

## DEPARTMENT OF COMPUTER ENGINEERING

Semester	T.E. Semester VI- SPCC
Subject	Software Engineering
Subject Professor In-charge	Prof. Pankaj Vanvari
Assisting Teachers	Prof. Pankaj Vanvari
Laboratory	M310B

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

Macro pre-processor

---

Approach:

1. Input File Reading (readfile function):

- The **readfile** function reads the contents of a given file (**fileName**) character by character and separates them into words based on spaces and newline characters.
- It stores the words in a vector called **input**, representing the input file's content.

2. Pass 1 (pass1 function):

- Pass 1 of the assembler involves scanning the input file and building necessary data structures like Macro Name Table (MNT), Macro Definition Table (MDT), Formal Parameter List (FPL), and Argument Parameter List (APL).
- The function scans through the input vector and identifies macro definitions (**MACRO** and **MEND** directives).
- For each macro definition, it extracts macro name, positional parameters, and the macro body, then stores this information in respective data structures (MNT, MDT, FPL).
- The MNT maps macro names to their respective positions in the MDT.
- The MDT stores the lines of the macro definition.

- The FPL maps macro names to their respective formal parameter lists.
- The APL, though not used in Pass 1, could be used in Pass 2 to handle actual arguments of macro calls.

### 3. Macro Expansion (Expand function):

- The **Expand** function performs macro expansion using the data structures built in Pass 1.
- It iterates through the input vector, identifies macro calls by checking the MNT, and expands them by substituting actual arguments for formal parameters.
- The expanded code is written to an output file named "macroexpanded.txt".

### 4. Challenges

- The most challenging part for me was the implementing the APL and FPL part not because it is difficult to implement, but because when we learning this in this part in the class I don't know why I thought that APL and FPL data bases were not adding anything extra to this architecture, so I started to implement it with that mind set, and it took me hours to understand that I was very wrong

---

### Implementation:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <map>
using namespace std;

int readfile(string &fileName,vector<string> &input){
    char ch;
    fstream fp;
    fp.open(fileName.c_str(), std::fstream::in);

    if (!fp)
    {
        cerr << "Error opening the file: " << fileName << endl;
        return 1; // Return an error code
    }

    string word;
    while (fp >> noskipws >> ch)
    {
```

```

        if (ch == '\n')
        {
            input.push_back(word);
            input.push_back(";");
            word = "";
        }
        else if (ch == ' ')
        {
            input.push_back(word);
            word = "";
        }
        else
        {
            word += ch;
        }
    }

    input.push_back(word);

    fp.close();
    return 0; // Return success code
}

void print_vector(vector<string> &input)
{
    for (int i = 0; i < input.size(); i++)
    {
        cout << input[i] << endl;
    }
    cout << endl;
}

void pass1( map<string,int>&MNT_MDTP,map<string,int>&MNT_PP,vector<string>&MDT,
map<string,vector<string> >&FPL,map<string,vector<string> >&APL,vector<string>in-
put,vector<string>&input_copy){

    //start going throught the inptus
    int inputsize=input.size();
    for(int i=0;i<inputsized;i++){

        if(input[i]=="") continue;
    }
}

```

```
if(input[i]=="MACRO"){

    //filling the MNT
    i++;
    i++;
    int MDNextindex=MDT.size();
    string line1="";
    string Macro_Name=input[i];
    line1=line1+Macro_Name+" ";
    MNT_MDT[Macro_Name]=MDNextindex;
    //counting number of positional parameter
    int ppc=0;
    i++;
    while(input[i]!=";"){
        int currindex=FPL[Macro_Name].size();
        //convert it to string
        string temp=to_string(currindex);

        line1=line1+temp+" ";
        //adding it to FPL and APL
        FPL[Macro_Name].push_back(input[i]);

        ppc++;
        i++;
    }
    MNT_PP[Macro_Name]=ppc;

    MDT.push_back(line1);

    string line="";
    i++;

    while(input[i!="MEND"]){

        if(input[i]==";"){
            i++;
            MDT.push_back(line+";");
            line="";
            continue;
        }

        //replacing the positional parameter
```

```

        if(input[i][0]=='&'){
            //find index[i] in the FPL[Macro_Name]

