

Báo cáo LAB05

IT3323 Mã lớp 161269

Bài 5 : Type checking part I

Họ và tên: Dương Công Thuyết

MSSV: 20225932

- I. Mô tả hàm compileConstant trong parser.c và ứng dụng của hàm đó trong kiểm tra kiểu
 1. Chức năng

Hàm compileConstant() có nhiệm vụ phân tích cú pháp và kiểm tra kiểu của hằng số có dấu (signed constants) trong phần khai báo CONST của chương trình KPL.

2. Cơ chế hoạt động

Hàm xử lý bốn trường hợp chính:

Trường hợp 1: Hằng số có dấu cộng (+)

```
case SB_PLUS:  
    eat(SB_PLUS);  
  
    constValue = compileConstant2();  
  
    if (constValue->type != TP_INT)  
        error(ERR_TYPE_INCONSISTENCY, currentToken->lineNo, currentToken->colNo);  
  
    break;
```

Loại bỏ dấu +

Gọi compileConstant2() để lấy giá trị hằng số

Kiểm tra kiểu: So sánh constValue->type != TP_INT

Nếu không phải integer → báo lỗi ERR_TYPE_INCONSISTENCY

Trường hợp 2: Hằng số có dấu trừ (-)

```
case SB_MINUS:  
    eat(SB_MINUS);  
  
    constValue = compileConstant2();  
  
    if (constValue->type != TP_INT)  
        error(ERR_TYPE_INCONSISTENCY, currentToken->lineNo, currentToken->colNo);  
  
    constValue->intValue = -constValue->intValue;
```

```
        break;
```

Loại bỏ dấu -

Gọi compileConstant2() để lấy giá trị

Kiểm tra kiểu: So sánh constValue->type != TP_INT

Nếu đúng là integer → đảo dấu giá trị: constValue->intValue = -constValue->intValue

Nếu không → báo lỗi ERR_TYPE_INCONSISTENCY

Trường hợp 3: Hằng ký tự

```
case TK_CHAR:  
    eat(TK_CHAR);  
    constValue = makeCharConstant(currentToken->string[0]);  
    break;
```

Nhận diện token ký tự 'X'

Tạo hằng số ký tự: makeCharConstant(currentToken->string[0])

Không cần kiểm tra kiểu vì ký tự không được phép có dấu

Trường hợp 4: Hằng số không dấu

```
default:  
    constValue = compileConstant2();  
    break;
```

Gọi compileConstant2() để xử lý số nguyên hoặc định danh hằng

Hàm con sẽ xử lý các trường hợp: số nguyên (TK_NUMBER) hoặc tên hằng (TK_IDENT)

3. Ứng dụng trong kiểm tra kiểu

Quy tắc kiểm tra

Hàm áp dụng quy tắc: Chỉ hằng số kiểu INTEGER mới được phép có dấu +/-

Vị trí sử dụng

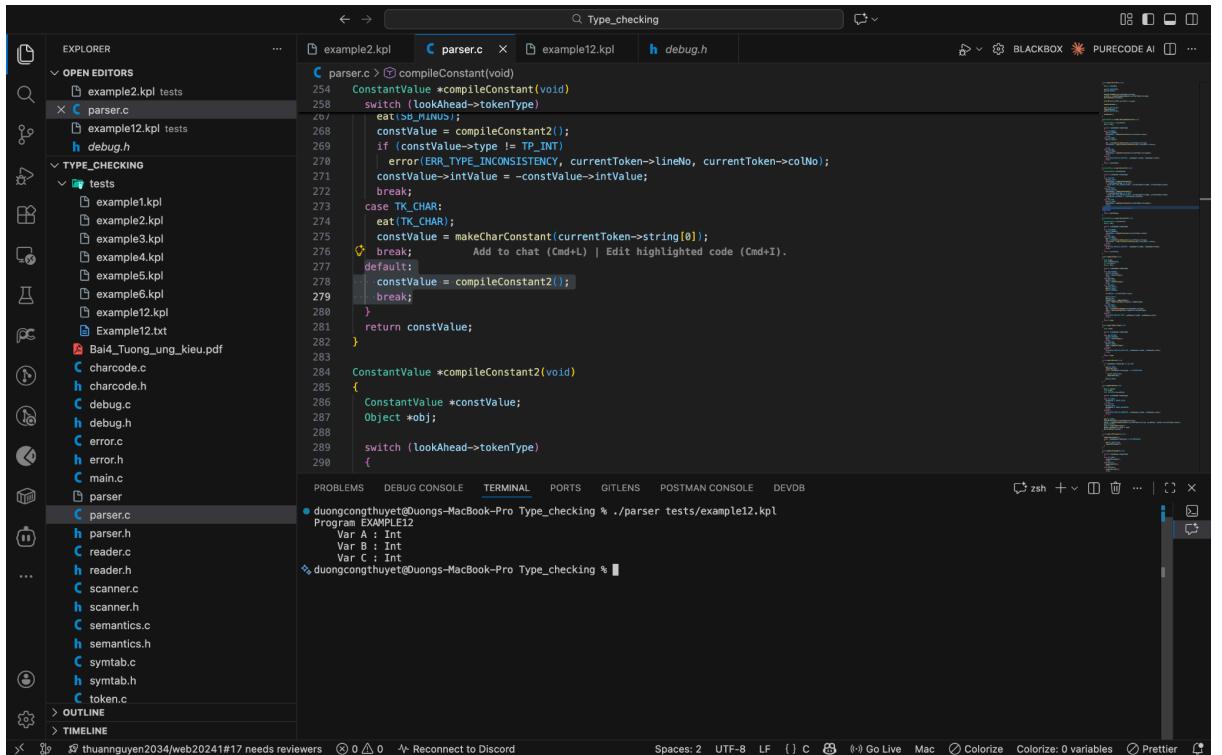
Được gọi trong hàm compileBlock() khi xử lý khai báo hằng số:

Ý nghĩa

Hàm đảm bảo tính nhất quán về kiểu dữ liệu ngay từ giai đoạn khai báo hằng số, ngăn chặn các lỗi như áp dụng toán tử số học (+/-) cho kiểu ký tự - một thao tác

không hợp lệ trong ngôn ngữ KPL. Việc kiểm tra này được thực hiện thông qua điều kiện `constValue->type != TP_INT` và báo lỗi ngay lập tức khi phát hiện vi phạm.

