HALSTEAD's Operators and Operands

/*
Indranil Nandy
MTech, CSE
Roll : 06CS6010
hi_i_am_indranil@yahoo.com
/*

Halstead's software science is an analytical technique to measure size, development effort, and development cost of software products. Halstead used two parameters operators and operands for this. Unfortunately, there is no general agreement among researchers on what is the most meaningful way to define the operators and operands for different programming languages.

However, a few general guidelines regarding identification of operators and operands for any programming language can be provided. Here is a suggestive list of operators and operands for the languages ANSI C, C++ and JAVA.

in C Language:

- **Operands :**

Tokens of the following categories are all counted as **operands** :

IDENTIFIER	all identifiers that are not reserved words
TYPENAME	(type specifiers) Reserved words that specify type: *int, float, char, double, long, short, signed, unsigned, void*. This class also includes some compiler specific nonstandard keywords.
CONSTANT	Character, numeric or string constants.

- **Operators :**

Tokens of the following categories are all counted as **operators** :

SCSPEC	(storage class specifiers) Reserved words that specify storage class: *auto, extern, register, static, typedef..*
TYPE_QUAL	(type qualifiers) Reserved words that qualify type: *const, final, volatile.*
RESERVED	Other reserved words of C: *break, case, continue, default, do, if, else, enum, for, goto, if, new, return, sizeof, struct, switch, union, while.* This class also includes some compiler specific nonstandard keywords.
OPERATOR	! != % %= & && || &= ( ) { } [ ] * *= + ++ += , - -- -= -> . ... / /= : :: < << <<= <= = == > >= >> >>= ? ^ ^= | |= ~ ; =& " " ' ' # ##

Some special cases are as follows :

- A pair of parenthesis is considered a single operator.
- The delimiter ; is considered a single operator.

The ternary operator '?' followed by ':' is considered a single operator as it is equivalent to "if-else" construct. x = (a == 3) ? a : b; is equivalent to:

```
if (a == 3)
        x = a;
else
```

x = b; .

- A label is considered an operator if it is used as the target of a GOTO statement.
- The following control structures *case ...: for (...), if (...), switch (...), while(...)* are treated in a special way. .
  The colon and the parentheses are considered to be a part of the constructs. The case and the colon or the "for (...)", "if (...)", "switch (...)", "while(...)" are counted together as one operator.
- The comments are considered neither an operator nor an operand.
- The function name is considered a single operator when it appears as calling a function ; but when it appears in declarations or in function definitions it is not counted as operator.
- Same is the case for the identifiers(or variables) and constants; when they appear in declaration they are not considered as operands, they are considered operands only when they appear with operators in expressions.
  As an example, func(a,b);-----here func, a and b are considered operands and ',' and ';' operators as it is calling a function, but for the following case we do not treat func, a and b as operands

```
int func(int a , int b) {
    …….
    …….
}
```

in C++ Language:

- **Operands :**

Tokens of the following categories are all counted as **operands** :

IDENTIFIER	all identifiers that are not reserved words
TYPENAME	(type specifiers) Reserved words that specify type: *bool , int, float, char, double, long, short, signed, unsigned,* void. This class also includes some compiler specific nonstandard keywords.
CONSTANT	Character, numeric or string constants.

- **Operators :**

Tokens of the following categories are all counted as **operators** :

SCSPEC	(storage class specifiers) Reserved words that specify storage class: *auto, extern, register, static, typedef, virtual, mutable, inline.*
TYPE_QUAL	(type qualifiers) Reserved words that qualify type: *const, friend, volatile, final.*
RESERVED	Other reserved words of C++: *break, case, continue, default, do, if, else, enum, for, goto, if, new, return, asm, operator, private, protected, public, sizeof, struct, switch, union, while, this, namespace, using, try, catch, throw, abstract, concrete, const_cast, static_cast, dynamic_cast, reinterpret_cast, typeid, template, explicit, true, false, typename*. This class also includes some compiler specific nonstandard keywords.
OPERATOR	! != % %= & && || &= ( ) { } [ ] * *= + ++ += , - -- -= -> . ... / /= : :: < << <<= <= = == > >= >> >>= ? ^ ^= | |= ~ ; =& " " ' ' # ## ~

Some special cases are as follows :

- A pair of parenthesis is considered a single operator.
- The delimiter ; is considered a single operator.