            //cout<<input[i]<<endl;
            int size=FPL[Macro_Name].size();
            for(int j=0;j<size;j++){
                //cout<<to_string(j)<<"test"<<endl;
                //cout<<input[i]<<endl;
                if(FPL[Macro_Name][j]==input[i]){
                    //cout<<to_string(j)<<"test"<<endl;
                    line=line+"#"+to_string(j)+" ";
                    break;
                }
            }

            }else{
                line=line+input[i]+" ";
            }

            i++;

        }

        MDT.push_back(input[i]);

    }

}

void Map_print(map<string, int> &m)
{
    for (std::map<std::string, int>::iterator it = m.begin(); it != m.end();
++it)
    {
        std::cout << it->first << " " << it->second << std::endl;
    }
}

```

```
void Map_print2(map<string, vector<string>> &m)
{
    for (auto it = m.begin(); it != m.end(); ++it)
    {
        std::cout << it->first << ": ";
        for (const auto &value : it->second)
        {
            std::cout << value << " ";
        }
        std::cout << std::endl;
    }
}

void Expand(map<string, int> &MNT_MDTP, map<string, int> &MNT_PP, vector<string>
&MDT, map<string, vector<string>> &FPL, vector<string> input, vector<string> &in-
put_copy) {
    cout << "Ready to expand" << endl;
    vector<string> output;
    int inputsize = input.size();

    // Open the output file for writing
    ofstream outputFile("macroexpanded.txt");
    if (!outputFile.is_open()) {
        cerr << "Error: Unable to open file for writing." << endl;
        return;
    }

    for (int i = 0; i < inputsize; i++) {
        if (input[i] == ";") {
            outputFile << endl;
            continue;
        };
        if (input[i] == "MACRO") {
            //skip the macro
            while (input[i] != "MEND") {
                i++;
            }
            continue;
        }
    }

    //if macro call is found
```

```

if (MNT_MDTP.find(input[i]) != MNT_MDTP.end()) {
    string Macro_Name = input[i];
    map<int, string> APL;
    int MDTindex = MNT_MDTP[input[i]];
    MDTindex++;

    i++;
    int ind = 0;
    while (input[i] != ";") {
        APL[ind] = input[i];
        i++;
        ind++;
        if(i >= inputsize) break;
    }

    // cout << "The macro name is " << Macro_Name << MDTindex<<endl;

    //creating the output
    while (MDT[MDTindex] != "MEND") {
        string line = MDT[MDTindex];
        int size = line.size();
        string temp = "";
        for (int i = 0; i < size; i++) {
            if (line[i] == ' ' || line[i] == ';') {
                outputFile << temp << " ";
                if (line[i] == ';') outputFile << endl;
                temp = "";
                continue;
            }

            if (line[i] == '#') {
                i++;
                int ind = line[i] - '0';
                outputFile << APL[ind] << " ";
                continue;
            }

            temp += line[i];
        }
        MDTindex++;
    }
}
}

```

```
// Close the output file
outputFile.close();

cout << "The macro expansion is done." << endl;
}

int main(){

    //macro name table structure
    map<string,int>MNT_MDTP;
    map<string,int>MNT_PP;
    vector<string>input_copy;

    //FPL and APL
    map<string,vector<string> >FPL,APL;

    //macro defination table
    vector<string>MDT;

    //input file
    vector<string>input;

    //files to read and save;
    string readfrom;
    string saveas;

    cout<<" Enter file name to read"<<endl;
    cin>>readfrom;
    if(readfile(readfrom,input)==0){

        //cout<<"The inputfile is"<<endl;
        //print_vector(input);
        //filling the datasets
        pass1(MNT_MDTP,MNT_PP,MDT,FPL,APL,input,input_copy);

        cout<<"The MNT is"<<endl;
        Map_print(MNT_MDTP);
        Map_print(MNT_PP);

        cout<<"The MDT is"<<endl;
        print_vector(MDT);

        // int temp=MDT.size();
        // cout<<temp<<endl;
```



```
cout<<"The FPL and APL is"<<endl;
Map_print2(FPL);

// cout<<"The input_copy is"<<endl;
// print_vector(input_copy);

//cout<<"The APL is"<<endl;
//Map_print2(APL);

// pass2(MNT_MDTP,MNT_PP,MDT,FPL,APL,input, input_copy);
Expand(MNT_MDTP,MNT_PP,MDT,FPL,input, input_copy);