II. Kết quả thực hiện với EXAMPLE12.KPL không có lỗi



```
example2.kpl parser.c example12.kpl debug.h
C parser.c > compileConstant(void)
254 ConstantValue *compileConstant(void)
255     switch (lookAhead->tokenType)
256         eat(LB_MINUS);
257         constValue = compileConstant();
258         if (constValue->type != TP_INT)
259             error(ERR_TYPE_INCONSISTENCY, currentToken->lineNo, currentToken->colNo);
260         constValue->intValue = -constValue->intValue;
261         break;
262     case TK_CHAR:
263         eat(TK_CHAR);
264         constValue = makeCharConstant(currentToken->string[0]);
265         break; // Add to chat (Cmd+L) | Edit highlighted code (Cmd+I).
266     default:
267         constValue = compileConstant2();
268         break;
269     }
270     return constValue;
271 }
272
273 ConstantValue *compileConstant2(void)
274 {
275     ConstantValue *constValue;
276     Object *obj;
277     switch (lookAhead->tokenType)
278     {
279         case FOR:
280             eat(FOR);
281             constValue = compileFor();
282             break;
283         case WHILE:
284             eat(WHILE);
285             constValue = compileWhile();
286             break;
287         case DO:
288             eat(DO);
289             constValue = compileDo();
290             break;
291         case IF:
292             eat(IF);
293             constValue = compileIf();
294             break;
295         case ELSE:
296             eat(ELSE);
297             constValue = compileElse();
298             break;
299         case REPEAT:
300             eat(REPEAT);
301             constValue = compileRepeat();
302             break;
303         case CASE:
304             eat(CASE);
305             constValue = compileCase();
306             break;
307         case DEFAULT:
308             eat(DEFAULT);
309             constValue = compileDefault();
310             break;
311         case END:
312             eat(END);
313             constValue = compileEnd();
314             break;
315         case RETURN:
316             eat(RETURN);
317             constValue = compileReturn();
318             break;
319         case EXIT:
320             eat(EXIT);
321             constValue = compileExit();
322             break;
323         case GOTO:
324             eat(GOTO);
325             constValue = compileGoto();
326             break;
327         case GOSUB:
328             eat(GOSUB);
329             constValue = compileGosub();
330             break;
331         case ENDPROC:
332             eat(ENDPROC);
333             constValue = compileEndProc();
334             break;
335         case ENDVAR:
336             eat(ENDVAR);
337             constValue = compileEndVar();
338             break;
339         case ENDFOR:
340             eat(ENDFOR);
341             constValue = compileEndFor();
342             break;
343         case ENDDO:
344             eat(ENDDO);
345             constValue = compileEndDo();
346             break;
347         case ENDCASE:
348             eat(ENDCASE);
349             constValue = compileEndCase();
350             break;
351         case ENDCOMPARE:
352             eat(ENDCOMPARE);
353             constValue = compileEndCompare();
354             break;
355         case ENDCONDITION:
356             eat(ENDCONDITION);
357             constValue = compileEndCondition();
358             break;
359         case ENDCOMMENT:
360             eat(ENDCOMMENT);
361             constValue = compileEndComment();
362             break;
363         case ENDCOMMENT2:
364             eat(ENDCOMMENT2);
365             constValue = compileEndComment2();
366             break;
367         case ENDCOMMENT3:
368             eat(ENDCOMMENT3);
369             constValue = compileEndComment3();
370             break;
371         case ENDCOMMENT4:
372             eat(ENDCOMMENT4);
373             constValue = compileEndComment4();
374             break;
375         case ENDCOMMENT5:
376             eat(ENDCOMMENT5);
377             constValue = compileEndComment5();
378             break;
379         case ENDCOMMENT6:
380             eat(ENDCOMMENT6);
381             constValue = compileEndComment6();
382             break;
383         case ENDCOMMENT7:
384             eat(ENDCOMMENT7);
385             constValue = compileEndComment7();
386             break;
387         case ENDCOMMENT8:
388             eat(ENDCOMMENT8);
389             constValue = compileEndComment8();
390             break;
391         case ENDCOMMENT9:
392             eat(ENDCOMMENT9);
393             constValue = compileEndComment9();
394             break;
395         case ENDCOMMENT10:
396             eat(ENDCOMMENT10);
397             constValue = compileEndComment10();
398             break;
399         case ENDCOMMENT11:
400             eat(ENDCOMMENT11);
401             constValue = compileEndComment11();
402             break;
403         case ENDCOMMENT12:
404             eat(ENDCOMMENT12);
405             constValue = compileEndComment12();
406             break;
407         case ENDCOMMENT13:
408             eat(ENDCOMMENT13);
409             constValue = compileEndComment13();
410             break;
411         case ENDCOMMENT14:
412             eat(ENDCOMMENT14);
413             constValue = compileEndComment14();
414             break;
415         case ENDCOMMENT15:
416             eat(ENDCOMMENT15);
417             constValue = compileEndComment15();
418             break;
419         case ENDCOMMENT16:
420             eat(ENDCOMMENT16);
421             constValue = compileEndComment16();
422             break;
423         case ENDCOMMENT17:
424             eat(ENDCOMMENT17);
425             constValue = compileEndComment17();
426             break;
427         case ENDCOMMENT18:
428             eat(ENDCOMMENT18);
429             constValue = compileEndComment18();
430             break;
431         case ENDCOMMENT19:
432             eat(ENDCOMMENT19);
433             constValue = compileEndComment19();
434             break;
435         case ENDCOMMENT20:
436             eat(ENDCOMMENT20);
437             constValue = compileEndComment20();
438             break;
439         case ENDCOMMENT21:
440             eat(ENDCOMMENT21);
441             constValue = compileEndComment21();
442             break;
443         case ENDCOMMENT22:
444             eat(ENDCOMMENT22);
445             constValue = compileEndComment22();
446             break;
447         case ENDCOMMENT23:
448             eat(ENDCOMMENT23);
449             constValue = compileEndComment23();
450             break;
451         case ENDCOMMENT24:
452             eat(ENDCOMMENT24);
453             constValue = compileEndComment24();
454             break;
455         case ENDCOMMENT25:
456             eat(ENDCOMMENT25);
457             constValue = compileEndComment25();
458             break;
459         case ENDCOMMENT26:
460             eat(ENDCOMMENT26);
461             constValue = compileEndComment26();
462             break;
463         case ENDCOMMENT27:
464             eat(ENDCOMMENT27);
465             constValue = compileEndComment27();
466             break;
467         case ENDCOMMENT28:
