```
#include <fstream>
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>


using namespace std;

bool isPunctuator(char ch)
{
   if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
      ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
      ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
      ch == '[' || ch == ']' || ch == '{' || ch == '}' ||
      ch == '&' || ch == '|')
      {
         return true;
      }
   return false;
}

bool validIdentifier(char* str)
{
   if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||
      str[0] == '3' || str[0] == '4' || str[0] == '5' ||
      str[0] == '6' || str[0] == '7' || str[0] == '8' ||
      str[0] == '9' || isPunctuator(str[0]) == true)
      {
```

```
        return false;
      }
   int i,len = strlen(str);
   if (len == 1)
   {
      return true;
   }
   else
   {
   for (i = 1 ; i < len ; i++)
   {
      if (isPunctuator(str[i]) == true)
      {
         return false;
      }
   }
   }
   return true;
}

bool isOperator(char ch)
{
   if (ch == '+' || ch == '-' || ch == '*' ||
      ch == '/' || ch == '>' || ch == '<' ||
      ch == '=' || ch == '|' || ch == '&')
   {
     return true;
   }
   return false;
```

```
}

bool isKeyword(char *str)
{
   if (!strcmp(str, "if") || !strcmp(str, "else") ||
      !strcmp(str, "while") || !strcmp(str, "do") ||
      !strcmp(str, "break") ||  !strcmp(str, "continue")
      || !strcmp(str, "int") || !strcmp(str, "double")
      || !strcmp(str, "float") || !strcmp(str, "return")
      || !strcmp(str, "char") || !strcmp(str, "case")
      || !strcmp(str, "long") || !strcmp(str, "short")
      || !strcmp(str, "typedef") || !strcmp(str, "switch")
      || !strcmp(str, "unsigned") || !strcmp(str, "void")
      || !strcmp(str, "static") || !strcmp(str, "struct")
      || !strcmp(str, "sizeof") || !strcmp(str,"long")
      || !strcmp(str, "volatile") || !strcmp(str, "typedef")
      || !strcmp(str, "enum") || !strcmp(str, "const")
      || !strcmp(str, "union") || !strcmp(str, "extern")
      || !strcmp(str,"bool"))
      {
         return true;
      }
   else
   {
     return false;
   }
}

bool isNumber(char* str)
```

```
{
    int i, len = strlen(str),numOfDecimal = 0;
    if (len == 0)
    {
        return false;
    }
    for (i = 0 ; i < len ; i++)
    {
        if (numOfDecimal > 1 && str[i] == '.')
        {
            return false;
        } else if (numOfDecimal <= 1)
        {
            numOfDecimal++;
        }
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] != '5'
            && str[i] != '6' && str[i] != '7' && str[i] != '8'
            && str[i] != '9' || (str[i] == '-' && i > 0))
            {
                return false;
            }
    }
    return true;
}

char* subString(char* realStr, int l, int r)
{
    int i;
```

```
    char* str = (char*) malloc(sizeof(char) * (r - l + 2));

    for (i = l; i <= r; i++)
    {
        str[i - l] = realStr[i];
        str[r - l + 1] = '\0';
    }
    return str;
}


void parse(char* str)
{
    int left = 0, right = 0;
    int len = strlen(str);
    while (right <= len && left <= right) {
        if (isPunctuator(str[right]) == false)
            {
                right++;
            }

        if (isPunctuator(str[right]) == true && left == right)
            {
            if (isOperator(str[right]) == true)
            {
                std::cout<< str[right] <<" IS AN OPERATOR\n";
            }
            right++;
```

```
        left = right;
        } else if (isPunctuator(str[right]) == true && left != right
              || (right == len && left != right))
        {
        char* sub = subString(str, left, right - 1);

        if (isKeyword(sub) == true)
                {
                   cout<< sub <<" IS A KEYWORD\n";
                }
        else if (isNumber(sub) == true)
                {
                   cout<< sub <<" IS A NUMBER\n";
                }
        else if (validIdentifier(sub) == true
               && isPunctuator(str[right - 1]) == false)
               {
                  cout<< sub <<" IS A VALID IDENTIFIER\n";
               }
        else if (validIdentifier(sub) == false
               && isPunctuator(str[right - 1]) == false)
               {
                  cout<< sub <<" IS NOT A VALID IDENTIFIER\n";
               }

        left = right;
        }
}
return;
```

```
}

int main()
{
   string myText;


 ifstream MyReadFile("filename.txt");

 while (getline (MyReadFile, myText)) {

   cout << myText<<endl;


 }


 MyReadFile.close();
     char* char_arr = &myText[0];

   char c[100] = &char_arr;
   parse(c);
   return 0;
}
```

