Brunel University London

重慶郵電大學
CHONGQING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

C language program report

COURSE: C programming language

CALSS: 34092102

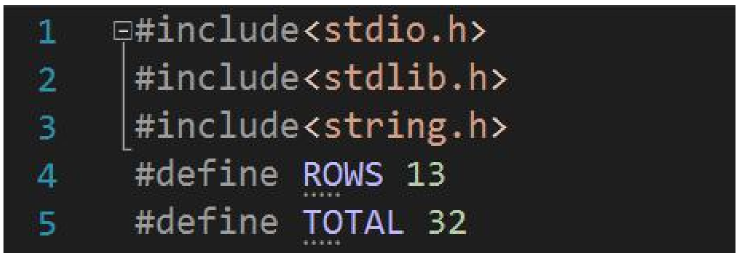
NAME: Xukang Liu

BRUNEL ID: 2161047

CQUPT ID: 2021215069

Program flow

1. Header file

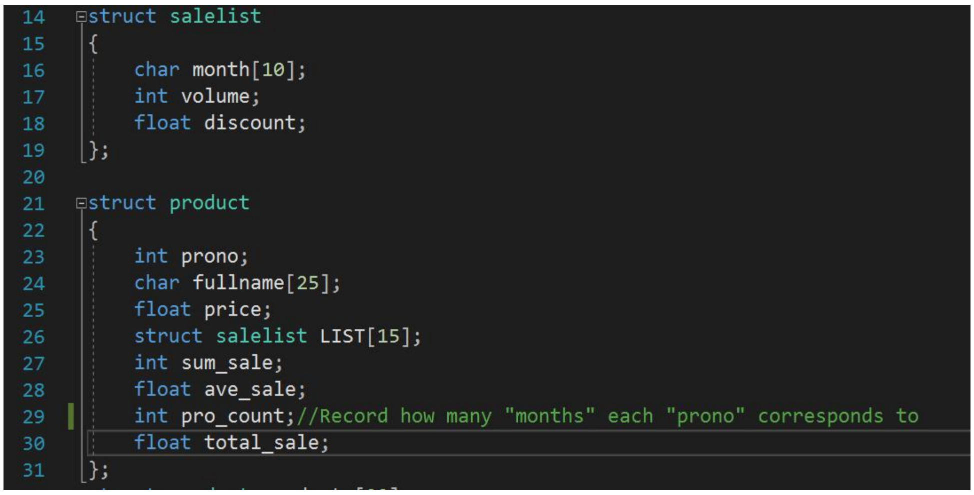


```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define ROWS 13
#define TOTAL 32
```

Use “string.h” to definite some string functions, such as “strcpy”,”strcmp”.

“ROWS” is the total number of products in the “product.txt” file and the number of pronos. “TOTAL” is the total rows in the “salelist.txt”file.

2. Structure definition

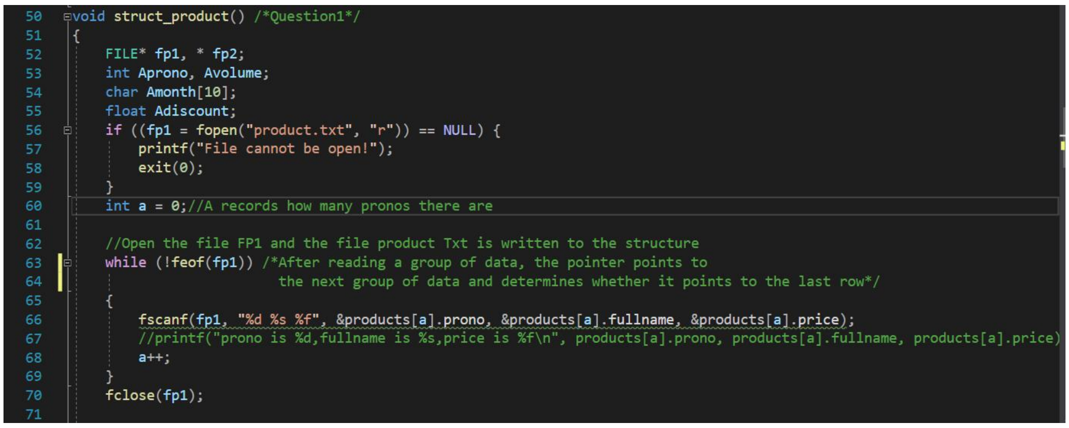


```c
struct salelist
{
    char month[10];
    int volume;
    float discount;
};

struct product
{
    int prono;
    char fullname[25];
    float price;
    struct salelist LIST[15];
    int sum_sale;
    float ave_sale;
    int pro_count;//Record how many "months" each "prono" corresponds to
    float total_sale;
};
```

This is the definition of structure. This is a nested structure, the outer layer is information in the "product.txt" file, and the inner layer is information in the "salelist.txt" file.Among them, "sum_sale" and "ave_sale" are the total sales per product and the average sales for each month."The pro_count" is used to record the sales of each product for several months, and the "total_sale" is to record the total annual profit of each product.

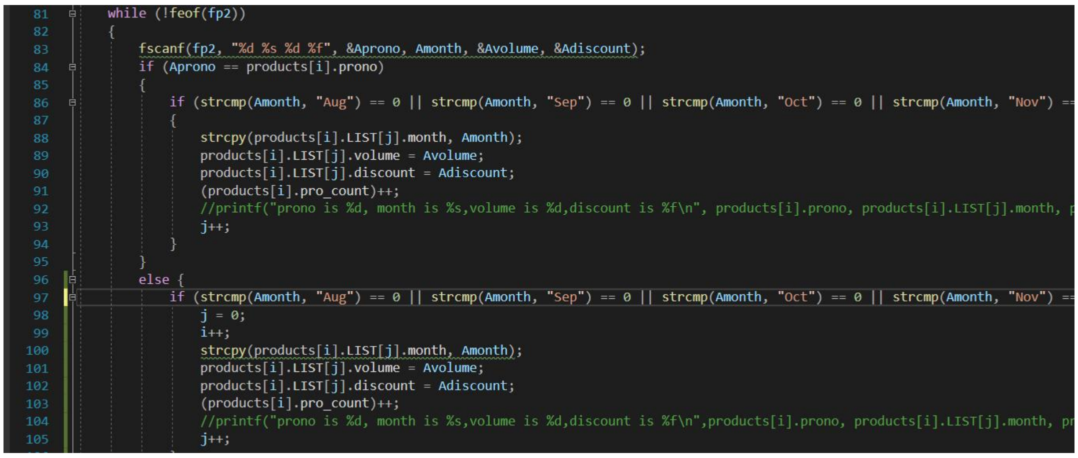
3. Read file

Here to set the first function, no return value, named "struct_product", this function is to open two files, and put the content of the file into the previously defined structure.



```c
void struct_product() /*Question1*/
{
    FILE* fp1, * fp2;
    int Aprono, Avolume;
    char Amonth[10];
    float Adiscount;
    if ((fp1 = fopen("product.txt", "r")) == NULL) {
        printf("File cannot be open!");
        exit(0);
    }
    int a = 0;//A records how many pronos there are

    //Open the file FP1 and the file product Txt is written to the structure
    while (!feof(fp1)) /*After reading a group of data, the pointer points to
                         the next group of data and determines whether it points to the last row*/
    {
        fscanf(fp1, "%d %s %f", &products[a].prono, &products[a].fullname, &products[a].price);
        //printf("prono is %d,fullname is %s,price is %f\n", products[a].prono, products[a].fullname, products[a].price)
        a++;
    }
    fclose(fp1);

```

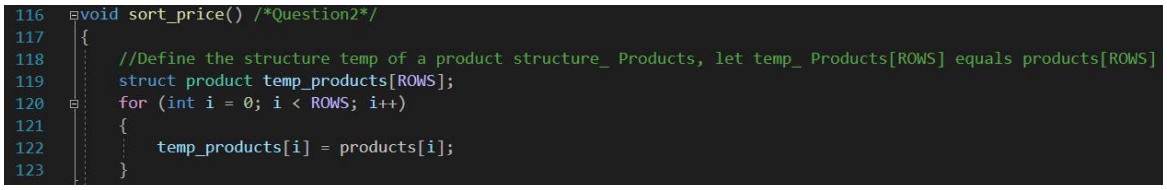
This is to put the first file into the structure, first with the "if" statement to judge whether you can open the file, if you can open, start going through the file, through the "while" loop line by line to put the data in the file into the structure, "!The feof () " is to judge whether to reach the end of the file and stop the loop if it arrives.If not, output " File cannot be open!", and then exit the run.



```c
    while (!feof(fp2))
    {
        fscanf(fp2, "%d %s %d %f", &Aprono, Amonth, &Avolume, &Adiscount);
        if (Aprono == products[i].prono)
        {
            if (strcmp(Amonth, "Aug") == 0 || strcmp(Amonth, "Sep") == 0 || strcmp(Amonth, "Oct") == 0 || strcmp(Amonth, "Nov") ==
            {
                strcpy(products[i].LIST[j].month, Amonth);
                products[i].LIST[j].volume = Avolume;
                products[i].LIST[j].discount = Adiscount;
                (products[i].pro_count)++;
                //printf("prono is %d, month is %s,volume is %d,discount is %f\n", products[i].prono, products[i].LIST[j].month, p
                j++;
            }
        }
        else {
            if (strcmp(Amonth, "Aug") == 0 || strcmp(Amonth, "Sep") == 0 || strcmp(Amonth, "Oct") == 0 || strcmp(Amonth, "Nov") ==
                j = 0;
                i++;
                strcpy(products[i].LIST[j].month, Amonth);
                products[i].LIST[j].volume = Avolume;
                products[i].LIST[j].discount = Adiscount;
                (products[i].pro_count)++;
                //printf("prono is %d, month is %s,volume is %d,discount is %f\n",products[i].prono, products[i].LIST[j].month, pr
                j++;
```