468             eat(ENDCOMMENT28);
469             constValue = compileEndComment28();
470             break;
471         case ENDCOMMENT29:
472             eat(ENDCOMMENT29);
473             constValue = compileEndComment29();
474             break;
475         case ENDCOMMENT30:
476             eat(ENDCOMMENT30);
477             constValue = compileEndComment30();
478             break;
479         case ENDCOMMENT31:
480             eat(ENDCOMMENT31);
481             constValue = compileEndComment31();
482             break;
483         case ENDCOMMENT32:
484             eat(ENDCOMMENT32);
485             constValue = compileEndComment32();
486             break;
487         case ENDCOMMENT33:
488             eat(ENDCOMMENT33);
489             constValue = compileEndComment33();
490             break;
491         case ENDCOMMENT34:
492             eat(ENDCOMMENT34);
493             constValue = compileEndComment34();
494             break;
495         case ENDCOMMENT35:
496             eat(ENDCOMMENT35);
497             constValue = compileEndComment35();
498             break;
499         case ENDCOMMENT36:
500             eat(ENDCOMMENT36);
501             constValue = compileEndComment36();
502             break;
503         case ENDCOMMENT37:
504             eat(ENDCOMMENT37);
505             constValue = compileEndComment37();
506             break;
507         case ENDCOMMENT38:
508             eat(ENDCOMMENT38);
509             constValue = compileEndComment38();
510             break;
511         case ENDCOMMENT39:
512             eat(ENDCOMMENT39);
513             constValue = compileEndComment39();
514             break;
515         case ENDCOMMENT40:
516             eat(ENDCOMMENT40);
517             constValue = compileEndComment40();
518             break;
519         case ENDCOMMENT41:
520             eat(ENDCOMMENT41);
521             constValue = compileEndComment41();
522             break;
523         case ENDCOMMENT42:
524             eat(ENDCOMMENT42);
525             constValue = compileEndComment42();
526             break;
527         case ENDCOMMENT43:
528             eat(ENDCOMMENT43);
529             constValue = compileEndComment43();
530             break;
531         case ENDCOMMENT44:
532             eat(ENDCOMMENT44);
533             constValue = compileEndComment44();
534             break;
535         case ENDCOMMENT45:
536             eat(ENDCOMMENT45);
537             constValue = compileEndComment45();
538             break;
539         case ENDCOMMENT46:
540             eat(ENDCOMMENT46);
541             constValue = compileEndComment46();
542             break;
543         case ENDCOMMENT47:
544             eat(ENDCOMMENT47);
545             constValue = compileEndComment47();
546             break;
547         case ENDCOMMENT48:
548             eat(ENDCOMMENT48);
549             constValue = compileEndComment48();
550             break;
551         case ENDCOMMENT49:
552             eat(ENDCOMMENT49);
553             constValue = compileEndComment49();
554             break;
555         case ENDCOMMENT50:
556             eat(ENDCOMMENT50);
557             constValue = compileEndComment50();
558             break;
559         case ENDCOMMENT51:
560             eat(ENDCOMMENT51);
561             constValue = compileEndComment51();
562             break;
563         case ENDCOMMENT52:
564             eat(ENDCOMMENT52);
565             constValue = compileEndComment52();
566             break;
567         case ENDCOMMENT53:
568             eat(ENDCOMMENT53);
569             constValue = compileEndComment53();
570             break;
571         case ENDCOMMENT54:
572             eat(ENDCOMMENT54);
573             constValue = compileEndComment54();
574             break;
575         case ENDCOMMENT55:
576             eat(ENDCOMMENT55);
577             constValue = compileEndComment55();
578             break;
579         case ENDCOMMENT56:
580             eat(ENDCOMMENT56);
581             constValue = compileEndComment56();
582             break;
583         case ENDCOMMENT57:
584             eat(ENDCOMMENT57);
585             constValue = compileEndComment57();
586             break;
587         case ENDCOMMENT58:
588             eat(ENDCOMMENT58);
589             constValue = compileEndComment58();
590             break;
591         case ENDCOMMENT59:
592             eat(ENDCOMMENT59);
593             constValue = compileEndComment59();
594             break;
595         case ENDCOMMENT60:
596             eat(ENDCOMMENT60);
597             constValue = compileEndComment60();
598             break;
599         case ENDCOMMENT61:
600             eat(ENDCOMMENT61);
601             constValue = compileEndComment61();
602             break;
603         case ENDCOMMENT62:
604             eat(ENDCOMMENT62);
605             constValue = compileEndComment62();
606             break;
607         case ENDCOMMENT63:
608             eat(ENDCOMMENT63);
609             constValue = compileEndComment63();
610             break;
611         case ENDCOMMENT64:
612             eat(ENDCOMMENT64);
613             constValue = compileEndComment64();
614             break;
615         case ENDCOMMENT65:
616             eat(ENDCOMMENT65);
617             constValue = compileEndComment65();
618             break;
619         case ENDCOMMENT66:
620             eat(ENDCOMMENT66);
621             constValue = compileEndComment66();
622             break;
623         case ENDCOMMENT67:
624             eat(ENDCOMMENT67);
625             constValue = compileEndComment67();
626             break;
627         case ENDCOMMENT68:
628             eat(ENDCOMMENT68);
629             constValue = compileEndComment68();
630             break;
631         case ENDCOMMENT69:
632             eat(ENDCOMMENT69);
633             constValue = compileEndComment69();
634             break;
635         case ENDCOMMENT70:
636             eat(ENDCOMMENT70);
637             constValue = compileEndComment70();
638             break;
639         case ENDCOMMENT71:
640             eat(ENDCOMMENT71);
641             constValue = compileEndComment71();
642             break;
643         case ENDCOMMENT72:
644             eat(ENDCOMMENT72);
645             constValue = compileEndComment72();
646             break;
647         case ENDCOMMENT73:
648             eat(ENDCOMMENT73);
649             constValue = compileEndComment73();
650             break;
651         case ENDCOMMENT74:
652             eat(ENDCOMMENT74);
653             constValue = compileEndComment74();
654             break;
655         case ENDCOMMENT75:
656             eat(ENDCOMMENT75);
657             constValue = compileEndComment75();
658             break;
659         case ENDCOMMENT76:
660             eat(ENDCOMMENT76);
661             constValue = compileEndComment76();
662             break;
663         case ENDCOMMENT77:
664             eat(ENDCOMMENT77);
665             constValue = compileEndComment77();
666             break;
667         case ENDCOMMENT78:
668             eat(ENDCOMMENT78);
669             constValue = compileEndComment78();
670             break;
