



File Edit Selection View Go Run ... Lab Assessment

EXPLORER ... Parser.cpp parser.cpp parser2.cpp X

LAB ASSESSMENT

OUTLINE

TIMELINE

```
26 enum TokenType
27 {
28     T_FLOAT,
29     T_WHILE,
30     T_RETURN,
31     T_ASSIGN,
32     T_LPAREN,
33     T_RPAREN,
34     T_LBRACE,
35     T_RBRACE,
36     T_STRING,
37     T_SEMICOLON,
38 };
39
40 string tokenTypeToString(TokenType type)
41 {
42     switch (type)
43     {
44     case T_IF:
45         return "T_IF";
46     case T_ID:
47         return "T_ID";
48     case T_EQ:
49         return "T_EQ";
50     case T_LE:
51         return "T_LE";
52     case T_GT:
53         return "T_GT";
54     case T_INT:
55         return "T_INT";
56     }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
History restored

PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>

Welcome to Visual Studio Code v1.94.0! Would you like to read the Release Notes?

Release Notes

Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32

11:13 PM 10/13/2024

File Edit Selection View Go Run ... Lab Assessment

EXPLORER ... Parser.cpp parser.cpp parser2.cpp X

LAB ASSESSMENT

OUTLINE

TIMELINE

```
62 case T_DIV:
63     return "T_DIV";
64 case T_NUM:
65     return "T_NUM";
66 case T_ELSE:
67     return "T_ELSE";
68 case T_PLUS:
69     return "T_PLUS";
70 case T_MINUS:
71     return "T_MINUS";
72 case T_FLOAT:
73     return "T_FLOAT";
74 case T_WHILE:
75     return "T_WHILE";
76 case T_RETURN:
77     return "T_RETURN";
78 case T_ASSIGN:
79     return "T_ASSIGN";
80 case T_LPAREN:
81     return "T_LPAREN";
82 case T_RPAREN:
83     return "T_RPAREN";
84 case T_LBRACE:
85     return "T_LBRACE";
86 case T_RBRACE:
87     return "T_RBRACE";
88 case T_STRING:
89     return "T_STRING";
90 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
History restored

PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>

Welcome to Visual Studio Code v1.94.0! Would you like to read the Release Notes?

Release Notes

Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32

11:13 PM 10/13/2024


```
File Edit Selection View Go Run ... Lab Assessment
EXPLORER
> LAB ASSESSMENT
> OUTLINE
> TIMELINE
Parser.cpp parser.cpp parser2.cpp x
E:\Study\Semester 7> Compiler Construction > week6 > parser2.cpp > Lex > tokenize()
class Lexer
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
...
vector<Token> tokenize()
{
    vector<Token> tokens;
    while (pos < src.size())
    {
        char current = src[pos];
        if (isspace(current))
        {
            pos++;
            continue;
        }
        else if (isdigit(current))
        {
            tokens.push_back(Token(T_NUM, consumeNumber()));
            continue;
        }
        else if (isalpha(current))
        {
            string word = consumeWord();
            if (word == "int")
            {
                tokens.push_back(Token(TokenType::T_INT, word));
            }
            else if (word == "float")
            {
                tokens.push_back(Token(TokenType::T_FLOAT, word));
            }
            else if (word == "if")
            {
                tokens.push_back(Token(TokenType::T_IF, word));
            }
            else if (word == "else")
            {
                tokens.push_back(Token(TokenType::T_ELSE, word));
            }
            else if (word == "return")
            {
                tokens.push_back(Token(TokenType::T_RETURN, word));
            }
            else
            {
                tokens.push_back(Token(TokenType::T_ID, word));
            }
            continue;
        }

        // Handle symbols and operators
        switch (current)
        {
            ...

```

```
File Edit Selection View Go Run ... Lab Assessment
EXPLORER
> LAB ASSESSMENT
> OUTLINE
> TIMELINE
Parser.cpp parser.cpp parser2.cpp x
E:\Study\Semester 7> Compiler Construction > week6 > parser2.cpp > Lex > tokenize()
class Lexer
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
...
vector<Token> tokenize()
{
    continue;
}

// Handle symbols and operators
switch (current)
{
    case '+':
        tokens.push_back(Token(TokenType::T_PLUS, "+"));
        break;
    case '-':
        tokens.push_back(Token(TokenType::T_MINUS, "-"));
        break;
    case '*':
        tokens.push_back(Token(TokenType::T_MUL, "*"));
        break;
    case '/':
        tokens.push_back(Token(TokenType::T_DIV, "/"));
        break;
    case '=':
        if (pos + 1 < src.size() && src[pos + 1] == '=')
        {
            tokens.push_back(Token(TokenType::T_EQ, "=="));
            pos++;
        }
        else
        {
            tokens.push_back(Token(TokenType::T_ASSIGN, "="));
        }
        break;
    case '<':
        if (pos + 1 < src.size() && src[pos + 1] == '=')
        {
            tokens.push_back(Token(TokenType::T_LE, "<="));
            pos++;
        }
        else
        {

```

The screenshot shows a C++ IDE with the following components:

- Menu Bar:** File, Edit, Selection, View, Go, Run, ...
- Toolbar:** Includes icons for file operations, search, and execution.
- Explorer Panel:** Shows the project structure with files like Parser.cpp, parser.cpp, and parser2.cpp.
- Outline Panel:** Displays the class structure, including the Lexer class and the tokenize function.
- Timeline Panel:** Shows the execution flow of the program.
- Main Editor:** Contains the implementation of the tokenize function in parser2.cpp. The function uses a vector to store tokens and a switch statement to handle different characters. The code is as follows:


```

class Lexer
{
public:
    vector<Token> tokenize()
    {
        int pos = 0;
        while (pos < current.length())
        {
            switch (current[pos])
            {
                case ' ':
                    tokens.push_back(Token(TokenType::T_LE, " "));
                    pos++;
                    break;
                case '<':
                    tokens.push_back(Token(TokenType::T_LT, "<"));
                    break;
                case '>':
                    tokens.push_back(Token(TokenType::T_GT, ">"));
                    break;
                case '(':
                    tokens.push_back(Token(TokenType::T_LPAREN, "("));
                    break;
                case ')':
                    tokens.push_back(Token(TokenType::T_RPAREN, ")"));
                    break;
                case '{':
                    tokens.push_back(Token(TokenType::T_LBRACE, "{"));
                    break;
                case '}':
                    tokens.push_back(Token(TokenType::T_RBRACE, "}"));
                    break;
                case ';':
                    tokens.push_back(Token(TokenType::T_SEMICOLON, ";"));
                    break;
                default:
                    cout << "Unexpected character: " << current[pos] << endl;
                    break;
            }
            pos++;
        }
        tokens.push_back(Token(TokenType::T_EOF, "EOF"));
        return tokens;
    }
};
      
```
- Windows Taskbar:** Shows the system clock as 11:14 PM on 10/13/2024, along with various application icons.

```
File Edit Selection View Go Run ... < -> Lab Assessment
EXPLORER ... Parser.cpp parser.cpp parser2.cpp x
> LAB ASSESSMENT E > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenizet()
103 class Lexer
132 vector<Token> tokenize()
133 {
134     int pos = 0;
135     while (pos < tokens.size())
136     {
137         pos++;
138     }
139     tokens.push_back(Token(TokenType::T_EOF, "EOF"));
140     return tokens;
141 }
142
143 class Parser
144 {
145 private:
146     vector<Token> tokens;
147     size_t pos;
148
149 public:
150     Parser(vector<Token> &tokens)
151     {
152         this->tokens = tokens;
153         this->pos = 0;
154     }
155     void parseStatement()
156     {
157         if (tokens[pos].type == T_INT)
158         {
159             parseDeclaration();
160         }
161         else if (tokens[pos].type == T_ID)
162         {
163             parseAssignment();
164         }
165         else if (tokens[pos].type == T_IF)
166         {
167             parseIfStatement();
168         }
169         else if (tokens[pos].type == T_RETURN)
170         {
171             parseReturnStatement();
172         }
173     }
174 }
```

```
File Edit Selection View Go Run ... Lab Assessment
EXPLORER
> LAB ASSESSMENT
> OUTLINE
> TIMELINE
Parser.cpp parser.cpp parser2.cpp X
E:\Study\Semester 7> Compiler Construction > week6 > parser2.cpp > Lex > tokenize()
233 class Parser
245 void parseStatement()
258 {
259 }
260 else if (tokens[pos].type == T_RETURN)
261 {
262     parseReturnStatement();
263 }
264 else if (tokens[pos].type == T_LBRACE)
265 {
266     parseBlock();
267 }
268 else
269 {
270     cout << "Syntax error: unexpected token " << tokens[pos].value << endl;
271     exit(1);
272 }
273 void parseProgram()
274 {
275     while (tokens[pos].type != T_EOF)
276     {
277         parseStatement();
278     }
279     cout << "Parsing completed successfully! No Syntax Error" << endl;
280 }
281 void parseBlock()
282 {
283     expect(T_LBRACE);
284     while (tokens[pos].type != T_RBRACE && tokens[pos].type != T_EOF)
285     {
286         parseStatement();
287     }
288     expect(T_RBRACE);
289 }
290 void parseDeclaration()
291 {
292 }
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
File Edit Selection View Go Run ... Lab Assessment
EXPLORER
> LAB ASSESSMENT
> OUTLINE
> TIMELINE
Parser.cpp parser.cpp parser2.cpp X
E:\Study\Semester 7> Compiler Construction > week6 > parser2.cpp > Lex > tokenize()
233 class Parser
273 void parseProgram()
278 {
279 }
280 cout << "Parsing completed successfully! No Syntax Error" << endl;
281 }
282 void parseBlock()
