Author presents the view that a programming language is a tool which should assist the programmer in the most difficult aspects of his art, namely program design, documentation, and debugging.
Author states that programmer should be able to understand language well. Language should be
able to assist programmer in design & debugging. It should be readable and should have less scope for errors.

Good language design can be summarized in following catchphrases :
Simplicity, Security, fast translation, efficient object code and readability.
Security can be tackled by ensuring the compiler used in debugging and for final compilation is same.
Techniques suggested by author to reduce time required for scan and hence overall compilation time :
1. Prescan : if source can be saved in compact form, where scanning need not to be repeated
2. Precompile : precompile part of a program using assembler directives.This is successfully incorporated in ALGOL programs.
3. Dump : Author states independent compilation is a wrong technique. Instead, he advocates for user program being able to execute dump instruction. This would cause complete binary dump of its code and use cases in user named file.
4. Arithmetic expressions : general purpose language required in order to implement the data structures programmer wants
5. Program structures like switch are useful
6. Block structures : security,locality of reference,flexibility
7. Procedures and parameters : easy to understand. Incorrect Calls gives error. Useful to achieve economies of scale
8. Strong typecasting important to avoid errors and inefficiencies introduced in the language.
In conclusion, language designer should consider what programmer really wants.