671         case ENDCOMMENT79:
672             eat(ENDCOMMENT79);
673             constValue = compileEndComment79();
674             break;
675         case ENDCOMMENT80:
676             eat(ENDCOMMENT80);
677             constValue = compileEndComment80();
678             break;
679         case ENDCOMMENT81:
680             eat(ENDCOMMENT81);
681             constValue = compileEndComment81();
682             break;
683         case ENDCOMMENT82:
684             eat(ENDCOMMENT82);
685             constValue = compileEndComment82();
686             break;
687         case ENDCOMMENT83:
688             eat(ENDCOMMENT83);
689             constValue = compileEndComment83();
690             break;
691         case ENDCOMMENT84:
692             eat(ENDCOMMENT84);
693             constValue = compileEndComment84();
694             break;
695         case ENDCOMMENT85:
696             eat(ENDCOMMENT85);
697             constValue = compileEndComment85();
698             break;
699         case ENDCOMMENT86:
700             eat(ENDCOMMENT86);
701             constValue = compileEndComment86();
702             break;
703         case ENDCOMMENT87:
704             eat(ENDCOMMENT87);
705             constValue = compileEndComment87();
706             break;
707         case ENDCOMMENT88:
708             eat(ENDCOMMENT88);
709             constValue = compileEndComment88();
710             break;
711         case ENDCOMMENT89:
712             eat(ENDCOMMENT89);
713             constValue = compileEndComment89();
714             break;
715         case ENDCOMMENT90:
716             eat(ENDCOMMENT90);
717             constValue = compileEndComment90();
718             break;
719         case ENDCOMMENT91:
720             eat(ENDCOMMENT91);
721             constValue = compileEndComment91();
722             break;
723         case ENDCOMMENT92:
724             eat(ENDCOMMENT92);
725             constValue = compileEndComment92();
726             break;
727         case ENDCOMMENT93:
728             eat(ENDCOMMENT93);
729             constValue = compileEndComment93();
730             break;
731         case ENDCOMMENT94:
732             eat(ENDCOMMENT94);
733             constValue = compileEndComment94();
734             break;
735         case ENDCOMMENT95:
736             eat(ENDCOMMENT95);
737             constValue = compileEndComment95();
738             break;
739         case ENDCOMMENT96:
740             eat(ENDCOMMENT96);
741             constValue = compileEndComment96();
742             break;
743         case ENDCOMMENT97:
744             eat(ENDCOMMENT97);
745             constValue = compileEndComment97();
746             break;
747         case ENDCOMMENT98:
748             eat(ENDCOMMENT98);
749             constValue = compileEndComment98();
750             break;
751         case ENDCOMMENT99:
752             eat(ENDCOMMENT99);
753             constValue = compileEndComment99();
754             break;
755         case ENDCOMMENT100:
756             eat(ENDCOMMENT100);
757             constValue = compileEndComment100();
758             break;
759         case ENDCOMMENT101:
760             eat(ENDCOMMENT101);
761             constValue = compileEndComment101();
762             break;
763         case ENDCOMMENT102:
764             eat(ENDCOMMENT102);
765             constValue = compileEndComment102();
766             break;
767         case ENDCOMMENT103:
768             eat(ENDCOMMENT103);
769             constValue = compileEndComment103();
770             break;
771         case ENDCOMMENT104:
772             eat(ENDCOMMENT104);
773             constValue = compileEndComment104();
774             break;
775         case ENDCOMMENT105:
776             eat(ENDCOMMENT105);
777             constValue = compileEndComment105();
778             break;
779         case ENDCOMMENT106:
780             eat(ENDCOMMENT106);
781             constValue = compileEndComment106();
782             break;
783         case ENDCOMMENT107:
784             eat(ENDCOMMENT107);
785             constValue = compileEndComment107();
786             break;
787         case ENDCOMMENT108:
788             eat(ENDCOMMENT108);
789             constValue = compileEndComment108();
790             break;
791         case ENDCOMMENT109:
792             eat(ENDCOMMENT109);
793             constValue = compileEndComment109();
794             break;
795         case ENDCOMMENT110:
796             eat(ENDCOMMENT110);
797             constValue = compileEndComment110();
798             break;
799         case ENDCOMMENT111:
800             eat(ENDCOMMENT111);
801             constValue = compileEndComment111();
802             break;
803         case ENDCOMMENT112:
804             eat(ENDCOMMENT112);
805             constValue = compileEndComment112();
806             break;
807         case ENDCOMMENT113:
808             eat(ENDCOMMENT113);
809             constValue = compileEndComment113();
810             break;
811         case ENDCOMMENT114:
812             eat(ENDCOMMENT114);
813             constValue = compileEndComment114();
814             break;
815         case ENDCOMMENT115:
816             eat(ENDCOMMENT115);
817             constValue = compileEndComment115();
818             break;
819         case ENDCOMMENT116:
820             eat(ENDCOMMENT116);
821             constValue = compileEndComment116();
822             break;
823         case ENDCOMMENT117:
824             eat(ENDCOMMENT117);
825             constValue = compileEndComment117();
826             break;
827         case ENDCOMMENT118:
828             eat(ENDCOMMENT118);
829             constValue = compileEndComment118();
830             break;
831         case ENDCOMMENT119:
832             eat(ENDCOMMENT119);
833             constValue = compileEndComment119();
834             break;
835         case ENDCOMMENT120:
836             eat(ENDCOMMENT120);
837             constValue = compileEndComment120();
838             break;
839         case ENDCOMMENT121:
840             eat(ENDCOMMENT121);
841             constValue = compileEndComment121();
842             break;
843         case ENDCOMMENT122:
844             eat(ENDCOMMENT122);
845             constValue = compileEndComment122();
846             break;
847         case ENDCOMMENT123:
848             eat(ENDCOMMENT123);
849             constValue = compileEndComment123();
850             break;
851         case ENDCOMMENT124:
852             eat(ENDCOMMENT124);
853             constValue = compileEndComment124();
854             break;
855         case ENDCOMMENT125:
856             eat(ENDCOMMENT125);
857             constValue = compileEndComment125();
858             break;
859         case ENDCOMMENT126:
860             eat(ENDCOMMENT126);
861             constValue = compileEndComment126();
862             break;
863         case ENDCOMMENT127:
864             eat(ENDCOMMENT127);
865             constValue = compileEndComment127();
866             break;
867         case ENDCOMMENT128:
868             eat(ENDCOMMENT128);
869             constValue = compileEndComment128();
870             break;
871         case ENDCOMMENT129:
872             eat(ENDCOMMENT129);
