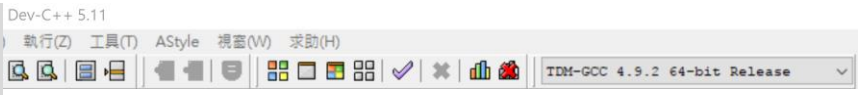
106-1 Data Structure
Assignment Program 2

104403512 吳宛儒

編譯環境



使用語言 C++

解題步驟

定義 MAXLEN 為 500

宣告 node 結構

```
 7  typedef struct node * nodePointer;
 8  struct node{
 9      char data;
10      nodePointer link;
11  };
```

```
13  nodePointer top;
```
設定初始 top 指向 stack 底端

設定 push 和 pop 函數

```
15  void push(char d)
16  {
17      nodePointer newNode = new node();
18      newNode->data = d;
19      newNode->link = top;
20      top = newNode;
21      return;
22  }
23
24  char pop()
25  {
26      nodePointer temp = top;
27      if(!temp)
28          return 'N';
29      top = top->link;
30      char item = temp->data;
31      free(temp);
32      return item;
33  }
```

設定清空 stack 的函數

(因為只能在外面宣告 top，不清楚為何)

```
35  void clear()
36  {
37      char x = pop();
38      while(x!='N')
39      {
40          x = pop();
41      }
42  }
```

讀入檔案

int i 存目前獨到的 line 陣列的 index

char array line 存入一行中的每個字元

last 最後用來存最後一個字元

```
int main()
{
    fstream f1,f2;
    f1.open("input.txt",ios::in);
    f2.open("output.txt", ios::out | ios::trunc);
    int i;
    char line[MAXLEN],last;
```

開始判斷 infix 是否 valid

```
    // Read each line one time
    while(f1.getline(line,sizeof(line),'\n'))
    {
        // Read each character one time
        for(i=0;i<sizeof(line);i++)
        {
            if(line[i]=='(')
            {
                push('(');
            }
            else if(line[i]==')')
            {
                char k = pop();
                if(k=='N')
                {
                    // If we meet the')' but the stack is empty
                    f2<<"The expression is invalid.\n";
                    break;
                }
            }
```

```
            else if(line[i]==line[MAXLEN-1])
            {
                // seem as the end of the line
                break;
            }
            // get the last character of the line
            last = line[i];

        }
```

如果 get 出來為 N(代表 stack 為空)或是 last 不等於任何一個運算符號

代表 infix 為 valid 可以做繼續中序轉後序式的操作

注意要將 pop() 出來判斷的東西先存入一個 char 變數，否則直接放入判斷式會 pop 很多次

並要將判斷後是不必 pop() 出來的 push() 回去

```
            //Read each line
            for(int n=0;n<sizeof(line);n++)
            {
                // token is the character now we meet
                char token = line[n];

                if(token == line[MAXLEN-1])
                    break;

                if(token == ')')
                {
                    char p = pop();
                    while (p != '(' && p !='N')
                    {
                        postfix[index++] = p;
                        p = pop();
                    }
                }
                else if(token == '(')
                {
                    // if we meet r is '(' , we have to push it back
                    push(token);
                }
```

```
114                 else if(token == '+' || token == '-')
115                 {
116                     char p = pop();
117                     while(p=='*' || p=='/' || p=='+' || p=='-')
118                     {
119                         postfix[index++] = p;
120                         p = pop();
121                     }
122                     if (p == '(')
123                     {
124                         // if we meet r is '(' , we have to push it back
125                         push(p);
126                     }
127                     push(token);
128                 }
129                 else if(token == '*' || token == '/')
130                 {
131                     char p = pop();
132                     while(p=='*' || p=='/')
133                     {
134                         postfix[index++] = p;
135                         p = pop();
136                     }
137                     if(p =='(' || p=='+' || p=='-')
138                     {
139                         // if we meet r is '('or'+'or'-' , we have to push it back
140                         push(p);
141                     }
142                     push(token);
143                 }
```

遇到 operand 數字直接印出(line 144)

最後要將剩下在 stack 內的東西印出(line 150)

再將 postfix 寫入 f2(output.txt)

```
144                 else
145                 {
146                     // if we meet operand
147                     postfix[index++] = token;
148                 }
149             }
150                 // clear the node in the stack
151                 nodePointer tmp = top;
152                 while(tmp!=NULL){
153                     postfix[index++] = tmp->data;
154                     tmp = tmp->link;
155                 }
156
157             f2 << postfix;
158             f2 << "\n";
159             clear();
```

Postfix 求值

```
161             //postfix to its value
162             for(int i=0;i<sizeof(postfix);i++){
163
164                 // token is the character we now meet
165                 char token = postfix[i];
166                 // declare op1,op2 to store the integer operand
167                 int op1, op2;
168
169                 if(token == postfix[MAXLEN-1])
170                     break;
```

如果遇到運算符號 要做計算

```
171                 // if we meet operator
172                 if( token =='+' || token =='-' || token =='*' || token =='/')
173                 {
174                     op2= pop()-'0';
175                     op1 = pop()-'0';
176 
177                     switch(token)
178                     {
179                         case '+':
180                             push ((op1+op2)+'0');
181                             break;
182                         case '-':
183                             push ((op1-op2)+'0');
184                             break;
185                         case '*':
186                             push ((op1*op2)+'0');
187                             break;
188                         case '/':
189                             float result = (float)op1/(float)op2 ;
190                             int num;
191                             if(op1%op2 != 0)
192                             {
193                                 num = result + 0.5;
194                             }
195                             else
196                             {
197                                 num = result;
198                             }
199                             push (num +'0');
200 
201                             break;
202                     }
203 
204                 }
```

```
205                 else
206                 {
207                     push(token);
208 
209                 }
210             }
211             //the rest in the stack is our result
212             // pop the result
213             f2 << pop()-'0';
214             f2 << "\n";
215 
216         }
```

最後是剛剛所排除掉的 如果 infix 判斷 valid 與否的堆疊非空或是 last 為運算符號則是 invalid

Line 221:每讀完一行清空 stack

將檔案關閉

```
217         else
218         {
219             f2<<"The expression is invalid.\n";
220         }
221         //clean the stack when the line is finished
222         clear();
223     }
224     //  Close two files
225     f1.close();
226     f2.close();
227 
228     system("pause");
229     return 0;
230 }
```

所遇問題

1. 讀取一行運算式中看起來為空的字元時(已結束)，卻讀到奇怪的字元，但後來莫名其妙好了。
2. 一開始未注意到判斷式每寫一個 pop() 會 pop() 一次，造成判斷錯誤，也未將不需要 pop() 出來的東西判斷後放回去。
3. 由於將 top 設為全域變數，必須在每判斷一行結束後以 clear() 清空資料。