

Language Evaluation Criteria

Readability : Ease with which programs can be read and understood

Writability : Ease with which a language can be used to create programs

Reliability : Conformation to specifications
↳ Performs as it specifies

Cost : The ultimate cost

- Cost to train programmers to learn the programming languages
- Availability of free compilers
- If the language is highly reliable or not. Because, poor reliability leads to high cost
- Maintaining programs

Reliability : A program is reliable if it performs to its specifications under all conditions. A few issues that have significant effect on reliability are :

- ① Type checking
- ② Exception handling
- ③ Aliasing
- ④ Readability & writability

Type checking :

- Testing for type errors in a given program either by the compiler, or during program execution
- Run-time type checking is expensive, so, compile-time type checking is more desirable
- Also, detecting errors at an earlier stage is more efficient
 - ↳ the earlier it is, the less expensive it is.

Example: In Java, all the type checkings are done at the compile time.

Failure to type check creates numerous problems. For instance in C-programming,

```
void test_fnc (int a)
```

```
{  
  some statements;  
}
```

```
int main (void)
```

```
{  
  some statements;  
  test_fnc (7.707);  
  return 0;
```

```
}
```

type of an actual parameter in a function call was not checked to determine whether the parameter type matches with the type the formal parameter of the fn^c.

As we see, types are mismatched here.

Exception handling :

It is about intercepting run-time errors and take corrective measures.

↳ ability of a programming language to detect error, take corrective measures and then continue to execute accurately is important for reliability.

C++, Java, C# all have extensive capabilities of exception handling
C does not have such facility.

Exception handling process could be predefined and user-defined exceptions and exception handlers.

```
void example () {  
    ... statements  
    average = sum / total  
    ... statements  
    return;  
    when zero-divide {  
        average = 0  
        printf ("Error-divisor (total) is zero");  
    }  
}
```

Here, the exception of division by zero, which is implicitly raised, causes control to transfer to the appropriate handler.

Aliasing :

Presence of two or more distinct referencing methods for the same memory location.

Generally, aliases is a dangerous feature in programming language

However, most programming languages allow some kind of aliasing. For instance, two pointers set to the same variable

↓
In such programs, users must remember that changing the value pointed by one of the two changes the value referenced by the other.

Readability and Writability :

A program written in a language that does not support natural ways to express the required algorithms will use unnatural ways.

↓
Unnatural approaches are less likely to be correct for all possible situations.

Lack of orthogonality

As in C,

values of all data types, except arrays, can be returned from a fn.

Example :

Not Possible
returning an
array from
a fn is not
possible.

```
int test-function (int array-1 [ ])  
{  
    some statements;  
    assignments;  
    control statements;  
    return array-1;  
}
```



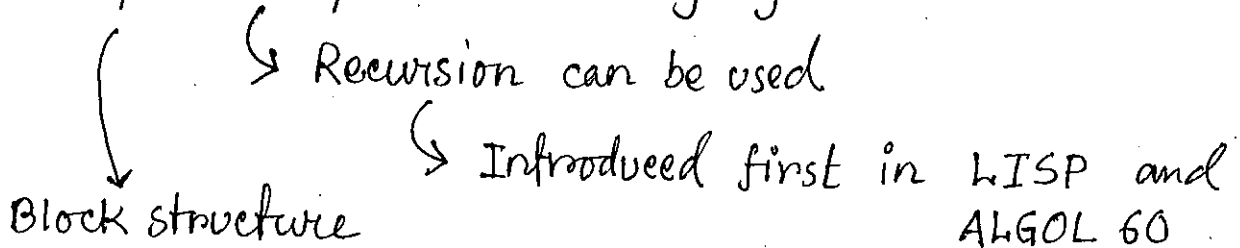
Programming Efficiency

☐ Maintainability:

- How easily new features can be added.
- How easy is it to find and correct errors.
- Good structure, readable syntax, comments, modularity help to achieve maintainability

☐ Expressiveness

- Complex structures and processes can be expressed cleanly, easily, concisely in the language.



☐ Readability

- It is a type of measure to assess how easy a computer program to understand.

↳ easiness to comprehend the computations in a program

For instance, COBOL appears very similar to English language.