873             constValue = compileEndComment129();
874             break;
875         case ENDCOMMENT130:
876             eat(ENDCOMMENT130);
877             constValue = compileEndComment130();
878             break;
879         case ENDCOMMENT131:
880             eat(ENDCOMMENT131);
881             constValue = compileEndComment131();
882             break;
883         case ENDCOMMENT132:
884             eat(ENDCOMMENT132);
885             constValue = compileEndComment132();
886             break;
887         case ENDCOMMENT133:
888             eat(ENDCOMMENT133);
889             constValue = compileEndComment133();
890             break;
891         case ENDCOMMENT134:
892             eat(ENDCOMMENT134);
893             constValue = compileEndComment134();
894             break;
895         case ENDCOMMENT135:
896             eat(ENDCOMMENT135);
897             constValue = compileEndComment135();
898             break;
899         case ENDCOMMENT136:
900             eat(ENDCOMMENT136);
901             constValue = compileEndComment136();
902             break;
903         case ENDCOMMENT137:
904             eat(ENDCOMMENT137);
905             constValue = compileEndComment137();
906             break;
907         case ENDCOMMENT138:
908             eat(ENDCOMMENT138);
909             constValue = compileEndComment138();
910             break;
911         case ENDCOMMENT139:
912             eat(ENDCOMMENT139);
913             constValue = compileEndComment139();
914             break;
915         case ENDCOMMENT140:
916             eat(ENDCOMMENT140);
917             constValue = compileEndComment140();
918             break;
919         case ENDCOMMENT141:
920             eat(ENDCOMMENT141);
921             constValue = compileEndComment141();
922             break;
923         case ENDCOMMENT142:
924             eat(ENDCOMMENT142);
925             constValue = compileEndComment142();
926             break;
927         case ENDCOMMENT143:
928             eat(ENDCOMMENT143);
929             constValue = compileEndComment143();
930             break;
931         case ENDCOMMENT144:
932             eat(ENDCOMMENT144);
933             constValue = compileEndComment144();
934             break;
935         case ENDCOMMENT145:
936             eat(ENDCOMMENT145);
937             constValue = compileEndComment145();
938             break;
939         case ENDCOMMENT146:
940             eat(ENDCOMMENT146);
941             constValue = compileEndComment146();
942             break;
943         case ENDCOMMENT147:
944             eat(ENDCOMMENT147);
945             constValue = compileEndComment147();
946             break;
947         case ENDCOMMENT148:
948             eat(ENDCOMMENT148);
949             constValue = compileEndComment148();
950             break;
951         case ENDCOMMENT149:
952             eat(ENDCOMMENT149);
953             constValue = compileEndComment149();
954             break;
955         case ENDCOMMENT150:
956             eat(ENDCOMMENT150);
957             constValue = compileEndComment150();
958             break;
959         case ENDCOMMENT151:
960             eat(ENDCOMMENT151);
961             constValue = compileEndComment151();
962             break;
963         case ENDCOMMENT152:
964             eat(ENDCOMMENT152);
965             constValue = compileEndComment152();
966             break;
967         case ENDCOMMENT153:
968             eat(ENDCOMMENT153);
969             constValue = compileEndComment153();
970             break;
971         case ENDCOMMENT154:
972             eat(ENDCOMMENT154);
973             constValue = compileEndComment154();
974             break;
975         case ENDCOMMENT155:
976             eat(ENDCOMMENT155);
977             constValue = compileEndComment155();
978             break;
979         case ENDCOMMENT156:
980             eat(ENDCOMMENT156);
981             constValue = compileEndComment156();
982             break;
983         case ENDCOMMENT157:
984             eat(ENDCOMMENT157);
985             constValue = compileEndComment157();
986             break;
987         case ENDCOMMENT158:
988             eat(ENDCOMMENT158);
989             constValue = compileEndComment158();
990             break;
991         case ENDCOMMENT159:
992             eat(ENDCOMMENT159);
993             constValue = compileEndComment159();
994             break;
995         case ENDCOMMENT160:
996             eat(ENDCOMMENT160);
997             constValue = compileEndComment160();
998             break;
999         case ENDCOMMENT161:
1000            eat(ENDCOMMENT161);
1001            constValue = compileEndComment161();
1002            break;
1003        case ENDCOMMENT162:
1004            eat(ENDCOMMENT162);
1005            constValue = compileEndComment162();
1006            break;
1007        case ENDCOMMENT163:
1008            eat(ENDCOMMENT163);
1009            constValue = compileEndComment163();
1010            break;
1011        case ENDCOMMENT164:
1012            eat(ENDCOMMENT164);
1013            constValue = compileEndComment164();
1014            break;
1015        case ENDCOMMENT165:
1016            eat(ENDCOMMENT165);
1017            constValue = compileEndComment165();
1018            break;
1019        case ENDCOMMENT166:
1020            eat(ENDCOMMENT166);
1021            constValue = compileEndComment166();
1022            break;
1023        case ENDCOMMENT167:
1024            eat(ENDCOMMENT167);
1025            constValue = compileEndComment167();
1026            break;
1027        case ENDCOMMENT168:
1028            eat(ENDCOMMENT168);
1029            constValue = compileEndComment168();
1030            break;
1031        case ENDCOMMENT169:
1032            eat(ENDCOMMENT169);
1033            constValue = compileEndComment169();
1034            break;
1035        case ENDCOMMENT170:
1036            eat(ENDCOMMENT170);
1037            constValue = compileEndComment170();
1038            break;
1039        case ENDCOMMENT171:
1040            eat(ENDCOMMENT171);
1041            constValue = compileEndComment171();
1042            break;
1043        case ENDCOMMENT172:
1044            eat(ENDCOMMENT172);
1045            constValue = compileEndComment172();
1046            break;
1047        case ENDCOMMENT173:
1048            eat(ENDCOMMENT173);
1049            constValue = compileEndComment173();
1050            break;
1051        case ENDCOMMENT174:
1052            eat(ENDCOMMENT174);
1053            constValue = compileEndComment174();
1054            break;
1055        case ENDCOMMENT175:
1056            eat(ENDCOMMENT175);
1057            constValue = compileEndComment175();
1058            break;
1059        case ENDCOMMENT176:
1060            eat(ENDCOMMENT176);
1061            constValue = compileEndComment176();
1062            break;
1063        case ENDCOMMENT177:
1064            eat(ENDCOMMENT177);
1065            constValue = compileEndComment177();
1066            break;
1067        case ENDCOMMENT178:
1068            eat(ENDCOMMENT178);
1069            constValue = compileEndComment178();
1070            break;
1071        case ENDCOMMENT179:
1072            eat(ENDCOMMENT179);