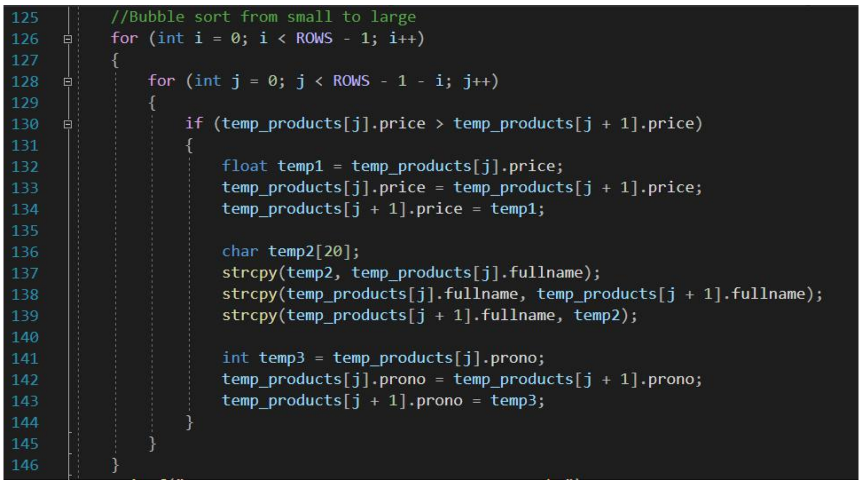
This is to put the second file into the structure, similar to putting the first file into the structure, but the second file has to put the judgment because it goes into the structure.For convenience, we first defined four intermediate variables, "Aprono, Amoth, Avolume, Adiscount".The "salelist.txt" file is nested in the "product.txt" file, and the connection between the two files is "prono", so first judge whether the prono of the read "salelist.txt" file is the same as the first file, and if the same, go to the next step.In order to prevent data in "month" file data that matches the real situation, use "strcmp()" for another judgment and put the data in if the real situation is met. If the "prono" is different for the two files, the "i" adds a numerical value and assigns the data to the next array.

4. Sort based on price



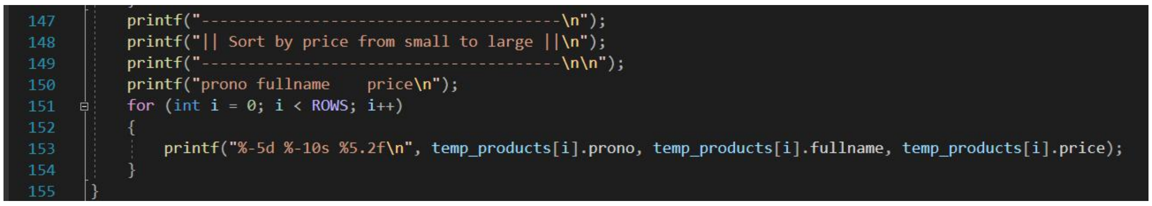
```c
void sort_price() /*Question2*/
{
    //Define the structure temp of a product structure_ Products, let temp_ Products[ROWS] equals products[ROWS]
    struct product temp_products[ROWS];
    for (int i = 0; i < ROWS; i++)
    {
        temp_products[i] = products[i];
    }
```

The function of this function is to sort the "product.txt" files from small to large by price size.In order not to affect the original construct, a new construct "temp_products [ROWS]" was first defined, traversed through the construct and assigned values one by one.



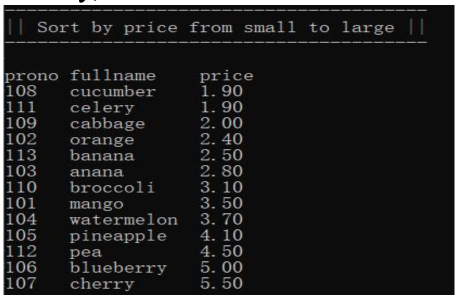
```c
    //Bubble sort from small to large
    for (int i = 0; i < ROWS - 1; i++)
    {
        for (int j = 0; j < ROWS - 1 - i; j++)
        {
            if (temp_products[j].price > temp_products[j + 1].price)
            {
                float temp1 = temp_products[j].price;
                temp_products[j].price = temp_products[j + 1].price;
                temp_products[j + 1].price = temp1;

                char temp2[20];
                strcpy(temp2, temp_products[j].fullname);
                strcpy(temp_products[j].fullname, temp_products[j + 1].fullname);
                strcpy(temp_products[j + 1].fullname, temp2);

                int temp3 = temp_products[j].prono;
                temp_products[j].prono = temp_products[j + 1].prono;
                temp_products[j + 1].prono = temp3;
            }
        }
    }
```

The sorting was started by bubbling sorting.



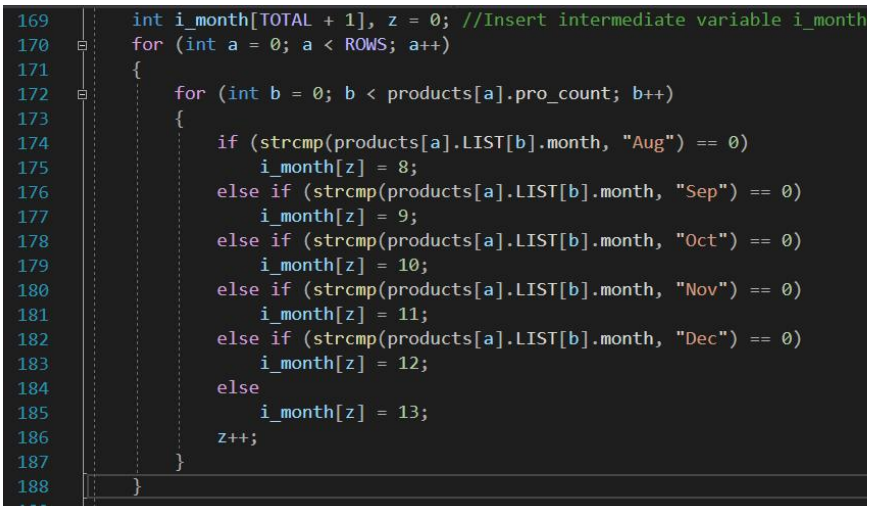
```c
    printf("--------------------------------------\n");
    printf("|| Sort by price from small to large ||\n");
    printf("--------------------------------------\n\n");
    printf("prono fullname    price\n");
    for (int i = 0; i < ROWS; i++)
    {
        printf("%-5d %-10s %5.2f\n", temp_products[i].prono, temp_products[i].fullname, temp_products[i].price);
    }
}
```

Finally, we traverse the structure and output the corresponding results.