The ternary operator '?' followed by ':' is considered a single operator as it is equivalent to "if-else" construct. x = (a == 3) ? a : b; is equivalent to:

```
if (a == 3)
        x = a;
else
        x = b; .
```

- A label is considered an operator if it is used as the target of a GOTO statement.
- The following control structures *case ...: for (...) if (...) switch (...) while(...)* and try-*catch (...)* are treated in a special way .
  The colon and the parentheses are considered to be a part of the constructs. The case and the colon or the "for (...)", "if (...)", "switch (...)", "while(...)", "try-catch( )" are counted together as one operator.
- The comments are considered neither an operator nor an operand.
- The function name is considered a single operator when it appears as calling a function ; but when it appears in declarations or in function definitions it is not counted as operator.
- Same is the case for the identifiers(or variables) and constants; when they appear in declaration they are not considered as operands, they are considered operands only when they appear with operators in expressions.
  As an example, func(a,b);-----here func, a and b are considered operands and ',' and ';' operators as it is calling a function, but for the following case we do not treat func, a and b as operands

```
int func(int a , int b) {
    …….
    …….
}
```

- Default parameter assignments are not counted, e.g.,

```
class Point {
        Point(int x = 0,int y = 0);
};
```

  is not counted.
- **new** and **delete** considered same as the function calls, mainly because they are equivalent to the function calls.

In JAVA Language:

- **Operands :**

Tokens of the following categories are all counted as **operands** :

IDENTIFIER	all identifiers that are not reserved words
TYPENAME	(type specifiers) Reserved words that specify type: *bool , byte, int, float, char, double, long, short, signed, unsigned, void*. This class also includes some compiler specific nonstandard keywords.
CONSTANT	Character, numeric or string constants.

- **Operators :**

Tokens of the following categories are all counted as **operators** :

SCSPEC	(storage class specifiers) Reserved words that specify storage class: *auto, extern, register, static, typedef, virtual, mutable, inline.*
TYPE_QUAL	(type qualifiers) Reserved words that qualify type: *const, friend, volatile, transient, final.*
RESERVED	Other reserved words of JAVA: *break, case, continue, default, do, if, else, enum, for, goto, if, new, return, asm, operator, private, protected, public, sizeof, struct, switch, union, while, this, namespace, using, try, catch, throw, throws, finally, strictfp, instanceof, interface, extends, implements, abstract, concrete, const_cast, static_cast, dynamic_cast, reinterpret_cast, typeid, template, explicit, true, false, typename*. This class also includes some compiler specific nonstandard keywords.
OPERATOR	! != % %= & && || &= ( ) { } [ ] * *= + ++ += , - -- -= -> . ... / /= : :: < << <<= <= = == > >= >> >>> >>= >>>= ? ^ ^= | |= ~ ; =& “ “ ‘ ‘ # ## ~

Some special cases are as follows :

- A pair of parenthesis is considered a single operator.
- The delimiter ; is considered a single operator.

The ternary operator ‘?’ followed by ‘:’ is considered a single operator as it is equivalent to “if-else” construct. x = (a == 3) ? a : b; is equivalent to:

```
if (a == 3)
        x = a;
else
        x = b; .
```

- A label is considered an operator if it is used as the target of a GOTO statement.
- The following control structures *case ...: for (...) if (...) switch (...) while(...)* and try-*catch (...)* are treated in a special way .
  The colon and the parentheses are considered to be a part of the constructs. The case and the colon or the “for (...)”, “if (...)”, “switch (...)”, “while(...)”, “try-catch( )” are counted together as one operator.
- The comments are considered neither an operator nor an operand.
- The function name is considered a single operator when it appears as calling a function ; but when it appears in declarations or in function definitions it is not counted as operator.
- Same is the case for the identifiers(or variables) and constants; when they appear in declaration they are not considered as operands, they are considered operands only when they appear with operators in expressions.
  As an example, func(a,b);-----here func, a and b are considered operands and ‘,’ and ‘;’ operators as it is calling a function, but for the following case we do not treat func, a and b as operands

```
int func(int a , int b) {
    …….
    …….
}
```

- Default parameter assignments are not counted, e.g.,

```
class Point {
        Point(int x = 0,int y = 0);
};
```

  is not counted.

- **new** and **delete** considered same as the function calls, mainly because they are equivalent to the function calls.
- There are two extra operators in JAVA : >>> and >>>+ -------each is considered a single operator.

Some more general cases for C, C++ and JAVA :

- In C and C++, regarding to the headers, '#', '<>' and 'include' are considered operators while header files are considered operands. Similarly, in JAVA, 'import' is an operator and in imported file name each operand is separated by a '.' operator.

  For example, let us consider "#include<stdio.h>"(in C/C++) and "import java.util.date"

  Here the operators are { #, include, <>, import, .} and the operands are { stdio.h, java, util, date}.

- In "printf" or "cout" statements each argument separated by a comma is considered an operand.

  For example, in the statement **printf("Hello World")** there is only one operand "Hello World", not two "Hello" and "World"; in the statement **scanf("%d",&x**) there are two operands "%d" and "&x".