1073            constValue = compileEndComment179();
1074            break;
1075        case ENDCOMMENT180:
1076            eat(ENDCOMMENT180);
1077            constValue = compileEndComment180();
1078            break;
1079        case ENDCOMMENT181:
1080            eat(ENDCOMMENT181);
1081            constValue = compileEndComment181();
1082            break;
1083        case ENDCOMMENT182:
1084            eat(ENDCOMMENT182);
1085            constValue = compileEndComment182();
1086            break;
1087        case ENDCOMMENT183:
1088            eat(ENDCOMMENT183);
1089            constValue = compileEndComment183();
1090            break;
1091        case ENDCOMMENT184:
1092            eat(ENDCOMMENT184);
1093            constValue = compileEndComment184();
1094            break;
1095        case ENDCOMMENT185:
1096            eat(ENDCOMMENT185);
1097            constValue = compileEndComment185();
1098            break;
1099        case ENDCOMMENT186:
1100            eat(ENDCOMMENT186);
1101            constValue = compileEndComment186();
1102            break;
1103        case ENDCOMMENT187:
1104            eat(ENDCOMMENT187);
1105            constValue = compileEndComment187();
1106            break;
1107        case ENDCOMMENT188:
1108            eat(ENDCOMMENT188);
1109            constValue = compileEndComment188();
1110            break;
1111        case ENDCOMMENT189:
1112            eat(ENDCOMMENT189);
1113            constValue = compileEndComment189();
1114            break;
1115        case ENDCOMMENT190:
1116            eat(ENDCOMMENT190);
1117            constValue = compileEndComment190();
1118            break;
1119        case ENDCOMMENT191:
1120            eat(ENDCOMMENT191);
1121            constValue = compileEndComment191();
1122            break;
1123        case ENDCOMMENT192:
1124            eat(ENDCOMMENT192);
1125            constValue = compileEndComment192();
1126            break;
1127        case ENDCOMMENT193:
1128            eat(ENDCOMMENT193);
1129            constValue = compileEndComment193();
1130            break;
1131        case ENDCOMMENT194:
1132            eat(ENDCOMMENT194);
1133            constValue = compileEndComment194();
1134            break;
1135        case ENDCOMMENT195:
1136            eat(ENDCOMMENT195);
1137            constValue = compileEndComment195();
1138            break;
1139        case ENDCOMMENT196:
1140            eat(ENDCOMMENT196);
1141            constValue = compileEndComment196();
1142            break;
1143        case ENDCOMMENT197:
1144            eat(ENDCOMMENT197);
1145            constValue = compileEndComment197();
1146            break;
1147        case ENDCOMMENT198:
1148            eat(ENDCOMMENT198);
1149            constValue = compileEndComment198();
1150            break;
1151        case ENDCOMMENT199:
1152            eat(ENDCOMMENT199);
1153            constValue = compileEndComment199();
1154            break;
1155        case ENDCOMMENT200:
1156            eat(ENDCOMMENT200);
1157            constValue = compileEndComment200();
1158            break;
1159        case ENDCOMMENT201:
1160            eat(ENDCOMMENT201);
1161            constValue = compileEndComment201();
1162            break;
1163        case ENDCOMMENT202:
1164            eat(ENDCOMMENT202);
1165            constValue = compileEndComment202();
1166            break;
1167        case ENDCOMMENT203:
1168            eat(ENDCOMMENT203);
1169            constValue = compileEndComment203();
1170            break;
1171        case ENDCOMMENT204:
1172            eat(ENDCOMMENT204);
1173            constValue = compileEndComment204();
1174            break;
1175        case ENDCOMMENT205:
1176            eat(ENDCOMMENT205);
1177            constValue = compileEndComment205();
1178            break;
1179        case ENDCOMMENT206:
1180            eat(ENDCOMMENT206);
1181            constValue = compileEndComment206();
1182            break;
1183        case ENDCOMMENT207:
1184            eat(ENDCOMMENT207);
1185            constValue = compileEndComment207();
1186            break;
1187        case ENDCOMMENT208:
1188            eat(ENDCOMMENT208);
1189            constValue = compileEndComment208();
1190            break;
1191        case ENDCOMMENT209:
1192            eat(ENDCOMMENT209);
1193            constValue = compileEndComment209();
1194            break;
1195        case ENDCOMMENT210:
1196            eat(ENDCOMMENT210);
1197            constValue = compileEndComment210();
1198            break;
1199        case ENDCOMMENT211:
1200            eat(ENDCOMMENT211);
1201            constValue = compileEndComment211();
1202            break;
1203        case ENDCOMMENT212:
1204            eat(ENDCOMMENT212);
1205            constValue = compileEndComment212();
1206            break;
1207        case ENDCOMMENT213:
1208            eat(ENDCOMMENT213);
1209            constValue = compileEndComment213();
1210            break;
1211        case ENDCOMMENT214:
1212            eat(ENDCOMMENT214);
1213            constValue = compileEndComment214();
1214            break;
1215        case ENDCOMMENT215:
1216            eat(ENDCOMMENT215);
1217            constValue = compileEndComment215();
1218            break;
1219        case ENDCOMMENT216:
1220            eat(ENDCOMMENT216);
1221            constValue = compileEndComment216();
1222            break;
1223        case ENDCOMMENT217:
1224            eat(ENDCOMMENT217);
1225            constValue = compileEndComment217();
1226            break;
1227        case ENDCOMMENT218:
1228            eat(ENDCOMMENT218);
1229            constValue = compileEndComment218();
1230            break;
1231        case ENDCOMMENT219:
1232            eat(ENDCOMMENT219);
1233            constValue = compileEndComment219();
1234            break;
1235        case ENDCOMMENT220:
1236            eat(ENDCOMMENT220);
1237            constValue = compileEndComment220();
1238            break;
1239        case ENDCOMMENT221:
1240            eat(ENDCOMMENT221);
1241            constValue = compileEndComment221();
1242            break;
1243        case ENDCOMMENT222:
1244            eat(ENDCOMMENT222);
1245            constValue = compileEndComment222();
1246            break;
1247        case ENDCOMMENT223:
1248            eat(ENDCOMMENT223);
1249            constValue = compileEndComment223();
1250            break;
1251        case ENDCOMMENT224:
1252            eat(ENDCOMMENT224);
1253            constValue = compileEndComment224();
1254            break;
1255        case ENDCOMMENT225:
1256            eat(ENDCOMMENT225);
1257            constValue = compileEndComment225();
1258            break;
1259        case ENDCOMMENT226:
1260            eat(ENDCOMMENT226);
1261            constValue = compileEndComment226();
1262            break;
1263        case ENDCOMMENT227:
1264            eat(ENDCOMMENT227);
1265            constValue = compileEndComment227();
1266            break;
1267        case ENDCOMMENT228:
1268            eat(ENDCOMMENT228);
1269            constValue = compileEndComment228();
1270            break;
1271        case ENDCOMMENT229:
1
```