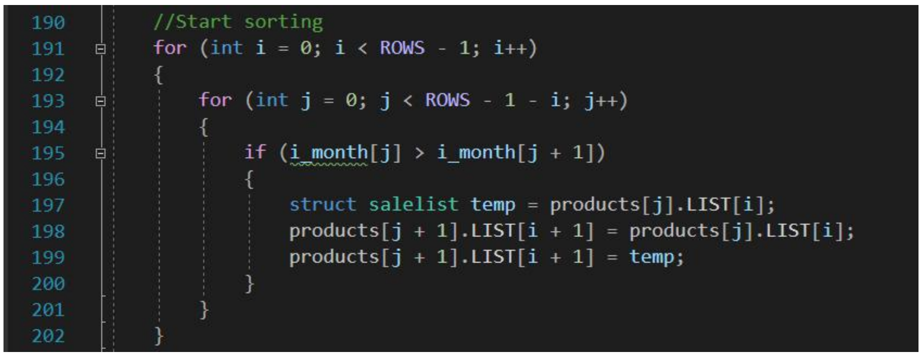
```
--------------------------------------
|| Sort by price from small to large ||
--------------------------------------

prono fullname    price
108   cucumber     1.90
111   celery       1.90
109   cabbage      2.00
102   orange       2.40
113   banana       2.50
103   anana        2.80
110   broccoli     3.10
101   mango        3.50
104   watermelon   3.70
105   pineapple    4.10
112   pea          4.50
106   blueberry    5.00
107   cherry       5.50
```

5. Sort based on month



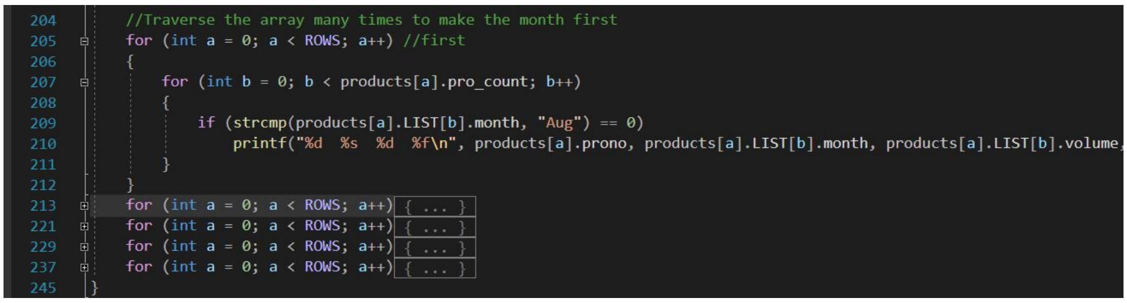
```c
    int i_month[TOTAL + 1], z = 0; //Insert intermediate variable i_month
    for (int a = 0; a < ROWS; a++)
    {
        for (int b = 0; b < products[a].pro_count; b++)
        {
            if (strcmp(products[a].LIST[b].month, "Aug") == 0)
                i_month[z] = 8;
            else if (strcmp(products[a].LIST[b].month, "Sep") == 0)
                i_month[z] = 9;
            else if (strcmp(products[a].LIST[b].month, "Oct") == 0)
                i_month[z] = 10;
            else if (strcmp(products[a].LIST[b].month, "Nov") == 0)
                i_month[z] = 11;
            else if (strcmp(products[a].LIST[b].month, "Dec") == 0)
                i_month[z] = 12;
            else
                i_month[z] = 13;
            z++;
        }
    }
```

This function is to output the data in "salelist.txt" in the order of months, but because months are stored as a string in the file, if you want to sort by month, you first need to convert into numbers and then sort.In order to sort, the intermediate variable "i_month []" was introduced to traverse the structure, the larger the month, the larger the number.



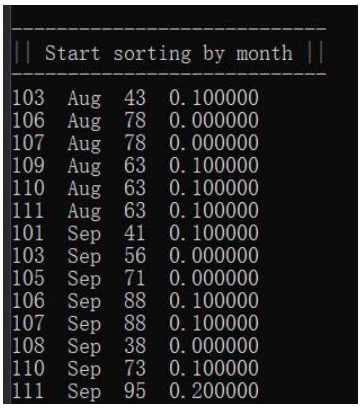
```c
    //Start sorting
    for (int i = 0; i < ROWS - 1; i++)
    {
        for (int j = 0; j < ROWS - 1 - i; j++)
        {
            if (i_month[j] > i_month[j + 1])
            {
                struct salelist temp = products[j].LIST[i];
                products[j + 1].LIST[i + 1] = products[j].LIST[i];
                products[j + 1].LIST[i + 1] = temp;
            }
        }
    }
```

After transformation into numbers, the intermediate structural variable is introduced here to rank the constructs of the entire nested inner layer.But if you sort this, you get the ranking of each "prono", which is, serial number prioritized, and then month.So to prioritize the months, use the following "for" loop.

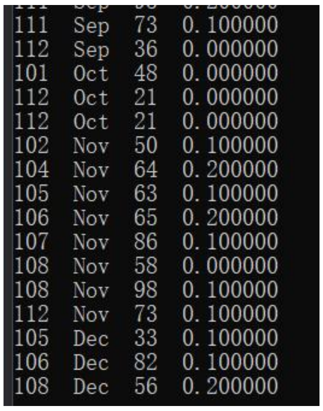


```c
    //Traverse the array many times to make the month first
    for (int a = 0; a < ROWS; a++) //first
    {
        for (int b = 0; b < products[a].pro_count; b++)
        {
            if (strcmp(products[a].LIST[b].month, "Aug") == 0)
                printf("%d  %s  %d  %f\n", products[a].prono, products[a].LIST[b].month, products[a].LIST[b].volume,
        }
    }
    for (int a = 0; a < ROWS; a++) { ... }
    for (int a = 0; a < ROWS; a++) { ... }
    for (int a = 0; a < ROWS; a++) { ... }
    for (int a = 0; a < ROWS; a++) { ... }
}
```

Because the month in the file is August-December, so with five "for" loops, each loop is proposed for a month, first in August, and then in September, so that until December, the month can be proposed to sort first.The results are shown in the figure below.

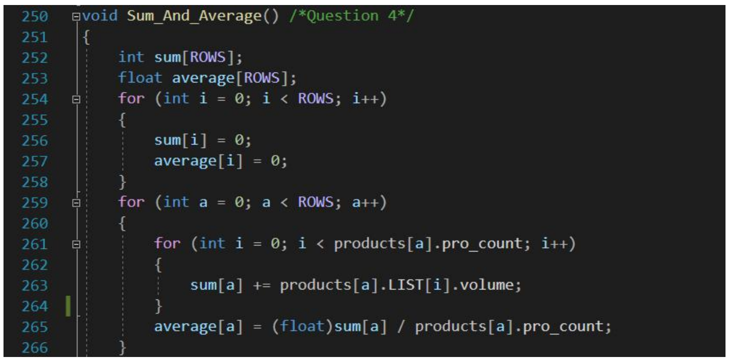


```
-----------------------------
|| Start sorting by month ||
-----------------------------
103  Aug  43  0.100000
106  Aug  78  0.000000
107  Aug  78  0.000000
109  Aug  63  0.100000
110  Aug  63  0.100000
111  Aug  63  0.100000
101  Sep  41  0.100000
103  Sep  56  0.000000
105  Sep  71  0.000000
106  Sep  88  0.100000
107  Sep  88  0.100000
108  Sep  38  0.000000
110  Sep  73  0.100000
111  Sep  95  0.200000
```



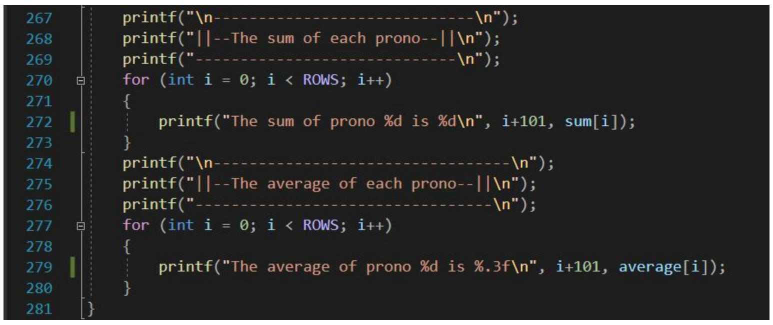
```
111  Sep  73  0.100000
112  Sep  36  0.000000
101  Oct  48  0.000000
112  Oct  21  0.000000
112  Oct  21  0.000000
102  Nov  50  0.100000
104  Nov  64  0.200000
105  Nov  63  0.100000
106  Nov  65  0.200000
107  Nov  86  0.100000
108  Nov  58  0.000000
108  Nov  98  0.100000
112  Nov  73  0.100000
105  Dec  33  0.100000
106  Dec  82  0.100000
108  Dec  56  0.200000
```

6.Calculate the total and the average sale volume



```c
void Sum_And_Average() /*Question 4*/
{
    int sum[ROWS];
    float average[ROWS];
    for (int i = 0; i < ROWS; i++)
    {
        sum[i] = 0;
        average[i] = 0;
    }
    for (int a = 0; a < ROWS; a++)
    {
        for (int i = 0; i < products[a].pro_count; i++)
        {
            sum[a] += products[a].LIST[i].volume;
        }
        average[a] = (float)sum[a] / products[a].pro_count;
    }
```

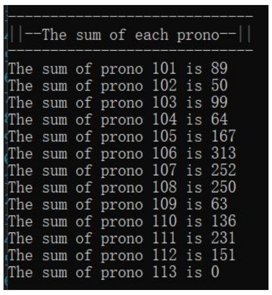
This function is to calculate the total and average sales volume for each product separately.The intermediate variables "sum []" and "average []" are introduced to sum and average the structure through "for" loops, but because "average []" is a "float" type variable, but "sum []" and " products []. The pro_count " are all integer variables, so converted with forced type to floating-point type.



