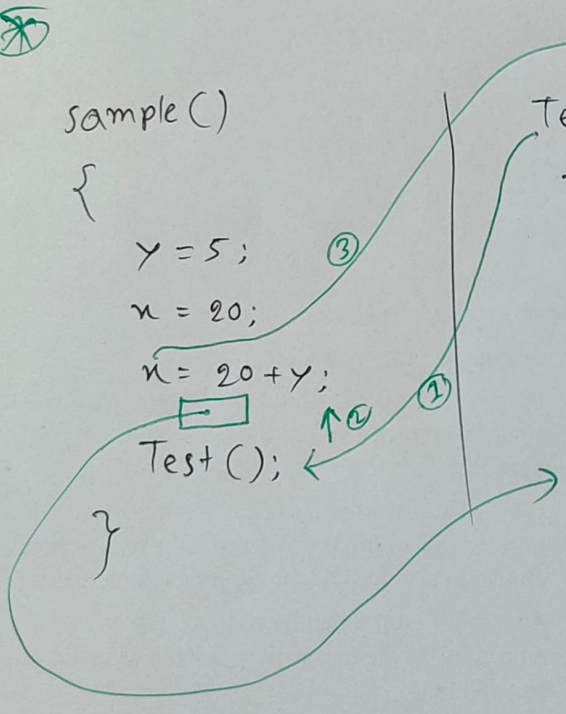
```
int a=10, b=20;

int main()
{
   int a=5;
   {
      int c;
      c = b/a;
      print("%d", c);
   }
}
```

value of b and a missing in the containg block.

value of a found in the parent block but b is still missing.

value of b found on the global definition.



Point of invocation. from which point function was called. In that point function will search for value of x. Means what was the value of x in that position.

```
sample1()
{
   printf(x);
}

sample2()
{
    double x;
    x = 5.5;
    sample1();
}
```

```
x = 0.25;
main()
{
    sample1();
    sample2();
}
```

⇒ What will be the output for static & dynamic? ***

important for exam

Output:

Static:	Dynamic:
0.25	0.25
0.25	5.5

Home-Task:

```
void main()
{
   m(a);
}
void m(y)
{
    a = 1;
    a = y-a;
    n(a);
    printf(a);
}
```

```
a = 3;
void n(x)
{
    x = x * a;
    printf(x);
}
```

What will be the output?

⇒

Static	Dynamic
6	4
2	2

ALGOL 58, 60:

- concept of data types introduced.
- concept of compound statement: { : begin
  ⋮
  } : end
- Identifiers could be of any length.
- Any dimensional array
- Nested statement.

58 → 60: upgrade

- upgraded
- Motivation: Fortran for IBM
  - → Limited portability
  - → still universal program was missing.
  - → for specific computation.

Syntax Analyzer:

- CFG : Content Free Grammer
  - ↳ BNF → EBNF (Modern language use)

For exam:

Functional Programming Basic is important.

L-11/02.10.2024/

Fortran Practice
Next Class Quize - | upto Dynamic Array

L-12/ 07.10.2024/

Quize-1

Syntax Analyzer:

- set of rules that define the 'structure' of the accurate source code.
- we use CFG → Context Free Grammer
  → Another one is BNF (Backus-Naur Form)

Why should this be context free?

⇒

<sentence>
- → <noun-phrase>
  - → <article> ⇒ a, the
  - → <noun> ⇒ Boy, Girl, Cat, Dog
- → <verb-phrase>
  - → <verb> ⇒ sees, pets, bites
  - → <noun-phrase>

<sentence> ⇒ <article> <noun> <verb> <article> <noun>

⇒ a boy sees a dog.

* Making sentences using or following the grammer rules and real meaning is an instance of content sensibility.

In contrast, whatever we can do/make is content flexibility.

* Context free grammer produces all the possible combination and check any thing that is not allowed and doesn't exist in the source code.

* CFG (Context Free Grammer)

* Lexemes:
  - a string of characters that are the lowest syntactical unit.
  - numerical literals.
  - identifiers ⇒ could be a single token
  - reserve words/ keywords (if, else, or, do, while, switch...)

* Tokens:
  - A group of/collection of lexemes given a name.
  - However, sometimes a single lexeme can also form token. ⇒ (+)

* Identifiers:
  - name of variable
  - name of a function
  - a name that identify something.

L-13 / 09.10.2024

Online

Fortran Practice