Nguyên nhân

Trong hàm parser.c có ba lần kiểm tra kiểu:

```
void compileForSt(void)
{
    Type *varType;
    Type *type1;
    Type *type2;

    eat(KW_FOR);
    eat(TK_IDENT);

    // check if the identifier is a variable
    varType = checkDeclaredVariable(currentToken->string)->varAttrs->type;
    checkIntType(varType);           // ← Kiểm tra 1: Biến phải là INTEGER

    eat(SB_ASSIGN);
    type1 = compileExpression();

    checkIntType(type1);           // ← Kiểm tra 2: Giá trị bắt đầu phải là INTEGER

    eat(KW_TO);
    type2 = compileExpression();

    checkIntType(type2);           // ← Kiểm tra 3: Giá trị kết thúc phải là INTEGER

    eat(KW_DO);
    compileStatement();
}
```

Kết luận

Vòng lặp FOR trong ngôn ngữ KPL bắt buộc:

- Biến điều khiển phải là kiểu INTEGER
- Giá trị bắt đầu phải là kiểu INTEGER
- Giá trị kết thúc phải là kiểu INTEGER

=> Không thể sử dụng biến ký tự CHAR để điều khiển vòng lặp FOR.

IV. Thay đổi trong các ví dụ đã cho để gây các lỗi:

1. Không tương ứng kiểu khi khai báo hằng

```

PROGRAM TestErrorConst;
CONST
  MAX = 100;
  LETTER = 'A';
  CH = +LETTER;
BEGIN
END.

```

The screenshot shows a code editor interface with several tabs open. The active tab is 'parser.c'. Below it, the 'TESTS' tab is selected, showing a list of test files: 'example1.kpl', 'example2.kpl', 'example3.kpl', 'example4.kpl', 'example5.kpl', 'example6.kpl', 'example12.kpl', 'Example12.txt', 'test_error_array.kpl', 'test_error_assign.kpl', 'test_error_const.kpl', 'test_error_expr.kpl', 'test_error_while.kpl', 'test_for_char.kpl', and 'Bai4_Tuong_ung_kieu.pdf'. The 'test_error_const.kpl' file is currently selected. In the bottom right corner, there is a terminal window showing the output of the test run, which includes the message '5-9:Type inconsistency'.