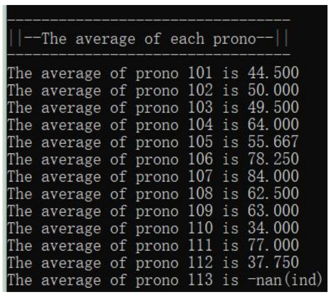
```c
    printf("\n----------------------------\n");
    printf("||--The sum of each prono--||\n");
    printf("----------------------------\n");
    for (int i = 0; i < ROWS; i++)
    {
        printf("The sum of prono %d is %d\n", i+101, sum[i]);
    }
    printf("\n--------------------------------\n");
    printf("||--The average of each prono--||\n");
    printf("--------------------------------\n");
    for (int i = 0; i < ROWS; i++)
    {
        printf("The average of prono %d is %.3f\n", i+101, average[i]);
    }
}
```

This is the output segment, where the traversing structure successively outputs the total and average sales volume of each product.

The results are shown in the figure below。

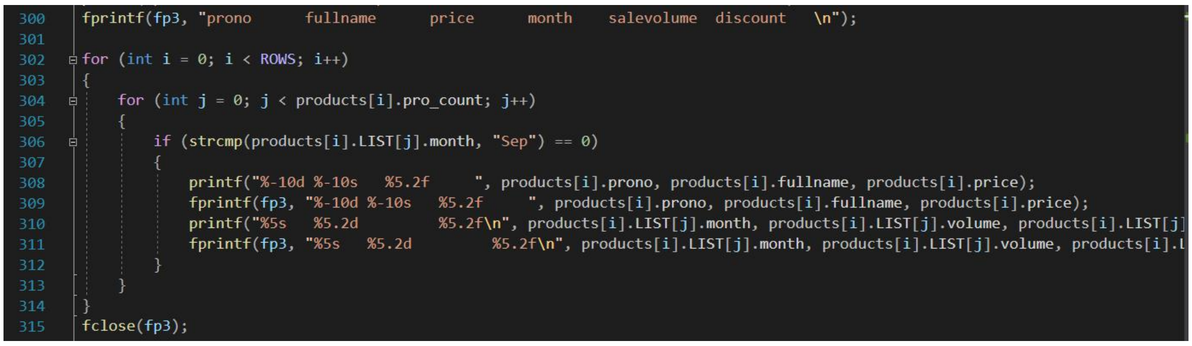


```
-----------------------------
||--The sum of each prono--||
-----------------------------
The sum of prono 101 is 89
The sum of prono 102 is 50
The sum of prono 103 is 99
The sum of prono 104 is 64
The sum of prono 105 is 167
The sum of prono 106 is 313
The sum of prono 107 is 252
The sum of prono 108 is 250
The sum of prono 109 is 63
The sum of prono 110 is 136
The sum of prono 111 is 231
The sum of prono 112 is 151
The sum of prono 113 is 0
```



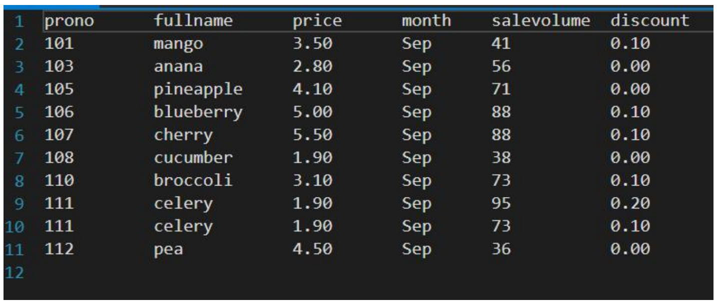
```
---------------------------------
||--The average of each prono--||
---------------------------------
The average of prono 101 is 44.500
The average of prono 102 is 50.000
The average of prono 103 is 49.500
The average of prono 104 is 64.000
The average of prono 105 is 55.667
The average of prono 106 is 78.250
The average of prono 107 is 84.000
The average of prono 108 is 62.500
The average of prono 109 is 63.000
The average of prono 110 is 34.000
The average of prono 111 is 77.000
The average of prono 112 is 37.750
The average of prono 113 is -nan(ind)
```

7.Output all sales product information



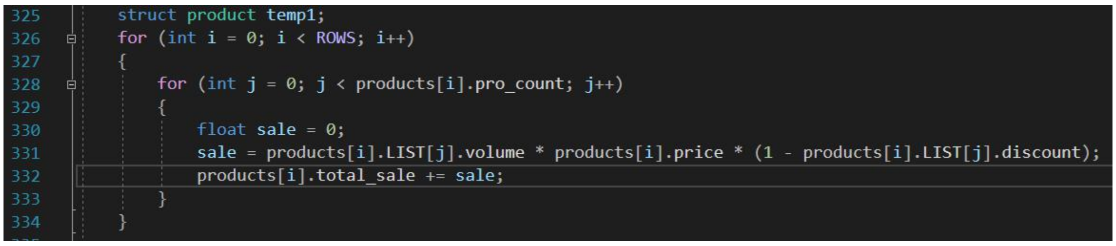
```
fprintf(fp3, "prono      fullname      price      month    salevolume  discount    \n");

for (int i = 0; i < ROWS; i++)
{
    for (int j = 0; j < products[i].pro_count; j++)
    {
        if (strcmp(products[i].LIST[j].month, "Sep") == 0)
        {
            printf("%-10d %-10s   %5.2f     ", products[i].prono, products[i].fullname, products[i].price);
            fprintf(fp3, "%-10d %-10s   %5.2f     ", products[i].prono, products[i].fullname, products[i].price);
            printf("%5s   %5.2d         %5.2f\n", products[i].LIST[j].month, products[i].LIST[j].volume, products[i].LIST[j]
            fprintf(fp3, "%5s   %5.2d         %5.2f\n", products[i].LIST[j].month, products[i].LIST[j].volume, products[i].L
        }
    }
}
fclose(fp3);
```

This function functions to output all the September sales data containing the product information, write the results to a new txt file, and name the file with my Brunel id.The "2161047.txt" file was already opened earlier with the "fopen ()", where my Brunel id is "2161047".First traverse through the structure, find out all the information about September, and then enter it into the file in the "fprintf()" format.The following figure shows the output results.



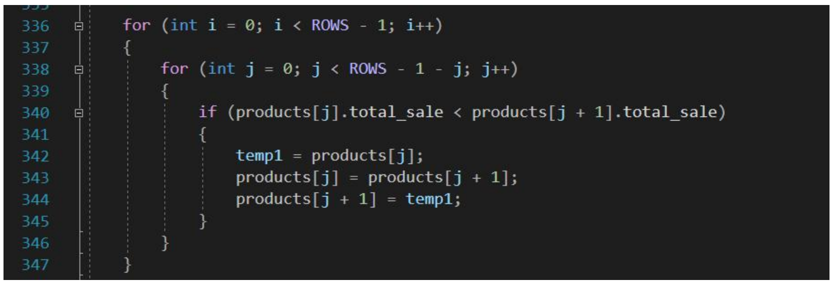
```
prono      fullname     price      month    salevolume  discount
101        mango        3.50       Sep      41          0.10
103        anana        2.80       Sep      56          0.00
105        pineapple    4.10       Sep      71          0.00
106        blueberry    5.00       Sep      88          0.10
107        cherry       5.50       Sep      88          0.10
108        cucumber     1.90       Sep      38          0.00
110        broccoli     3.10       Sep      73          0.10
111        celery       1.90       Sep      95          0.20
111        celery       1.90       Sep      73          0.10
112        pea          4.50       Sep      36          0.00
```

8.Calculate and output the total sale



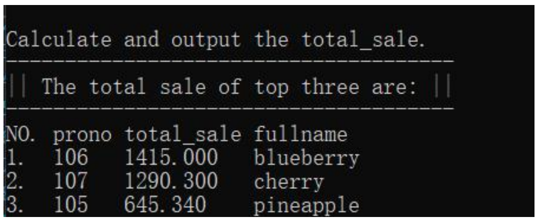
```
    struct product temp1;
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < products[i].pro_count; j++)
        {
            float sale = 0;
            sale = products[i].LIST[j].volume * products[i].price * (1 - products[i].LIST[j].discount);
            products[i].total_sale += sale;
        }
    }
```

This function is to calculate the total sale for each product and then output the top three products.The total sale of each product is first calculated according to the formula and put into the structure.

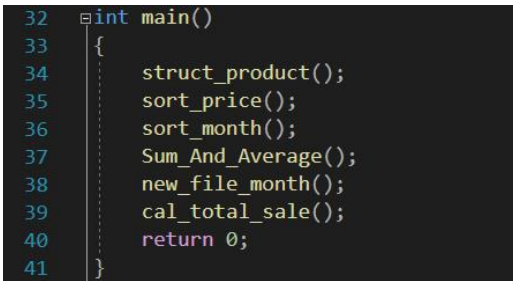

```
for (int i = 0; i < ROWS - 1; i++)
{
    for (int j = 0; j < ROWS - 1 - j; j++)
    {
        if (products[j].total_sale < products[j + 1].total_sale)
        {
            temp1 = products[j];
            products[j] = products[j + 1];
            products[j + 1] = temp1;
        }
    }
}
```

The intermediate variable "temp1" of a construct was then defined, sorted from high to low by total sale, and then the top three products were output.