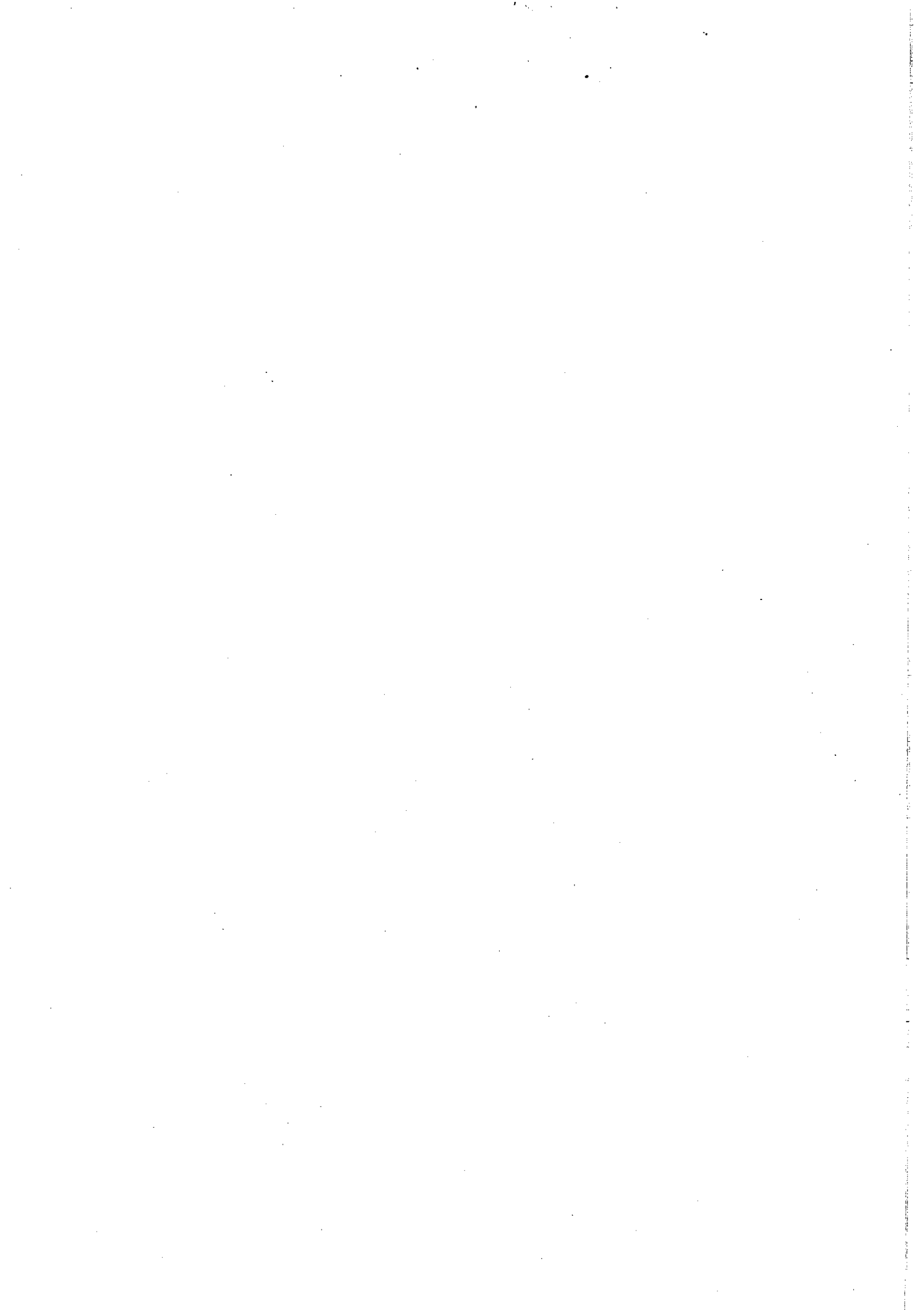
compute : can be used to do arithmetic and store results in a variable.

divide : divide two numbers

multiply : multiply

add add two numbers

move moves from one variable to other



☐ Simplicity

- Language constructs should be simple to understand
- Syntax should be easy so that it can aid the programmer.
- Generality:

~~Language constructs should behave~~
Language should have ^{few} special cases. For instance, "==" in C-language is not general. Because, "==" can not be used to compare two arrays and structures.
↳ we need elementwise comparison.

Another example would be CONSTANTS in PASCAL — because, constants can not be assigned an expression whereas expression assignment is possible for variables.

In C, nesting ^{of} functions are not possible

- Uniformity:

Language constructs with similar meanings should look similar

C does not treat * uniformly.

↳ used for multiplication
↳ for dereferencing pointers