Explanation:

This C++ program appears to be designed to analyze and classify different elements in a C or C++ source code file, such as keywords, operators, numbers, and identifiers. It reads the content of a file named "filename.txt" and processes each line using various functions.

Let's break down the code and discuss its components:

Header Files

```
include <stream>  include <iostream>  include <stdlib.h>  include <string.h>  include <ctype.h>
```

These are the standard C++ header files for file input/output (**`<fstream>`**), input/output through the console (**`<iostream>`**), dynamic memory allocation (**`<stdlib.h>`**), string manipulation (**`<string.h>`**), and character handling (**`<ctype.h>`**).

```
using  namespace
```

This line declares that entities in the code belong to the **`std`** namespace.

Function Declarat

```
bool isPunctuator  char  bool validIdentifier  char  bool isOperator  char  bool isKeyword  char
 bool isNumber  char  char  subString  char  int  int  void parse  char
```

These are the function prototypes. They define the functions that will be implemented later in the code.

Utility Function

```
bool isPunctuator  char
```

This function checks if a given character is a punctuation symbol.

```
bool validIdentifier  char
```

This function checks if a given string is a valid identifier.

```
bool isOperator  char
```

This function checks if a given character is an operator.

```
bool isKeyword  char
```

This function checks if a given string is a keyword in C/C++

```
bool isNumber  char
```

This function checks if a given string represents a valid number.

```
char   subString  char              int    int
```

This function extracts a substring from a given string.

Parsing Function

```
void parse  char
```

This function parses a given string, identifying and printing keywords, operators, numbers, and valid identifiers.

Main Function

```
int main
```

The **`main`** function reads the content of the "filename.txt" file line by line using an **`ifstream`**. For each line, it prints the line to the console. Then, it converts the line to a C-style string (**`char*`**) and calls the **`parse`** function to analyze and classify the elements in the line.

File Handling

```
          MyReadFile  "filename.txt"   while   getline
                close
```

This block of code opens and reads the content of the "filename.txt" file line by line using an **`ifstream`**, storing each line in the **`myText`** variable. The lines are then printed to the console. Finally, the file is closed.

Conversion and Parsing

char 0 char 100 parse

These lines convert the string **`myText`** to a C-style string (**`char*`**) named **`char_arr`**. However, there's an issue in the next line: **`char c[100] = &char_arr;`**. It attempts to assign a pointer to an array, which is incorrect. It should be **`char* c = char_arr;`**. After the correction, the **`parse`** function is called with the C-style string as an argument.

Parsing Output

The **`parse`** function processes each character in the input string, identifying and printing keywords, operators, numbers, and valid identifiers.

Token Classification Functions:

isPunctuator(char ch): Checks if a character is a punctuator (e.g., space, arithmetic operators, parentheses, etc.).

validIdentifier(char* str): Determines if a given string is a valid identifier by checking its first character and subsequent characters for punctuators.

isOperator(char ch): Identifies whether a character is an operator.

isKeyword(char* str): Checks if a given string matches known C++ keywords.

Lexical Analysis:

isNumber(char* str): Identifies if a string is a valid number.

subString(char* realStr, int l, int r): Extracts a substring from a given string.

Summary

In summary, the code reads a C/C++ source code file, processes each line, and identifies and classifies keywords, operators, numbers, and valid identifiers using various functions. However, there is a mistake in the conversion of the string to a C-style string in the **`main`** function, and it should be corrected for the program to work as intended.