Here are the output results。


```
Calculate and output the total_sale.
--------------------------------------
|| The total sale of top three are: ||
--------------------------------------
NO. prono total_sale fullname
1.  106   1415.000   blueberry
2.  107   1290.300   cherry
3.  105   645.340    pineapple
```

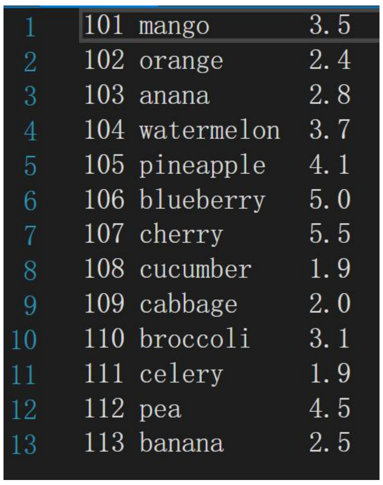
9.Main function


```
int main()
{
    struct_product();
    sort_price();
    sort_month();
    Sum_And_Average();
    new_file_month();
    cal_total_sale();
    return 0;
}
```

Since each previous step has a function, so the main function only needs to call the following function to implement the program running.

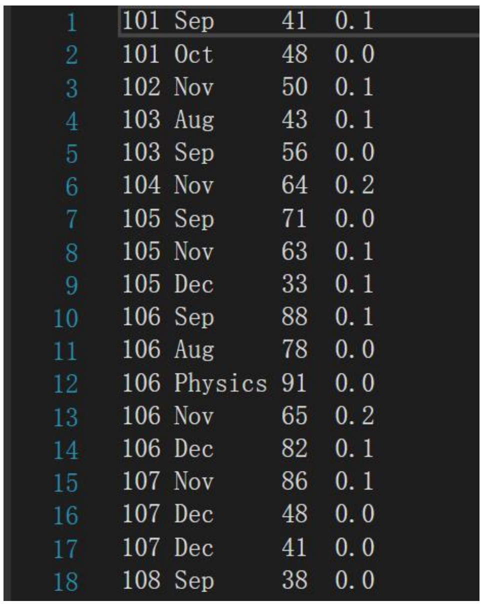
My Txt File

1. product.txt

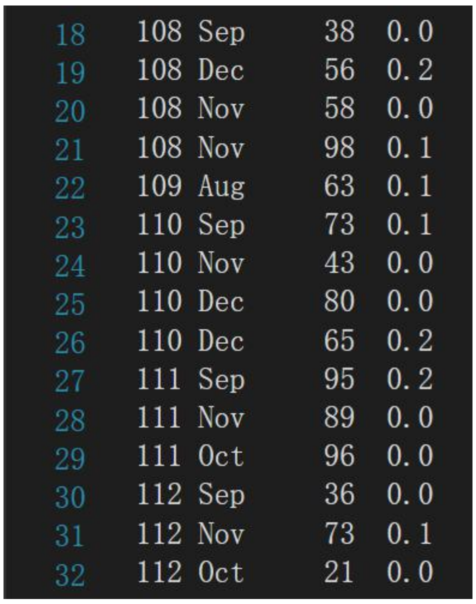


```
101 mango      3.5
102 orange     2.4
103 anana      2.8
104 watermelon 3.7
105 pineapple  4.1
106 blueberry  5.0
107 cherry     5.5
108 cucumber   1.9
109 cabbage    2.0
110 broccoli   3.1
111 celery     1.9
112 pea        4.5
113 banana     2.5
```

2. salelist.txt



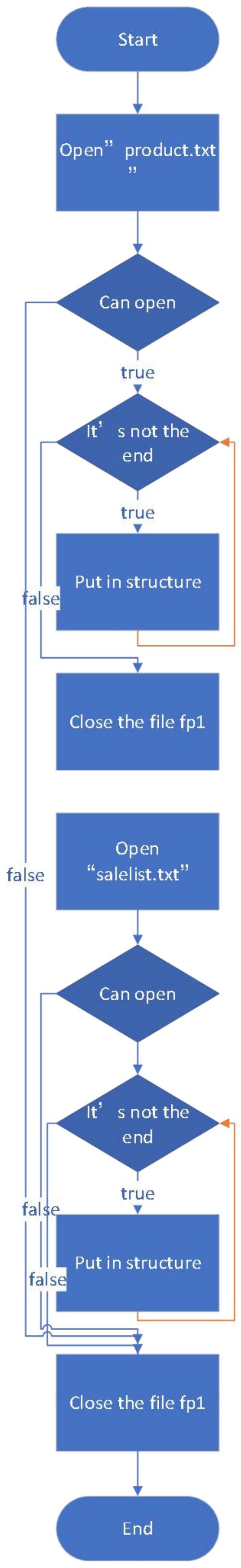
```
101 Sep     41  0.1
101 Oct     48  0.0
102 Nov     50  0.1
103 Aug     43  0.1
103 Sep     56  0.0
104 Nov     64  0.2
105 Sep     71  0.0
105 Nov     63  0.1
105 Dec     33  0.1
106 Sep     88  0.1
106 Aug     78  0.0
106 Physics 91  0.0
106 Nov     65  0.2
106 Dec     82  0.1
107 Nov     86  0.1
107 Dec     48  0.0
107 Dec     41  0.0
108 Sep     38  0.0
```



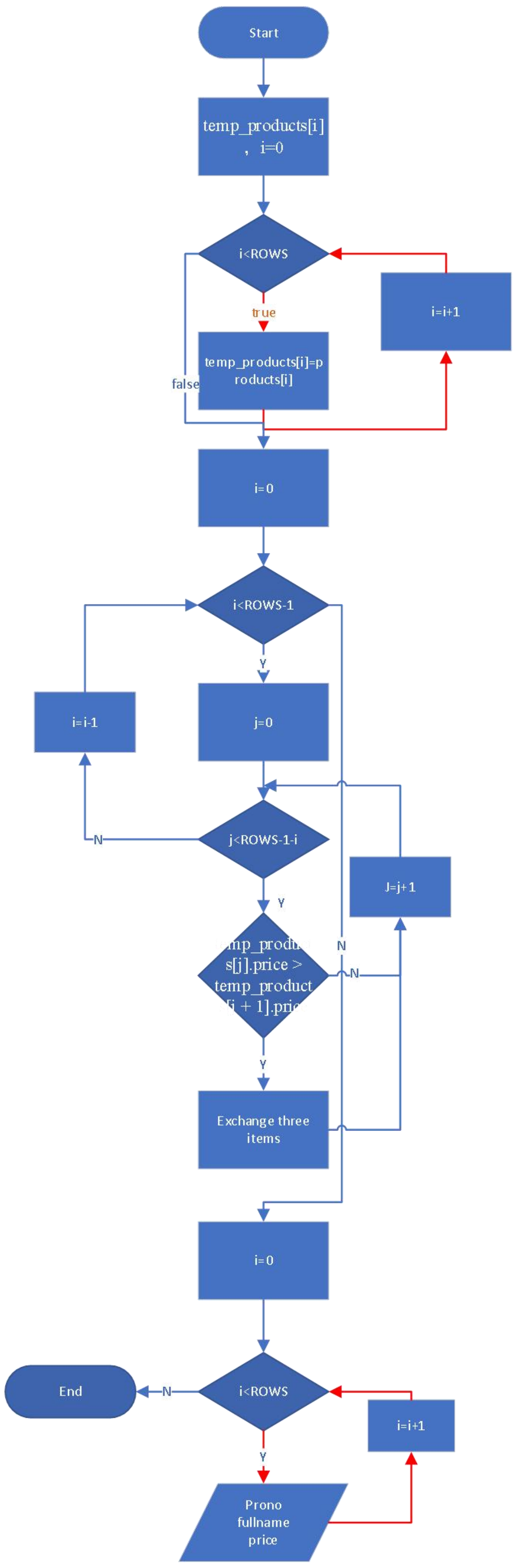
```
108 Sep     38  0.0
108 Dec     56  0.2
108 Nov     58  0.0
108 Nov     98  0.1
109 Aug     63  0.1
110 Sep     73  0.1
110 Nov     43  0.0
110 Dec     80  0.0
110 Dec     65  0.2
111 Sep     95  0.2
111 Nov     89  0.0
111 Oct     96  0.0
112 Sep     36  0.0
112 Nov     73  0.1
112 Oct     21  0.0
```

