



**AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH**

**Faculty of Science and Technology**

**Report of**

**Compiler Design Section:**

**A**

**Spring 2023-2024**

**Supervised by**

**NAZMUS SAKIB SHAN**

**Report on**

**Program to Determine Whether the Given Input Is Identifier or Not**

**Submitted by:**

**Mim, Mahidul Islam**

**ID: 21-45620-3**

CODE:

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <regex>
```

```
#include <string>
```

```
#include <sstream>
```

```
#include <vector>
```

```
using namespace std;
```

```
bool matchPattern(string input, string pattern)
```

```
{
```

```
    regex regexPattern(pattern);
```

```
    if (regex_match(input, regexPattern))
```

```
    {
```

```
        return true;
```

```
    }
```

```
    else
```

```
    {
```

```
        return false;
```

```
    }
```

```
}
```

```
bool checkDigit(char c)
```

```
{
```

```
    if(c=='0' || c=='1' || c=='2' || c=='3' || c=='4' || c=='5' || c=='6' || c=='7' || c=='8' || c=='9')
```

```
    {
```

```
        return true;
```

```
    }
```

```
    else
```

```
        return false;
```

```
}
```

```
bool isNumericConstant(string s)
```

```
{
```

```
    if((s[0] == '-' || s[0] == '+' || s[0] == '.') && s.length() == 1)
```

```
    {
```

```

        return false;
    }
    else
    {
        for(int i= 0; i<s.length(); i++)
        {
            if(s[i] == '-' || s[i] == '+' || s[i] == '.' || s[i] == 'e' || s[i] == '^')
            {
                i++;
            }
            if(!checkDigit(s[i])) && (s[i] != '^' || s[i] != 'e')
            {
                return false;
            }
        }
        if(s[0] == '^' || s[0] == 'e')
        {
            return false;
        }
        return true;
    }
}

```

```

bool isDataType(string s)
{
    if(s == "int" || s=="double" || s=="long" || s=="bool" || s=="float" || s=="short" || s=="string" ||
s=="public" || s=="private" || s=="protected" || s=="static" || s=="virtual" || s=="const" || s=="void" ||
s=="signed" || s=="unsigned" || s=="return" || s=="char")
    {
        return true;
    }
    else
        return false;
}

```

```

bool isKeyWord(string s)
{

```

```

if(s == "if" || s=="else" || s=="do" || s=="while" || s=="for" || s=="cin" || s=="cout" || s=="const")
{
    return true;
}
else
    return false;
}

```

```

void showOperators(string s)
{
    int j =1;
    for(int i=0; i<s.length(); i++)
    {
        char c = s[i];
        if(c == '+' || c == '-' || c== '*' || c == '%' || c == '=')
        {
            cout<< "Operator " << j++<< ": " << c<< ", ";
        }
    }
}

```

```

bool isIdentifier(string s)
{
    int j =0;
    char c1 = s[0];
    int validity = 0;

    if(s.length()==0)
    {
        cout<< "String empty"<<endl;
        return false;
    }
    else
    {
        if(c1=='0' || c1=='1' || c1=='2' || c1=='3' || c1=='4' || c1=='5' || c1=='6' || c1=='7' || c1=='8' ||
c1=='9')
        {

```

```

        cout<< "Identifiers cannot have numbers at the beginning. "<<endl;
        validity++;
    }
    else
    {
        for(int i = 0; i<s.length(); i++)
        {
            char c = s[i];

            if(c=='#' || c=='<' || c=='>' || c=='?' || c=='-' || c=='+' || c=='*' || c=='/' || c=='%' || c=='$' ||
c=='&' || c=='^' || c=='@' || c==',' || c=='.' || c=='(' || c==')' || c=='"' || c==';' || c==':' || c=='|' || c=='='
|| c=='!' || c=='\'' || c=='[' || c==']' || c=='{' || c=='}' || c=='\\')
            {
                cout<< "Error at index '" << i<<"". Identifier cannot contain special character. "<<c<<endl;
                validity++;
            }
            if(c == ' ')
            {
                cout<< "Error at index '" << i<<"". Identifier cannot contain empty spaces. "<<endl;
                validity++;
            }

            if(s == "int" || s=="double" || s=="long" || s=="bool" || s=="float" || s=="short" || s=="string"
|| s=="if" || s=="else" || s=="asm" || s=="new" || s=="switch" || s=="case" || s=="auto" ||
s=="operator" || s=="template" || s=="break" || s=="enum" || s=="public" || s=="private" || s=="this"
|| s=="protected" || s=="for" || s=="do" || s=="while" || s=="static" || s=="try" || s=="catch" ||
s=="throw" || s=="sizeof" || s=="virtual" || s=="const" || s=="void" || s=="signed" || s=="default" ||
s=="continue" || s=="goto" || s=="unsigned" || s=="return" || s=="char" || s=="extern" || s=="enum"
|| s=="struct" || s=="friend" || s=="inline" || s=="volatile" || s=="class" || s=="register" || s=="typedef"
|| s=="union")
            {
                cout<< "Identifier cannot be a keyword. "<<endl;
                validity++;
                break;
            }
        }
    }
}
}

```

