CS3300 Compiler Design (July 2024)

Assignment 1

Due August 25 23:59 on moodle

Create a lexer and a parser using lex-yacc for a C-like language. The constructs supported are:

- Functions and function calls
  - main is compulsory, but we can check that during code generation, not in A1.
  - printf should get supported.
- Declarations: global and local
  - using static, const variables should result in error
- Types: int, char, float, arrays of these types (1D and 2D)
  - supporting arrays 3D or more is optional.
  - using other types such as double or unsigned or long should result in error.
  - Pointers are optional to be supported, but we need to support passing arrays as parameters to functions.
  - Address-of operator should result in an error.
- Expressions: (), +, -, *, / and ** where ** is the exponentiation operator
  - bitwise operators should result in an error.
  - Precedence: () > ** > * / > + -.
  - Associativity: ** is right-associative, others are left-associative.
  - Expression may contain constants and array index expressions.
- Assignments: lvalue = expression;
  - lvalue is a variable or an array location.
- Conditionals: if, if-else.
  - Conditions can use ==, !=, <, <=, >, >=.
  - These comparison operators are supported for primitive types int, char, float. Using these with arrays should result in error.
  - Supporting && and || is optional.
- Loops: while, for.
  - Using do-while should result in error.
  - Conditions are the same as in if statements.
  - Using break and continue is optional.
  - Comma operator should result in error.
- Return: return with and without argument.
  - Primitive data types can be returned.
- All other constructs should result in error: switch, struct, union, typedef, #define, etc.

Input: Program written in your C-like language
Output:

- No output if the program is valid.
- If the program is syntactically invalid, print only the first line number (in the source program) where the error occurred. You do not need to parse further.

Examples:

Valid Example Program 1.

```
int main()
{
    printf("Hello World!");
    return 0;
}
```

Valid Example Program 2.

```
int main(){
    int  switch ;
    int    union;


    switch=   10;

    union=  3;

    int     double;

    double= switch ** union;

    printf("Result is--> %d", double);

    return 0;
}
```

Valid Example Program 3.

```
int main()

{
    int dowhile;

    dowhile=100;

    int i;

    for(i=0; i<10; i=i+1)

    {
        dowhile=dowhile+1;
        printf("Bye Bye %d", i);

    }



    if(dowhile>111)
    {
        if(i<10)
        {
            printf("Failed");
        }
        else
        {
            printf("Success");
        }
    }

    return 0;

}
```

Invalid Example Program 1.

```
int main(int a)
{
    int b;

    b=72;



    int a;
    a=10;
    return 0;
```

Output: 10 // this is the first line where your compiler should get an error.

Invalid Example Program 2.

```c
int main()
{


    int j;
    j=0;

    for(; j<10; j=j+1){
        printf("Chennai is always hot");
    }

    double var=100;

    printf("this should not print: %ld", var);

    return 0;

}
```

Output: 11

Invalid Example Program 3.

```c
int main()
{
    int const;
    int default;

    const=0;
    default=8;

    do
    {
        default+1;
        const=const+1;
        default=default%const;
    }while(const<8);

    printf("%d ", default);
    return 0;
}
```

Output: 10