Flow Chart

FUNCTION 1

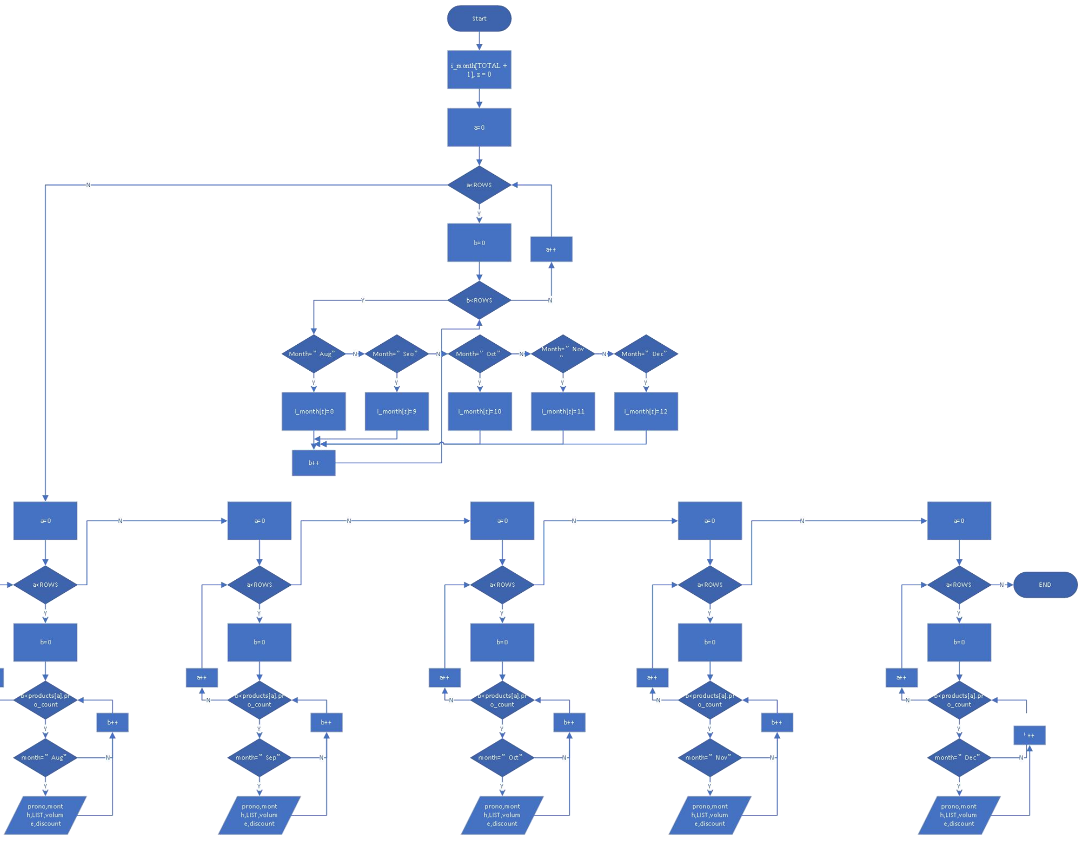


FUNCTION2

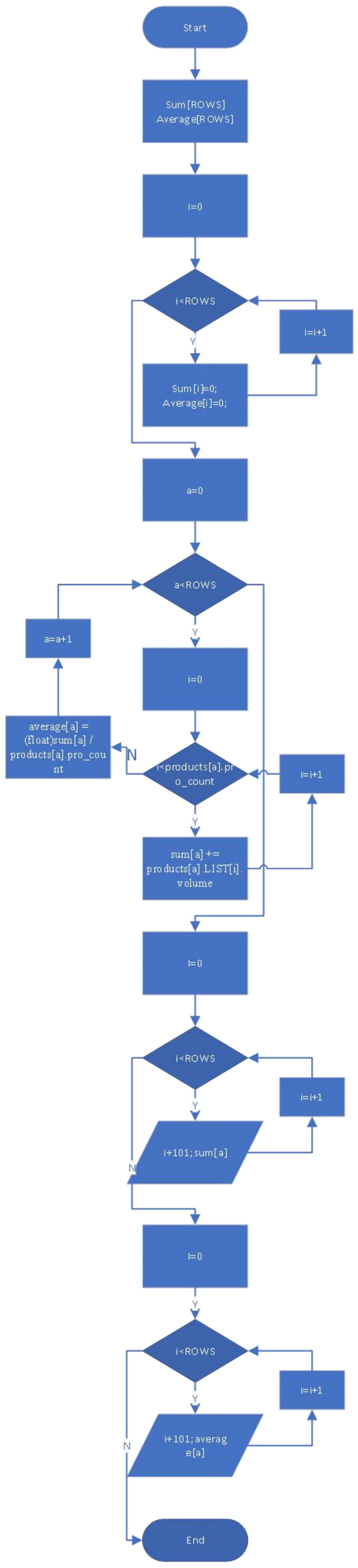


Start
temp_products[i] , i=0
i<ROWS
true
false
temp_products[i]=products[i]
i=i+1
i=0
i<ROWS-1
Y
N
j=0
i=i-1
j<ROWS-1-i
N
Y
J=j+1
mp_produ s[j].price > temp_product [j + 1].pric
N
Y
Exchange three items
i=0
i<ROWS
N
End
Y
i=i+1
Prono fullname price

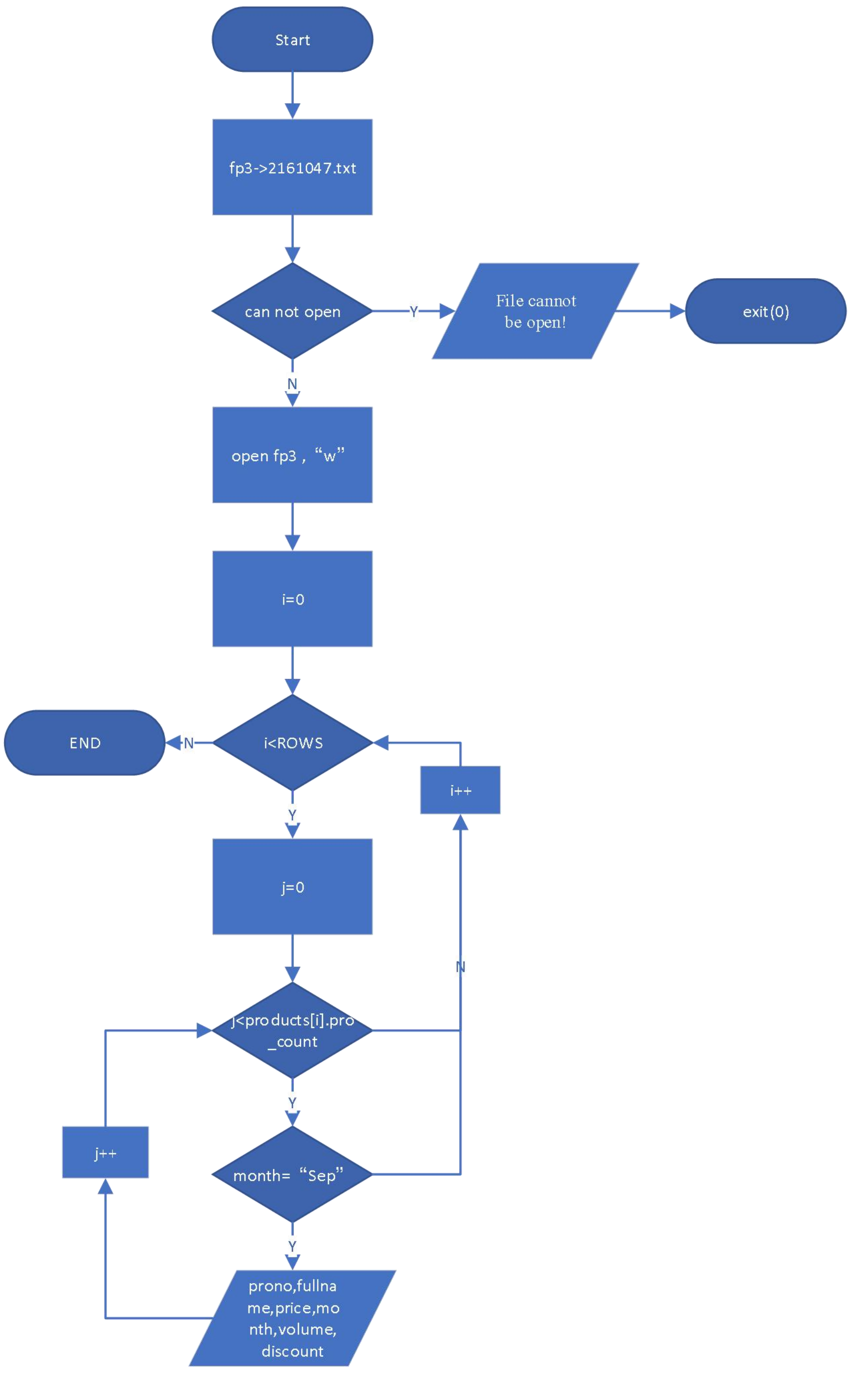
FUNCTION 3


Start
i_month[TOTAL + 1], z = 0
a=0
a<ROWS
b=0
a++
b<ROWS
Month=" Aug"
Month=" Seo"
Month=" Oct"
Month=" Nov "
Month=" Dec"
i_month[z]=8
i_month[z]=9
i_month[z]=10
i_month[z]=11
i_month[z]=12
b++
a=0
a<ROWS
b=0
a++
b<products[a].pro_count
b++
month=" Aug"
prono,month,LIST,volume,discount
month=" Sep"
month=" Oct"
month=" Nov"
month=" Dec"
END
Y
N