283 {
284     expect(T_LBRACE);
285     while (tokens[pos].type != T_RBRACE && tokens[pos].type != T_EOF)
286     {
287         parseStatement();
288     }
289     expect(T_RBRACE);
290 }
291 void parseDeclaration()
292 {
293     expect(T_INT);
294     expect(T_ID);
295     expect(T_SEMICOLON);
296 }
297 void parseAssignment()
298 {
299     expect(T_ID);
300     expect(T_ASSIGN);
301     parseExpression();
302     expect(T_SEMICOLON);
303 }
304 void parseIfStatement()
305 {
306     expect(T_IF);
307     expect(T_LPAREN);
308     parseExpression();
309     expect(T_RPAREN);
310     parseStatement();
311 }
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
File Edit Selection View Go Run ... Lab Assessment
EXPLORER ... Parser.cpp parser.cpp parser2.cpp x
> LAB ASSESSMENT
> OUTLINE
> TIMELINE
E:\Study\Semester 7> Compiler Construction > week6 > parser2.cpp > Lex > tokenize()
233 class Parser
234 void parseDeclaration()
299 }
300
301 void parseAssignment()
302 {
303     expect(T_ID);
304     expect(T_ASSIGN);
305     parseExpression();
306     expect(T_SEMICOLON);
307 }
308
309 void parseIfStatement()
310 {
311     expect(T_IF);
312     expect(T_LPAREN);
313     parseExpression();
314     expect(T_RPAREN);
315     parseStatement();
316
317     if (tokens[pos].type == T_ELSE)
318     {
319         expect(T_ELSE);
320         parseStatement();
321     }
322 }
323
324 void parseReturnStatement()
325 {
326     expect(T_RETURN);
327     parseExpression();
328     expect(T_SEMICOLON);
329 }
330
331 void parseExpression()
332 {
333     parseTerm();
334     while (tokens[pos].type == T_PLUS || tokens[pos].type == T_MINUS)
335     {
336         pos++;
```

```
File Edit Selection View Go Run ... Lab Assessment
EXPLORER ... Parser.cpp parser.cpp parser2.cpp x
> LAB ASSESSMENT
> OUTLINE
> TIMELINE
E:\Study\Semester 7> Compiler Construction > week6 > parser2.cpp > Lex > tokenize()
323 class Parser
324 void parseReturnStatement()
325 {
326     expect(T_RETURN);
327     parseExpression();
328     expect(T_SEMICOLON);
329 }
330
331 void parseExpression()
332 {
333     parseTerm();
334     while (tokens[pos].type == T_PLUS || tokens[pos].type == T_MINUS)
335     {
336         pos++;
337         parseTerm();
338     }
339
340     if (tokens[pos].type == T_GT)
341     {
342         pos++;
343         parseExpression();
344     }
345 }
346
347 void parseTerm()
348 {
349     parseFactor();
350     while (tokens[pos].type == T_MUL || tokens[pos].type == T_DIV)
351     {
352         pos++;
353         parseFactor();
354     }
355 }
356
357 void parseFactor()
358 {
359     if (tokens[pos].type == T_NUM || tokens[pos].type == T_ID)
360     {
361         pos++;
362     }
363     else if (tokens[pos].type == T_LPAREN)
364     {
```

```
File Edit Selection View Go Run ... Lab Assessment
EXPLORER
> LAB ASSESSMENT
> OUTLINE
> TIMELINE
Parser.cpp parser.cpp parser2.cpp X
E:\Study\Semester 7> Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
233 class Parser
247 void parseTerm()
252 {
253     pos++;
254     parseFactor();
255 }
256
257 void parseFactor()
258 {
259     if (tokens[pos].type == T_NUM || tokens[pos].type == T_ID)
260     {
261         pos++;
262     }
263     else if (tokens[pos].type == T_LPAREN)
264     {
265         expect(T_LPAREN);
266         parseExpression();
267         expect(T_RPAREN);
268     }
269     else
270     {
271         cout << "Syntax error: unexpected token " << tokens[pos].value << endl;
272         exit(1);
273     }
274 }
275
276 void expect(TokenType type)
277 {
278     if (tokens[pos].type == type)
279     {
280         pos++;
281     }
282     else
283     {
284         cout << "Syntax error: expected " << type << " but found " << tokens[pos].value << endl;
285         exit(1);
286     }
287 }
288
289 ;
```

```
File Edit Selection View Go Run ... Lab Assessment
EXPLORER
> LAB ASSESSMENT
> OUTLINE
> TIMELINE
Parser.cpp parser.cpp parser2.cpp X
E:\Study\Semester 7> Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
376 void expect(TokenType type)
384 {
385     cout << "Syntax error: expected " << type << " but found " << tokens[pos].value << endl;
386     exit(1);
387 }
388
389 ;
390
391 int main()
392 {
393     string sourceCode = R"(
394     int a;
395     int b;
396     b=a+10;
397     if(b>10){
398     return b;
399     }
400     else{
401     return 0;
402     }
403     )";
404     Lexer lexer(sourceCode);
405
406     vector<Token> tokens = lexer.tokenize();
407     Parser parser(tokens);
408     parser.parseProgram();
409     cout << tokens.size() << endl;
410     for (const Token &token : tokens)
411     {
412         cout << "Token Type: " << token.typeToString(token.type) << ", Value: " << token.value << endl;
413     }
414
415     return 0;
416 }
417
```