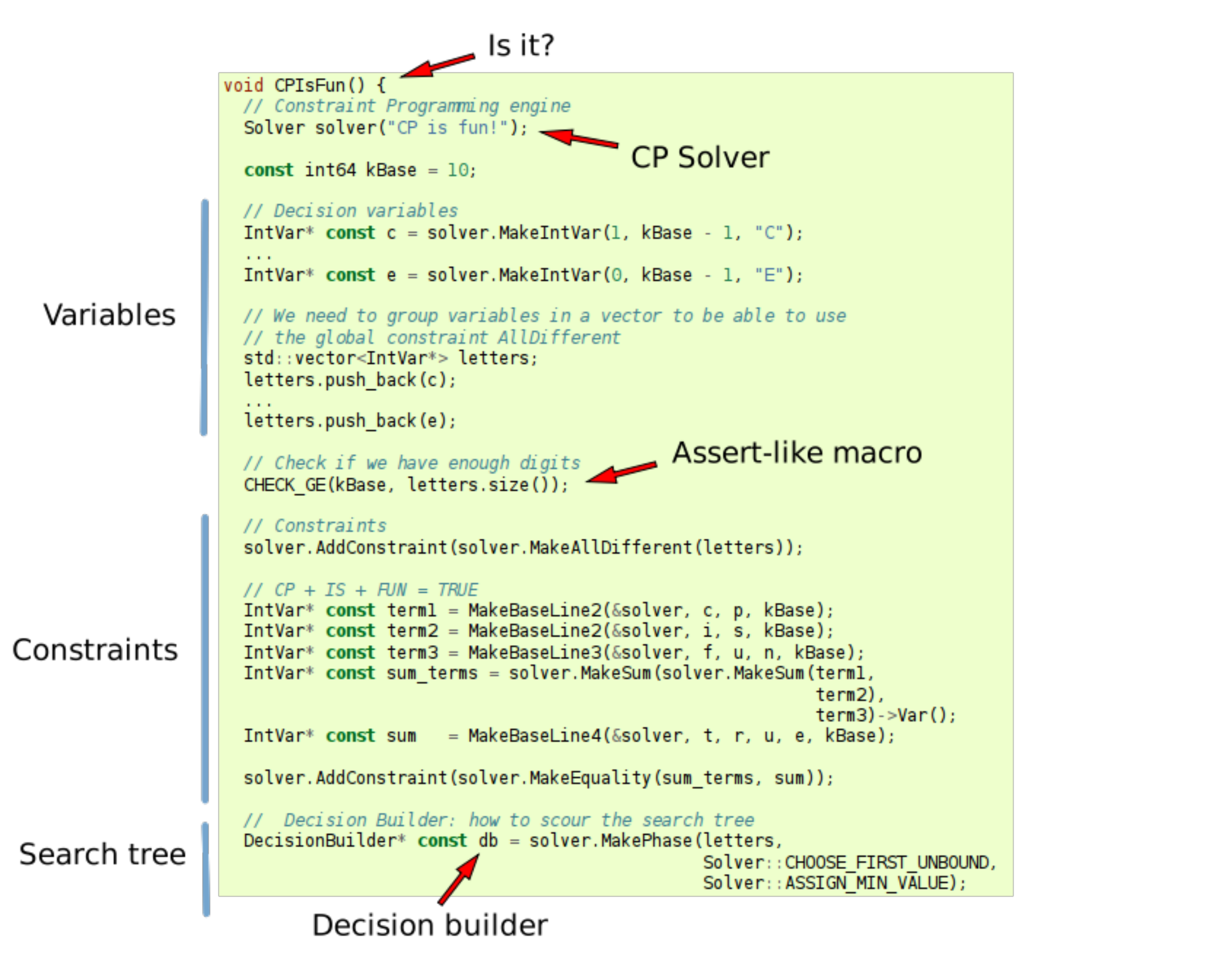


```
void CPIsFun() {
  // Constraint Programming engine
  Solver solver("CP is fun!");

  const int64 kBase = 10;

  // Decision variables
  IntVar* const c = solver.MakeIntVar(1, kBase - 1, "C");
  ...
  IntVar* const e = solver.MakeIntVar(0, kBase - 1, "E");

  // We need to group variables in a vector to be able to use
  // the global constraint AllDifferent
  std::vector<IntVar*> letters;
  letters.push_back(c);
  ...
  letters.push_back(e);

  // Check if we have enough digits
  CHECK_GE(kBase, letters.size());

  // Constraints
  solver.AddConstraint(solver.MakeAllDifferent(letters));

  // CP + IS + FUN = TRUE
  IntVar* const term1 = MakeBaseLine2(&solver, c, p, kBase);
  IntVar* const term2 = MakeBaseLine2(&solver, i, s, kBase);
  IntVar* const term3 = MakeBaseLine3(&solver, f, u, n, kBase);
  IntVar* const sum_terms = solver.MakeSum(solver.MakeSum(term1,
                                                          term2),
                                           term3)->Var();
  IntVar* const sum   = MakeBaseLine4(&solver, t, r, u, e, kBase);

  solver.AddConstraint(solver.MakeEquality(sum_terms, sum));

  //  Decision Builder: how to scour the search tree
  DecisionBuilder* const db = solver.MakePhase(letters,
                                               Solver::CHOOSE_FIRST_UNBOUND,
                                               Solver::ASSIGN_MIN_VALUE);
```