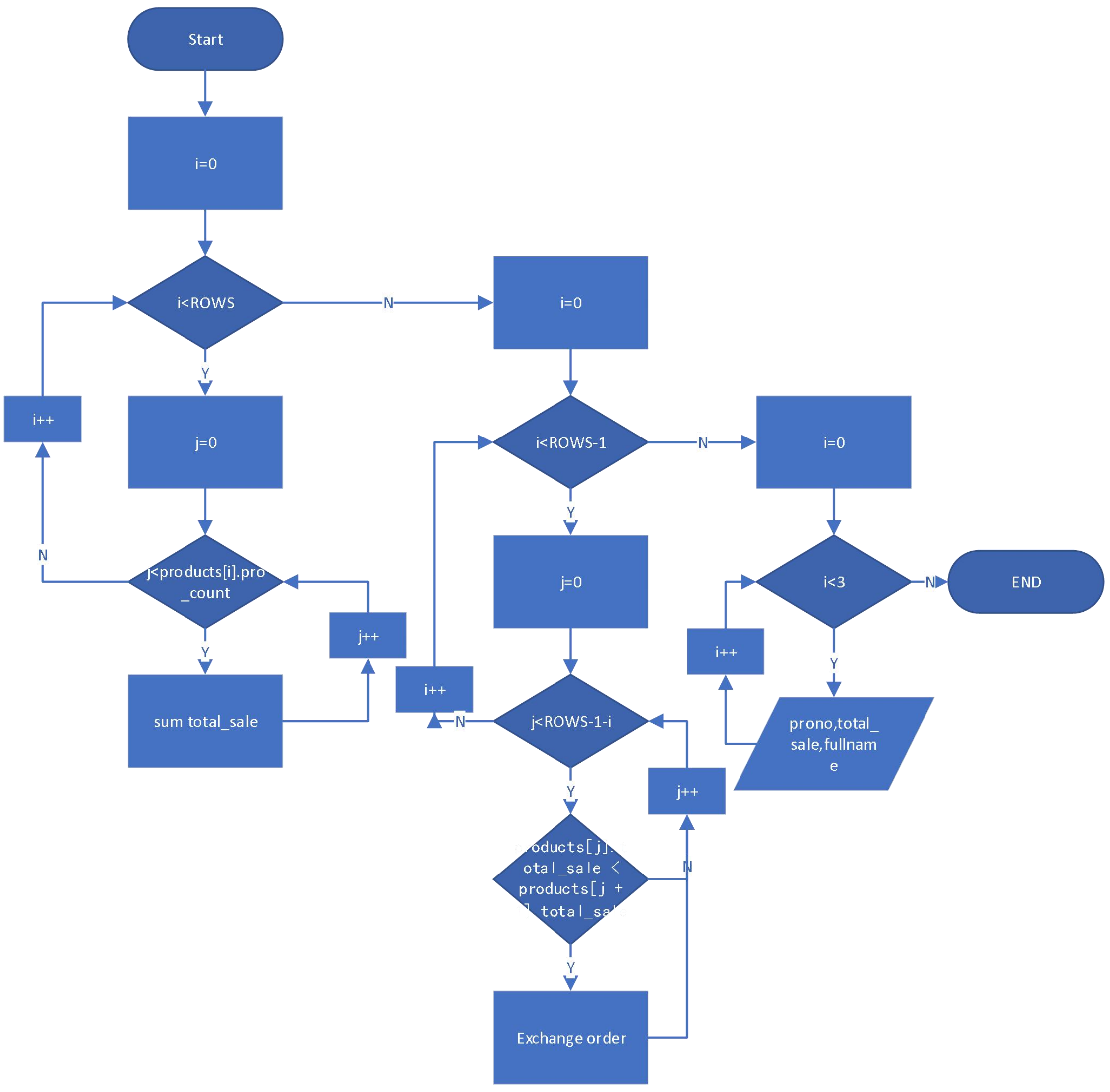
FUNCTION 4


Start
Sum[ROWS]
Average[ROWS]
i=0
i<ROWS
Y
i=i+1
Sum[i]=0;
Average[i]=0;
a=0
a<ROWS
Y
a=a+1
i=0
average[a] = (float)sum[a] / products[a].pro_count
N
i<products[a].pro_count
i=i+1
Y
sum[a] += products[a].LIST[i].volume
I=0
i<ROWS
i=i+1
Y
i+101; sum[a]
N
I=0
Y
i<ROWS
i=i+1
Y
i+101; average[a]
N
End

FUNCTION 5


Start
fp3->2161047.txt
can not open
Y
File cannot be open!
exit(0)
N
open fp3 , “w”
i=0
END
N
i<ROWS
i++
Y
j=0
N
j<products[i].pro_count
Y
j++
month= “Sep”
Y
prono,fullname,price,month,volume, discount

FUNCTION 6


Start
i=0
i<ROWS
N
Y
i++
j=0
N
j<products[i].pro_count
Y
j++
sum total_sale
i=0
i<ROWS-1
N
Y
j=0
i++
N
j<ROWS-1-i
Y
j++
oducts[j].
otal_sale <
products[j +
] total_sa
N
Y
Exchange order
i=0
i<3
N
END
i++
Y
prono,total_sale,fullname

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define ROWS 13
#define TOTAL 32

void struct_product();
void sort_price();
void sort_month();
void Sum_And_Average();
void new_file_month();
void cal_total_sale();

struct salelist
{
    char month[10];
    int volume;
    float discount;
};

struct product
{
    int prono;
    char fullname[25];
    float price;
    struct salelist LIST[15];
    int pro_count;//Record how many "months" each "prono" corresponds to
    float total_sale;
};
struct product products[20];

int main()
{
    struct_product();
    sort_price();
    sort_month();
    Sum_And_Average();
    new_file_month();
    cal_total_sale();
    return 0;
}

/*
@Function struct_product
@Desc Open the file which are "product" and "salelist",then read the data in the file,
judge it as valid information, and import it into the structure
*/
void struct_product() /*Question1*/
{
    FILE* fp1, * fp2;
    int Aprono, Avolume;
    char Amonth[10];
    float Adiscount;
    if ((fp1 = fopen("product.txt", "r")) == NULL) {
        printf("File cannot be open!");
        exit(0);
    }
    int a = 0;//a records how many pronos there are

    //Open the file FP1 and the file product Txt is written to the structure
    while (!feof(fp1)) /*After reading a group of data, the pointer points to
                          the next group of data and determines whether it points to the last row*/
    {
        fscanf(fp1, "%d %s %f", &products[a].prono, &products[a].fullname, &products[a].price);
        //printf("prono is %d,fullname is %s,price is %f\n", products[a].prono, products[a].fullname, products[a].price);
        a++;
    }
    fclose(fp1);



    //Open the file fp2
    if ((fp2 = fopen("salelist.txt", "r")) == NULL)
    {
        printf("File cannot be open!");
        exit(0);
    }
    int i = 0, j = 0;
    while (!feof(fp2))
    {
        fscanf(fp2, "%d %s %d %f", &Aprono, Amonth, &Avolume, &Adiscount);
        if (Aprono == products[i].prono)
        {
            if (strcmp(Amonth, "Aug") == 0 || strcmp(Amonth, "Sep") == 0 || strcmp(Amonth, "Oct") == 0 || strcmp(Amonth, "Nov") == 0 || strcmp(Amonth, "Dec") == 0)
            {
                strcpy(products[i].LIST[j].month, Amonth);
                products[i].LIST[j].volume = Avolume;
                products[i].LIST[j].discount = Adiscount;
                (products[i].pro_count)++;
                //printf("prono is %d, month is %s,volume is %d,discount is %f\n", products[i].prono, products[i].LIST[j].month, products[i].LIST[j].volume, products[i].LIST[j].discount);
                j++;
            }
        }
        else {
            if (strcmp(Amonth, "Aug") == 0 || strcmp(Amonth, "Sep") == 0 || strcmp(Amonth, "Oct") == 0 || strcmp(Amonth, "Nov") == 0 || strcmp(Amonth, "Dec") == 0){
                j = 0;
                i++;
                strcpy(products[i].LIST[j].month, Amonth);
                products[i].LIST[j].volume = Avolume;
                products[i].LIST[j].discount = Adiscount;
                (products[i].pro_count)++;
                //printf("prono is %d, month is %s,volume is %d,discount is %f\n",products[i].prono, products[i].LIST[j].month, products[i].LIST
                  [j].volume, products[i].LIST[j].discount);
                j++;
            }
        }
    }
    fclose(fp2);
}