```
if(Validity == 0)
{
    return true;
}
else
    return false;
}
```

```
bool isSingleLine(string s)
{
    for(int i=0; i<s.length(); i++)
    {
        char c = s[i];
        if(c == '/')
        {
            if(s[i+1] == '/')
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

```
bool isMultiLine(string s)
{
    for(int i=0; i<s.length(); i++)
    {
        char c = s[i];
        if(c == '/')
        {
            if(s[i+1] == '*')
            {
                return true;
            }
        }
    }
}
```

```

    }
    else
    {
        return false;
    }
}
}

```

```

bool isComplete(string s)
{
    for(int i=0; i<s.length(); i++)
    {
        char c = s[i];
        if(s[i-1] != '/' && c == '*')
        {
            if(s[i+1] == '/')
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}

```

```

void commentCheck(string s)
{
    if(isSingleLine(s))
    {
        cout<< "Single line comment. "<<endl<<endl;
    }
    else if(isMultiLine(s))
    {
        if(isComplete(s))

```

```

    {
        cout<< "Multiline comment. "<<endl<<endl;
    }
    else
    {
        cout<< "Multiline comment without end. "<<endl<<endl;
    }
}
else
{
    if(matchPattern(s, "#include<+[A-Za-z]+>") || matchPattern(s, "using namespace +[A-Za-z]+;") ||
    !isDataType(s) || isDataType(s) || isIdentifier(s) || !isIdentifier(s))
    {

    }
    else
    {
        cout<< "Invalid comment. "<<endl<<endl;
    }
}
}

```

```

bool isHeader(string s)

```

```

{
    if(matchPattern(s, "#include<+[A-Za-z]+>") || matchPattern(s, "#include<+[A-Za-z]+>\\s*$"))
    {
        return true;
    }
    else
    {
        return false;
    }
}

```

```

bool isNamespace(string s)

```

```

{
    if(matchPattern(s, "using namespace +[A-Za-z]+;"))

```



```

{
    return true;
}
else
{
    return false;
}
}

```

```
bool isMethod(string s)
```

```

{
    string s1, s2, s3;
    stringstream ss(s);
    ss >> s1;

    if(isDataType(s1))
    {
        if(matchPattern(s, "\\b(int|void|float|double|string|char)\\s+\\w+\\s*\\((.*?\\)\\s*\\{\\}")
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}

};

```

```
bool isStatement(string s)
```

```

{
    string s1, s2, s3;
    stringstream ss(s);
    ss >> s1;

    if(isKeyWord(s1))
    {

```

```

        if(matchPattern(s, "\\s*\\|if\\s*\\|\\. *?\\|\\s*\\|{")
        {
            return true;
        }
        else
        {
            return false;
        }
    }

};

```

```

bool isEnd(string s)
{
    if(matchPattern(s, ".+?;\\s*$"))
    {
        return true;
    }
    else
        return false;
}

```

```

int main ()
{
    string line;
    string s;
    string s2;
    string s3;
    string s4;
    string s5;
    ifstream MyReadFile("lex_input.txt");

    while (getline(MyReadFile, line))
    {
        stringstream ss(line);
        ss >> s;
        ss >> s2;

```

```

ss >> s3;
ss >> s4;
ss >> s5;
cout << line << " ";
if(isHeader(line))
{
    cout<< "Header. ";
}
if(isNamespace(line))
{
    cout<< "Namespace. ";
}

showOperators(line);

if(!isHeader(line) && !isNamespace(line) && line != "" && !isMethod(line) && !isKeyWord(s) && s !=
"return")
{
    if(isKeyWord(s))
    {
        if(isDataType(s2))
        {
            isIdentifier(s3);
        }
        else
        {
            isIdentifier(s2);
        }
    }
    else if(isDataType(s))
    {
        isIdentifier(s2);
    }
    else
    {
        if(!matchPattern(line, "\\s*\\}\\s*$"))
        {

```

```

        cout<< "Invalid Datatype. ";
    }
}

commentCheck(line);
if(line != "" && !isHeader(line) && line != "}" && !isMethod(line) && !isStatement(line))
{
    if(!isEnd(line) && !matchPattern(line, "\\s*\\}\\s*$"))
    {
        cout<< "Expected ; at the end. ";
    }
}
cout<<endl;

}
MyReadFile.close();
cout << "// Code developed and designed by Mim, Mahidul Islam" << endl;
cout << "// ID: 21-45620-3" << endl;
cout << "// Sec: A" << endl;
cout << "// Course Name: Compiler Design" << endl;
cout << "// Instructor: NAZMUS SAKIB SHAN" << endl;

}

```

Output:

```
"C:\Users\ASUS\Documents\C" × + v
#include<iostream>;
using namespace std; Namespace.

int main() {
    cout << "Welcome";
    int x = 24 % 10; Operator 1: =, Operator 2: %,
    if (x == 4) { Operator 1: =, Operator 2: =,
        x = 40; Operator 1: =, Invalid Datatype.
    }
    itn y = 50; Operator 1: =, Invalid Datatype.
    int #z = 60; Operator 1: =, Error at index '0'. Identifier cannot contain special character. #

    Error at index '0'. Identifier cannot contain special character. #
    Expected ; at the end.
    return 0;
}
// Code developed and designed by Mim, Mahidul Islam
// ID: 21-45620-3
// Sec: A
// Course Name: Compiler Design
// Instructor: NAZMUS SAKIB SHAN

Process returned 0 (0x0) execution time : 0.119 s
Press any key to continue.
|
```

Link: [MahidulMim/Compiler-Design \(github.com\)](https://github.com/MahidulMim/Compiler-Design)