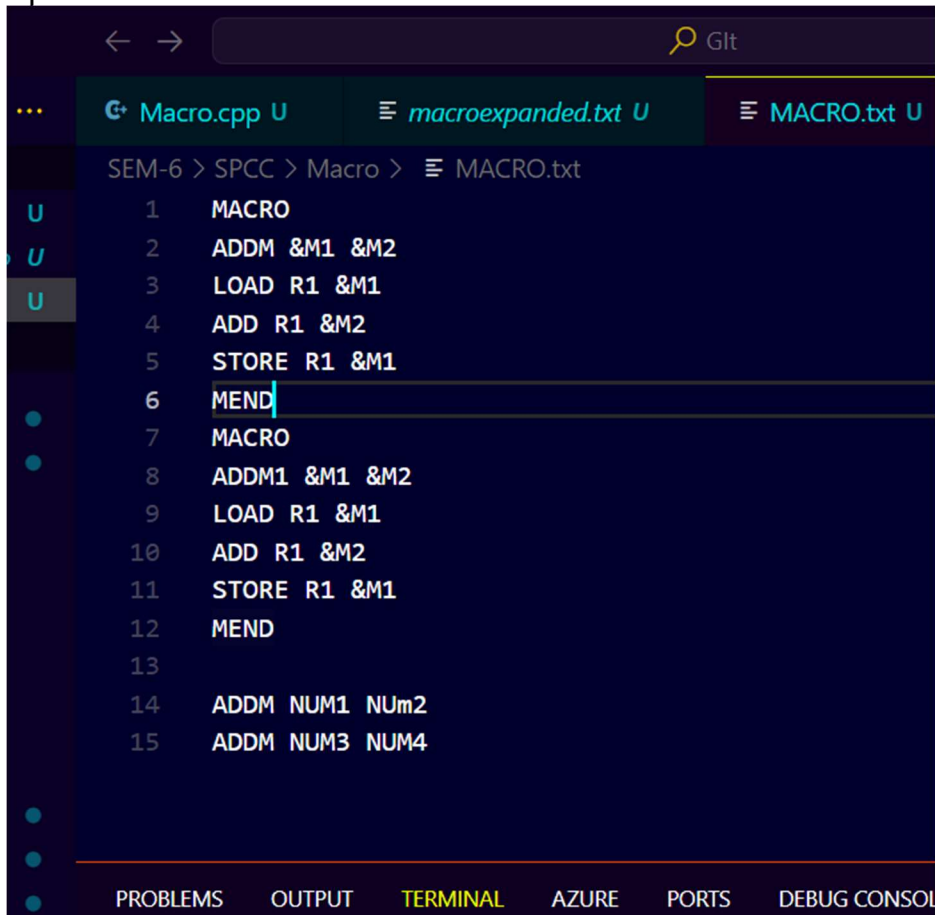
}

return 0;
}
```

---

End Result:

Input file:



```
SEM-6 > SPCC > Macro > MACRO.txt
1  MACRO
2  ADDM &M1 &M2
3  LOAD R1 &M1
4  ADD R1 &M2
5  STORE R1 &M1
6  MEND
7  MACRO
8  ADDM1 &M1 &M2
9  LOAD R1 &M1
10 ADD R1 &M2
11 STORE R1 &M1
12 MEND
13
14 ADDM NUM1 NUM2
15 ADDM NUM3 NUM4
```

PROBLEMS OUTPUT **TERMINAL** AZURE PORTS DEBUG CONSOLE

```

PS E:\Git> cd "e:\Git\SEM-6\SPCC\Macro\" ; if ($?) { g++ Macro.cpp -o Macro } ; if ($?) { .\Macro }
Enter file name to read
macro.txt
The MNT is
ADDM 0
ADDM1 5
ADDM 2
ADDM1 2
The MDT is
ADDM 0 1
LOAD R1 #0 ;
ADD R1 #1 ;
STORE R1 #0 ;
MEND
ADDM1 0 1
LOAD R1 #0 ;
ADD R1 #1 ;
STORE R1 #0 ;
MEND

The FPL and APL is
ADDM: &M1 &M2
ADDM1: &M1 &M2
Ready to expand
The macro expansion is done.
PS E:\Git\SEM-6\SPCC\Macro>

```

Output file

SEM-6 > SPCC > Macro > ≡ macroexpanded.txt

```

1
2
3
4    LOAD R1 NUM1
5    ADD R1 NUM2
6    STORE R1 NUM1
7    LOAD R1 NUM3
8    ADD R1 NUM4
9    STORE R1 NUM3
10

```