Code test:

CH = +LETTER; (* LETTER là hằng ký tự *)

Kết quả: 5-9:Type inconsistency

TAI SAO BỊ LỖI:

Dấu + chỉ được phép đứng trước hằng số INTEGER, không được dùng với CHAR.

LETTER có kiểu CHAR nên không thể có dấu +.

HÀM XỬ LÝ:

- compileConstant() (parser.c, dòng 267-268)

- Gặp dấu +, gọi compileConstant2() lấy giá trị
- Kiểm tra: if (constValue->type != TP_INT)
- Kết quả: constValue->type = TP_CHAR ≠ TP_INT → Báo lỗi

- error() (error.c)

- In ra: "Type inconsistency"
- Dừng chương trình

2. Dùng sai số chiêu của mảng so với khai báo

```

PROGRAM TestErrorArray;
VAR A : ARRAY(. 10 .) OF INTEGER;
x : INTEGER;
BEGIN
x := A;
END.

```

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows files like example2.kpl, parser.c, test_error_assign.kpl, test_error_const.kpl, test_error_array.kpl, and several test files (example1.kpl, example2.kpl, etc.).
- EDITOR:** Displays the K&R C code above.
- TERMINAL:** Shows command-line output from running the parser on test_error_array.kpl, indicating two type inconsistency errors.
- STATUS BAR:** Shows file paths, line numbers, and other development tools.

Code test:

```

VAR A : ARRAY(. 10 .) OF INTEGER;

x : INTEGER;

x := A; (* Gán toàn bộ mảng cho biến integer *)

```

Kết quả: 5-8:Type inconsistency

TẠI SAO BỊ LỖI:

Không thể gán mảng cho biến integer. Phải dùng chỉ số: $x := A(. 1 .)$

HÀM XỬ LÝ:

- compileAssignSt() (parser.c, dòng 489-498)
 - Lấy kiểu vẽ trái: $x \rightarrow TP_INT$
 - Lấy kiểu vẽ phải: $A \rightarrow TP_ARRAY$
 - Gọi checkTypeEquality(TP_INT, TP_ARRAY)

- checkTypeEquality() (semantics.c, dòng 149-152)

- Gọi compareType(type1, type2)
- compareType() so sánh TP_INT vs TP_ARRAY → trả về 0 (không bằng)
- Báo lỗi ERR_TYPE_INCONSISTENCY

3. Một biểu thức bắt đầu bằng + nhưng phần sau đó lại có kiểu ký tự

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows files like parser.c, test_error_assign.kpl, test_error_array.kpl, test_error_const.kpl, test_error_expr.kpl, test_error_while.kpl, and example1-6.kpl.
- TYPE_CHECKING:** Shows tests for various programs and examples.
- EDITOR:** Displays the following K&R C code:

```
PROGRAM TestErrorExpr;
1 VAR x : INTEGER;
2 ch : CHAR;
3 BEGIN
4 ch := 'A';
5 x := +ch;
6 END.
```
- PROBLEMS:** Lists several errors from the terminal output:
 - duongcongthuyet@Duongs-MacBook-Pro Type_checking % ./parser tests/test_error_const.kpl
 - 5-9:Type inconsistency
 - 5-8:Type inconsistency
 - 6-9:Type inconsistency
 - duongcongthuyet@Duongs-MacBook-Pro Type_checking % ./parser tests/test_error_array.kpl
 - duongcongthuyet@Duongs-MacBook-Pro Type_checking % ./parser tests/test_error_expr.kpl
 - duongcongthuyet@Duongs-MacBook-Pro Type_checking %

Code test:

VAR ch : CHAR;

$x := +ch;$ (* Dấu + với biến ký tự *)

Kết quả: 6-9:Type inconsistency

TAI SAO BỊ LỖI:

Toán tử đơn ngôi + chỉ áp dụng cho kiểu INTEGER, không được dùng với CHAR.

HÀM XỬ LÝ:

- compileExpression() (parser.c, dòng 677-693)

- Gặp dấu +, gọi compileExpression2()
- Nhận được kiểu của ch → TP_CHAR
- Gọi checkIntType(TP_CHAR)

- checkIntType() (semantics.c, dòng 116-120)

- Kiểm tra: if (type->typeClass == TP_INT)
- Kết quả: TP_CHAR ≠ TP_INT → Báo lỗi ERR_TYPE_INCONSISTENCY

4. Không tương ứng kiểu trong lệnh gán, lệnh while

```
PROGRAM TestErrorWhile;
VAR x : INTEGER;
    ch : CHAR;
BEGIN
    x := 10;
    ch := 'A';
    WHILE x < ch DO
        x := x + 1;
END.
```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS GITLENS POSTMAN CONSOLE DEVDB

- duongcongthuyet@Duongs-MacBook-Pro Type_checking % ./parser tests/test_error_const.kpl
- 5-9:Type inconsistency
- duongcongthuyet@Duongs-MacBook-Pro Type_checking % ./parser tests/test_error_array.kpl
- 5-8:Type inconsistency
- duongcongthuyet@Duongs-MacBook-Pro Type_checking % ./parser tests/test_error_expr.kpl
- 6-9:Type inconsistency
- duongcongthuyet@Duongs-MacBook-Pro Type_checking %

Code test:

```
VAR x : INTEGER;  
    ch : CHAR;  
WHILE x < ch DO (* So sánh INTEGER với CHAR *)
```

Kết quả: 7-13: Type inconsistency

TAI SAO BỊ LỖI:

Trong điều kiện so sánh, hai vế phải cùng kiểu. Không được so sánh INTEGER với CHAR.

HÀM XỬ LÝ:

- compileCondition() (parser.c, dòng 649-675)
 - Lấy kiểu vế trái: x → TP_INT
 - Kiểm tra là kiểu cơ bản → OK
 - Lấy kiểu vế phải: ch → TP_CHAR
 - Gọi checkTypeEquality(TP_INT, TP_CHAR)
- checkTypeEquality() (semantics.c, dòng 149-152)
 - compareType(TP_INT, TP_CHAR) → trả về 0
 - Báo lỗi ERR_TYPE_INCONSISTENCY