**Лексический анализатор** – первый из “слоев” компилятора, отвечающий за выделение лексем для последующей обработки. Лексема – минимальная единица некоего словаря, представляющего наш язык. В роли лексемы могут служить служебные слова, операторы, идентификаторы и так далее.

Виды токенов:

* Identifier - идентификатор
* Keyword - ключевое слово
* Int – целое число
* Float - число с плавающей точкой
* Binary – бинарное число
* Octal – восьмеричное число
* Hex – шестнадцатеричное число
* Char - символ
* String - строка
* Array - массив
* Addition - сложение
* Subtraction - вычитание
* Multiplication - умножение
* Division - деление
* Comparison - сравнение
* Assignment - присвоение
* Separator - разделитель
* Bracket - скобка
* LogicalOperator – логический оператор
* NegationOperator – оператор отрицания
* Error – ошибка

Лексемы

1) ключевые слова:

* main
* void
* int
* float
* char
* string
* bool
* if"
* else
* return
* while
* for
* true
* false

2) скобки:

* (
* )
* {
* }

3) разделители: ' ', ';', ',', ':', '\t', '\n', где ' ', '\t', '\n' игнорируются

Пример работы:

|  |  |
| --- | --- |
| int main(string[] args)  {  int array[10];  int bin = 0b0101010101;  int oct = 0o7456;  int f = 5 / 0xA4B5EF; //inline comment  if (f == 7)  {  f = -3;  }/\*multi  line  comment \*/  bool flag = false;  if (a != 4 || b <= 4 && true || !flag)  {  bool b = 3 <= 3;  }  float f = 0.5-0.5E;  char ch0 = 'tt'tt';  char ch = '';  char ch1 = '"';  char ch2 = '\\'';  char ch3 = '\\\r';  string s = "dhdh\"dhssss";  string s1 = "sss\"sssss'sdsdds\\";  print(ch);  } | 1 0 int Keyword  1 4 main Keyword  1 7 ( Bracket  1 9 string Keyword  1 14 [] Array  1 18 args Identifier  1 21 ) Bracket  2 0 { Bracket  3 2 int Keyword  3 6 array Identifier  3 10 [10] Array  3 14 ; Separator  4 2 int Keyword  4 6 bin Identifier  4 9 = Multiplication  4 12 0b0101010101 Binary  4 23 ; Separator  5 2 int Keyword  5 6 oct Identifier  5 9 = Multiplication  5 12 0o7456 Octal  5 17 ; Separator  7 2 int Keyword  7 6 f Identifier  7 7 = Multiplication  7 10 5 Int  7 11 / Division  7 14 0xA4B5EF Hex  7 21 ; Separator  7 23 / Division  8 2 if Keyword  8 4 ( Bracket  8 6 f Identifier  8 8 == Comparison  8 11 7 Int  8 11 ) Bracket  9 1 { Bracket  10 3 f Identifier  10 4 = Multiplication  10 6 - Subtraction  10 8 3 Int  10 8 ; Separator  11 1 } Bracket  11 2 / Division  15 2 bool Keyword  15 7 flag Identifier  15 11 = Multiplication  15 14 false Keyword  15 18 ; Separator  17 2 if Keyword  17 4 ( Bracket  17 6 a Identifier  17 8 != Comparison  17 11 4 Int  17 13 || LogicalOperator  17 16 b Identifier  17 18 <= Comparison  17 21 4 Int  17 23 && LogicalOperator  17 26 true Keyword  17 31 || LogicalOperator  17 33 ! NegationOperator  17 35 flag Identifier  17 38 ) Bracket  18 1 { Bracket  19 3 bool Keyword  19 8 b Identifier  19 9 = Multiplication  19 12 3 Int  19 14 <= Comparison  19 17 3 Int  19 17 ; Separator  20 1 } Bracket  22 2 float Keyword  22 8 f Identifier  22 9 = Multiplication  22 12 0.5 Float  22 14 - Subtraction  22 16 0.5E Error  22 19 ; Separator  24 2 char Keyword  24 7 ch0 Identifier  24 10 = Multiplication  24 12 'tt' Char  24 17 tt Identifier  24 18 '; Error  25 2 char Keyword  25 7 ch Identifier  25 9 = Multiplication  25 11 '' Error  25 13 ; Separator  26 2 char Keyword  26 7 ch1 Identifier  26 10 = Multiplication  26 12 '"' Char  26 15 ; Separator  27 2 char Keyword  27 7 ch2 Identifier  27 10 = Multiplication  27 12 '\\' Char  27 16 '; Error  28 2 char Keyword  28 7 ch3 Identifier  28 10 = Multiplication  28 12 '\\\r' Char  28 18 ; Separator  30 2 string Keyword  30 9 s Identifier  30 10 = Multiplication  30 12 "dhdh\"dhssss" String  30 26 ; Separator  31 2 string Keyword  31 9 s1 Identifier  31 11 = Multiplication  31 13 "sss\"sssss'sdsdds\\"; Error  33 2 print Identifier  33 6 ( Bracket  33 8 ch Identifier  33 9 ) Bracket  33 10 ; Separator  34 0 } Bracket |