The Payroll Program Loop Revisited

```
cin >> empNum;
moreData = (empNum != 0);        // Initialize flag

while (moreData)
{
     .
     .
     .
     cin >> empNum;              // Get the next
                                 // employee number
     moreData = (empNum != 0);   // Update the flag
                                 // accordingly
}
```


Looping Subtasks

- Counting
- Summing
- Keeping Track of a Previous Value

Counting

```
count = 0;                  // Initialize counter
cin.get(inChar);            // Read the first character

while (inChar != '.')
{
     count++;               // Increment counter
     cin.get(inChar);       // Get the next character
}
```


Summing

```
sum = 0;
count = 1;                          // Initialize count

while (count <= 10)
{
     cin >> number;                 // Input a value
     sum = sum + number;            // Add the value to
     count++;                       // the sum
}
```


Another Summing Example

```
count = 0;                  // Initialize event counter
sum = 0;                    // Initialize sum
lessThanTen = true;         // Initialize flag

while (lessThanTen)
{
   cin >> number;           // Get the next value
   if (number % 2 == 1)     // Is the value odd?
   {
      count++;              // Yes – Increment counter
      sum = sum + number;           // Add value to sum
      lessThanTen = (count < 10);  // Update flag
   }
}
```


Recognizing Two Character Sequences

```
//*********************************************
// NotEqualCount program
// This program counts the occurrences of "!=" in a
// data file
//*********************************************

#include <iostream>
#include <fstream>                // For file I/O
using namespace std;

int main()
{
    int      count;     // Number of != operators
    char     prevChar;     // Last character read
```

Recognizing Two Character Sequences

```
char     currChar;     // Character read in this loop
                       // iteration
ifstream inFile;             // Data file
inFile.open("myfile.dat");   // Attempt to open
                             // input file
if ( !inFile )              // Was it opened?
{
   cout << "** Can't open input file **"  // No—print
                                          // message
        << endl;
   return 1;                          // Terminate
                                      // program
}
```


More Recognizing Two Character Sequences

```
count = 0;             // Initialize counter
inFile.get(prevChar);  // Initialize previous value
inFile.get(currChar);  // Initialize current value

while (inFile) // While previous input succeeded...
{
    if (currChar == '=' &&  // Test for event
        prevChar == '!')
        count++;            // Increment counter
    prevChar = currChar;    // Replace previous value
                            //   with current value
    inFile.get(currChar);   // Get next value
}
```


Recognizing Two Character Sequences (The End)

```
    cout << count << " != operators were found."
         << endl;
    return 0;
}
```


How to Design Loops

- Points to Consider
  - What is the condition that ends the loop?
  - How should the condition be initialized?
  - How should the condition be updated?
  - What is the process being repeated?
  - How should the process be initialized?
  - How should the process be updated?
  - What is the state of the program on exiting the loop?

Is This a Well-Formed Loop?

```
commaCount = 1;
cin.get(inChar);

while (inchar != '\n')
{
    if (inChar == ',')
        commaCount++;
    cin.get(inChar);
}
cout << commaCount << endl;
```


Is This a Well-Formed Loop? - No, So Fix It

```
commaCount = 0;
cin.get(inChar);

while (inchar != '\n')
{
    if (inChar == ',')
        commaCount++;
    cin.get(inChar);
}
cout << commaCount << endl;
```

Nested Logic

```
cin.get(inChar);           // Initialize outer loop
while (cin)                // Outer loop test
{
   commaCount = 0;         // Initialize inner loop
                           // (Priming read for outer
                           // loop also primes this one
   while (inChar != '\n')  // Inner loop test
   {
      if (inChar == ',')
         commaCount++;
      cin.get(inChar);  // Update inner loop condition
   }
   cout << commaCount << endl;
   cin.get(inChar); // Update outer loop condition
}
```


Another Nested Loop Example

```
cin >> starCount;
while (cin)
{
   loopCount = 1;
   while (loopCount <= starCount)
   {
      cout << '*';
      loopCount++;
   }
   cout << endl;
   cin >> starCount;
}
cout << "Goodbye" << endl;
```


Problem-Solving Case Study

- Problem: You are asked to compute the average salaries for male and female employees of a company.
- Input: A file, incFile (`ifstream`), of floating-point salary amounts (`float`), one per line, preceded by 'F' or 'M' (`char`).
- Output: All the input data. The number of females (`int`), average income (`float`). The number of males (`int`), average income (`float`)

Problem-Solving Case Study (Continued)

- Discussion: There are three main steps
  - Process the data, counting and summing the salary amounts for each sex
  - Compute the averages
  - Print the calculated results
- Assumptions: There is at least one male and one female among all the data sets. The only gender codes in the file are 'M' and 'F' – any other codes are counted as 'M'.


```
//*****************************************************
// Incomes program
// This program reads a file of income amounts
// classified by gender and computes the average
// income for each gender
//*****************************************************

#include <iostream>
#include <iomanip>    // For setprecision()
#include <fstream>    // For file I/O
#include <string>     // For string type

using namespace std;
```


Problem Solving Implementation

```
int main()
{
   char     sex;       // Coded 'F' = female, 'M' = male
   int      femaleCount; // Number of female income amounts
   int      maleCount;   // Number of male income amounts
   float    amount;      // Amount of income for a person
   float    femaleSum;   // Total of female income amounts
   float    maleSum;     // Total of male income amounts
   float    femaleAverage;    // Average female income
   float    maleAverage;      // Average male income
   ifstream incFile;          // File of income amounts
    string     fileName;         // External name of file
```

Problem Solving Implementation

```
cout << fixed << showpoint    // Set up floating pt.
     << setprecision(2);      //   output format

// Separately count females and males, and sum incomes

// Initialize ending condition
cout << "Name of the income data file: ";
cin >> fileName;
incFile.open(fileName.c_str());  // Open input file
                                 // and verify attempt
if ( !incFile )
{
   cout << "** Can't open input file **" << endl;
   return 1;
}
```



```
incFile >> sex >> amount;    // Perform priming read

femaleCount = 0;             // Initialize process
femaleSum = 0.0;
maleCount = 0;
maleSum = 0.0;

while (incFile)
{
    // Update process
    cout << "Sex: " << sex << " Amount: "
         << amount << endl;
```


Problem Solving Implementation

```
    if (sex == 'F')
    {
        femaleCount++;
        femaleSum = femaleSum + amount;
    }
    else
    {
        maleCount++;
        maleSum = maleSum + amount;
    }
    // Update ending condition
    incFile >> sex >> amount;
}
```


Problem Solving Implementation

```
// Compute average incomes
femaleAverage = femaleSum / float(femaleCount);
maleAverage = maleSum / float(maleCount);

// Output results

cout << "For " << femaleCount << " females, "
     << " the average income is " << femaleAverage
     << endl;
cout << "For " << maleCount << " males, the average "
     << "income is " << maleAverage << endl;
return 0;
}
```


One More `while` Loop

```
int main()
{
   ifstream inFile;
   ofstream outFile;
   inFile.open("prob4.in");
   outFile.open("prob4.out");
   char ch;
   inFile.get(ch);
   while(inFile)
   {
      if (ch != '\t')
         outFile.put(ch);
      infile.get(ch);
}
```