miniPy 프로젝트 설명서

1. 언어 개요 및 설계 의도

- 목표: C 베이스 miniC를 파이썬풍 문법(들여쓰기 기반, 세미콜론/중괄호 없음)으로 단순화한 언어.
- 동적 정수 타입: 모든 값은 int(long) 하나로 취급해 타입 시스템을 단순화.
- 실행 모델: 코드 생성 없이 AST를 바로 해석하는 인터프리터(`src/interp.c`)로 빠른 피드백.
- 포인터 실험: 파이썬이 제공하지 않는 포인터를 C식 `&`/`*` 표기로 실험해보고자 "셀 핸들" 방식(실제 주소 대신 값 슬롯 참조)으로 도입.

2. 문법(Grammar) 정의

EBNF 개요:

```
program   = stmt_list ;
stmt_list = stmt { stmt } ;
stmt      = simple_stmt NEWLINE | compound_stmt | NEWLINE ;

simple_stmt =
    IDENT "=" expr
  | "*" expr "=" expr
  | PRINT_KW "(" expr ")"
  | IDENT "=" INPUT_KW "(" ")"
  | BREAK_KW
  | CONTINUE_KW
  | RETURN_KW [expr]
  | expr ;

compound_stmt =
    IF_KW expr ":" suite
  | IF_KW expr ":" suite ELSE_KW ":" suite
  | WHILE_KW expr ":" suite
  | FOR_KW IDENT IN_KW RANGE_KW "(" expr "," expr ["," expr] ")" ":" suite
;

suite     = simple_stmt NEWLINE | NEWLINE INDENT stmt_list DEDENT ;

expr      =
    NUMBER | IDENT | "&" IDENT | "*" expr
  | expr ("+"|"-"|"*"|"/"|"%") expr
  | expr ("<"|">"|"<="|">="|"=="|"!=") expr
  | expr ("&&"|"||") expr
  | "!" expr
  | "(" expr ")" ;
```

토큰/렉싱:

- 키워드: `def`, `print`, `input`, `if`, `else`, `while`, `for`, `in`, `range`, `return`, `break`, `continue`

- 연산자: `+ - * / % < > <= >= == != && || ! &`
- 식별자/숫자, 주석 `# ...`
- 줄 단위 들여쓰기 계산으로 `NEWLINE`, `INDENT`, `DEDENT` 생성 → 블록 표현.

3. 전체 구조(흐름)

텍스트 흐름도:

```
소스코드
  └ lexer (parser/scanner.l)
      └ 토큰 + INDENT/DEDENT 생성
         └ parser (parser/parser.y, bison)
            └ AST 구성
               └ interpret(AST) (src/interp.c)
                  └ 실행 결과 출력
```

- **Lexer**: 토큰화, 들여쓰기 스택으로 INDENT/DEDENT 생성, 공백/주석 스킵.
- **Parser**: Bison 문법으로 AST 생성, `program` 액션에서 `interpret` 호출.
- **AST**: `include/ast.h`, `src/ast.c` — 이진 트리 + third 포인터로 조건/반복/블록/포인터/논리/return/break/continue 표현.
- **인터프리터**: `src/interp.c` — 블록 스코프 프레임과 동적 Value로 실행, 포인터는 ref 필드를 통해 역참조/대입.
- **드라이버/빌드**: `src/main.c`, `Makefile` — 입력 파일/STDIN → `yyparse()` → `interpret(AST)`; 코드 생성 단계 없음.

4. 구현된 기능

- 문법: 들여쓰기 기반 파이썬풍, 동적 정수 타입.
- 연산: 산술 `+ - * / %`, 비교 `< > <= >= == !=`, 논리 `&& || !`.
- 제어문: `if / if-else`, `while`, `for x in range(start, end[, step])`, `break`, `continue`, `return`.
- 함수: `def name(params): suite`, `name(args)` 호출, `return` 값 전달 (전역 함수 테이블 기반).
- I/O: `print(expr)`, `input()`(대입과 함께).
- 포인터: `&name`, `*expr`, `*expr = expr` (셀 핸들 모델).
- 스코프: 블록 스코프/섀도잉 지원.
- 테스트: `tests/` 10개 샘플, `run_tests.sh`로 일괄 실행.

5. 미구현/제한 사항

- 타입 한정: 정수만, 문자열/리스트 등 복합 타입 없음.
- 포인터: 실제 주소 아님, 연산 제한적.
- 오류 메시지: 기본 `parse error`, 소스 위치/친절도 부족.
- 문법 경고: Bison shift/reduce 경고 다수(우선순위 정의 단순).
- 코드 생성/IR/최적화 단계 없음.
- 일부 테스트(`test10_pointer_chain`) 결과는 단순 포인터 모델 한계로 6,5,6,5에 고정.