/*
@function sort_price
@desc Sort the records based on price
*/
void sort_price() /*Question2*/
{
    //Define the structure temp of a product structure_ Products, let temp_ Products
      [ROWS] equals products[ROWS]
    struct product temp_products[ROWS];
    for (int i = 0; i < ROWS; i++)
    {
        temp_products[i] = products[i];
    }

    //Bubble sort from small to large
    for (int i = 0; i < ROWS - 1; i++)
    {
        for (int j = 0; j < ROWS - 1 - i; j++)
        {
            if (temp_products[j].price > temp_products[j + 1].price)
            {
                float temp1 = temp_products[j].price;
                temp_products[j].price = temp_products[j + 1].price;
                temp_products[j + 1].price = temp1;

                char temp2[20];
                strcpy(temp2, temp_products[j].fullname);
                strcpy(temp_products[j].fullname, temp_products[j + 1].fullname);
                strcpy(temp_products[j + 1].fullname, temp2);

                int temp3 = temp_products[j].prono;
                temp_products[j].prono = temp_products[j + 1].prono;
                temp_products[j + 1].prono = temp3;
            }
        }
    }
    printf("---------------------------------------\n");
    printf("|| Sort by price from small to large ||\n");
    printf("---------------------------------------\n\n");
    printf("prono fullname    price\n");
    for (int i = 0; i < ROWS; i++)
    {
        printf("%-5d %-10s %5.2f\n", temp_products[i].prono, temp_products
          [i].fullname, temp_products[i].price);
    }
}

/*
```

```c
@function sort_month
@desc Sort product’ s records based on month,and output the information of
  "product.txt",
but don’t out the information of "salelist.txt"
*/
void sort_month() /*Question 3*/
{
    printf("\n");
    printf("----------------------------\n");
    printf("|| Start sorting by month ||\n");
    printf("----------------------------\n");

    int i_month[TOTAL + 1], z = 0; //Insert intermediate variable i_month[], replace
      the string with an integer for comparison
    for (int a = 0; a < ROWS; a++)
    {
        for (int b = 0; b < products[a].pro_count; b++)
        {
            if (strcmp(products[a].LIST[b].month, "Aug") == 0)
                i_month[z] = 8;
            else if (strcmp(products[a].LIST[b].month, "Sep") == 0)
                i_month[z] = 9;
            else if (strcmp(products[a].LIST[b].month, "Oct") == 0)
                i_month[z] = 10;
            else if (strcmp(products[a].LIST[b].month, "Nov") == 0)
                i_month[z] = 11;
            else if (strcmp(products[a].LIST[b].month, "Dec") == 0)
                i_month[z] = 12;
            z++;
        }
    }

    //Start sorting
    for (int i = 0; i < ROWS - 1; i++)
    {
        for (int j = 0; j < ROWS - 1 - i; j++)
        {
            if (i_month[j] > i_month[j + 1])
            {
                struct salelist temp = products[j].LIST[i];
                products[j + 1].LIST[i + 1] = products[j].LIST[i];
                products[j + 1].LIST[i + 1] = temp;
            }
        }
    }

    //Traverse the array many times to make the month first
    for (int a = 0; a < ROWS; a++) //first
    {
        for (int b = 0; b < products[a].pro_count; b++)
        {
            if (strcmp(products[a].LIST[b].month, "Aug") == 0)
                printf("%d  %s  %d  %f\n", products[a].prono, products[a].LIST
                  [b].month, products[a].LIST[b].volume, products[a].LIST
                  [b].discount);
        }
    }
    for (int a = 0; a < ROWS; a++) //second
    {
        for (int b = 0; b < products[a].pro_count; b++)
        {
            if (strcmp(products[a].LIST[b].month, "Sep") == 0)
                printf("%d  %s  %d  %f\n", products[a].prono, products[a].LIST
                  [b].month, products[a].LIST[b].volume, products[a].LIST
                  [b].discount);
        }
    }
    for (int a = 0; a < ROWS; a++)  //third
    {
        for (int b = 0; b < products[a].pro_count; b++)
        {
            if (strcmp(products[a].LIST[b].month, "Oct") == 0)
                printf("%d  %s  %d  %f\n", products[a].prono, products[a].LIST
                  [b].month, products[a].LIST[b].volume, products[a].LIST
                  [b].discount);
        }
    }
    for (int a = 0; a < ROWS; a++) //fourth
    {
        for (int b = 0; b < products[a].pro_count; b++)
        {
            if (strcmp(products[a].LIST[b].month, "Nov") == 0)
                printf("%d  %s  %d  %f\n", products[a].prono, products[a].LIST
                  [b].month, products[a].LIST[b].volume, products[a].LIST
                  [b].discount);
        }
    }
    for (int a = 0; a < ROWS; a++) //fifth
    {
        for (int b = 0; b < products[a].pro_count; b++)
        {
            if (strcmp(products[a].LIST[b].month, "Dec") == 0)
                printf("%d  %s  %d  %f\n", products[a].prono, products[a].LIST
                  [b].month, products[a].LIST[b].volume, products[a].LIST
                  [b].discount);
        }
    }
}

/*
@function Sum_And_Average
@desc Calculate the total sale volume and average sale volume of each product
  respectively,
and output the them on screen.
*/
void Sum_And_Average() /*Question 4*/
{
    int sum[ROWS];
    float average[ROWS];
    for (int i = 0; i < ROWS; i++)
    {
        sum[i] = 0;
```

```c
        average[i] = 0;
    }
    for (int a = 0; a < ROWS; a++)
    {
        for (int i = 0; i < products[a].pro_count; i++)
        {
            sum[a] += products[a].LIST[i].volume;
        }
        average[a] = (float)sum[a] / products[a].pro_count;
    }
    printf("\n------------------------------\n");
    printf("||--The sum of each prono--||\n");
    printf("------------------------------\n");
    for (int i = 0; i < ROWS; i++)
    {
        printf("The sum of prono %d is %d\n", i+101, sum[i]);
    }
    printf("\n----------------------------------\n");
    printf("||--The average of each prono--||\n");
    printf("----------------------------------\n");
    for (int i = 0; i < ROWS; i++)
    {
        printf("The average of prono %d is %.3f\n", i+101, average[i]);
    }
}

/*
@function new_file_month
@desc Output all sales data of September with product information, write the result to a
new txt file and name the file with my Brunel ID.
*/
void new_file_month() /*Question 5*/
{
    printf("\n");
    FILE* fp3;
    if ((fp3 = fopen("2161047.txt", "w")) == NULL) {
        printf("File cannot be open!");
        exit(0);
    }
    printf("-------------------------------------------------------\n");
    printf("|| All salelist of each product by month September: ||\n");
    printf("-------------------------------------------------------\n");
    printf("prono      fullname      price      month     salevolume  discount   \n");
    fprintf(fp3, "prono      fullname      price      month     salevolume  discount   \n");

    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < products[i].pro_count; j++)
        {
            if (strcmp(products[i].LIST[j].month, "Sep") == 0)
            {
                printf("%-10d %-10s   %5.2f      ", products[i].prono, products[i].fullname, products[i].price);
                fprintf(fp3, "%-10d %-10s   %5.2f      ", products[i].prono, products
                    [i].fullname, products[i].price);
                printf("%5s   %5.2d         %5.2f\n", products[i].LIST[j].month, products[i].LIST[j].volume, products[i].LIST[j].discount);
                fprintf(fp3, "%5s   %5.2d         %5.2f\n", products[i].LIST[j].month, products[i].LIST[j].volume, products[i].LIST[j].discount);
            }
        }
    }
    fclose(fp3);
}

/*
@function cal_total_sale
@desc Calculate the total sale of each product, and output the top three on the screen.
*/
void cal_total_sale() /*Question 6*/
{
    printf("\nCalculate and output the total_sale.\n");
    struct product temp1;
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < products[i].pro_count; j++)
        {
            float sale = 0;
            sale = products[i].LIST[j].volume * products[i].price * (1 - products[i].LIST[j].discount);
            products[i].total_sale += sale;
        }
    }

    for (int i = 0; i < ROWS - 1; i++)
    {
        for (int j = 0; j < ROWS - 1 - j; j++)
        {
            if (products[j].total_sale < products[j + 1].total_sale)
            {
                temp1 = products[j];
                products[j] = products[j + 1];
                products[j + 1] = temp1;
            }
        }
    }

    printf("--------------------------------------\n");
    printf("|| The total sale of top three are: ||\n");
    printf("--------------------------------------\n");
    printf("NO. prono total_sale fullname\n");
    for (int i = 0; i < 3; i++)
    {
        printf("%d.  %d   %-8.3f   %s\n", i + 1, products[i].prono, products[i].total_sale, products[i].fullname);
    }
}

```