Computer Science 102
Lab 11

In this lab you will you will gain experience in

- creating and manipulating arrays of class instances
- controlling the format of C++ output streams
- building a sort routine capable of sorting the array on multiple keys.

We will use a simple account class to illustrate. Each account has an account number, a name, and an account balance as attributes.

```
class account_t
{
public:
   account_t(void);
   friend ostream & operator<<(ostream &out, const account_t &);
   friend istream & operator>>(istream &in, account_t &);

   friend void   sorttab(account_t *tab, int count);
   friend int    compnums(account_t &, account_t &);
   friend int    compnames(account_t &, account_t &);
   friend int    compbals(account_t &, account_t &);
   friend int    swap(account_t &, account_t &);

private:
    int    acctnum;
    char   name[20];
    double balance;
};
```

You will use the following main program. Do not implement the sort until the rest of the program is working.

```
#include "account.h"

int main()
{
   account_t accts[20];
   int i = 0;

   int count;

   while (cin >> accts[i])
      i = i + 1;

   count = i;

#if 0
   sorttab(&accts[0], compnums, count);
   printaccts(accts, count);
   cout << endl;


   sorttab(&accts[0], compnames, count);
   printaccts(accts, count);
   cout << endl;

    sorttab(&accts[0], compbals, count);
#endif

   printaccts(accts, count);
   cout << endl;

}
```

Step 1:

Build simple overloads of the >> and << operators that will read and print a complete account record. Build the printacct() function

```
void   printaccts(account_t *table, int count);
```

that will print use the overloaded << operator to print each record in the table with two spaces between the fields and a new line at the end of each record. The output should look something like the following:

```
23434  Westall  23.12
31003  Smith  1.2
31024  Taylor  12.3
10230  Jones  1345.3
302  Wilson  123.22
```

This output obviously has some significant deficiencies:

- Wilson's account number is 00302 but the leading zeros were lost causing the names to not line up.
- The variable length names cause the balances to not line up.
- The balances are printed with a variable number of digits to the right of the decimal.

The C++ stream output facility has some mechanisms that you can use to address these problems and make the output look like:

```
23434  Westall             23.12
31003  Smith               1.20
31024  Taylor             12.30
10230  Jones            1345.30
00302  Wilson            123.22
```

Before you can use them you must add *iomanip* to your include collection.

```
#include <iostream>
#include <iomanip>
using namespace std;
```

When you have done that you may embed the following specifications into a cout << cascade:

setw(n)	where *n* is a constant or variable specifying the width of the next field. e.g. setw(5) causes the next value printed to occupy a width of 5 bytes.
setfill(c)	where *c* is a constant or variable specifying the character that should be used to fill fields: e.g. setfill('0') can be used to deal with account numbers that start with leading 0's, but then setfill(' ') must be used to return to normal blank fill.
setiosflags(ios::right)	force the data to be right justified.
resetiosflags(ios::right)	disable forced right justification.
setiosflags(ios::left)	force the data to be left justified.
resetiosflags(ios::left)	disable forced left justification.
setprecision(n)	where *n* is a constant or variable specifying how many digits to the right of the decimal should be printed.

For example, to force a double precison value to be printed left justified in a field of 8 bytes with 3 digits to the right of the decimal and then disabled forced justification use:

```
out << setw(8) << setprecison(3) << setiosflags(ios::left) <<
          value << resetiosflags(ios::left) << endl;
```

Step 2:

Using the techniques of the previous page format the output as shown: The account number should be right justified in a five byte field using a fill character of '0'. The name should be left justified in a 16 byte field that is ' ' filled. The account balance should be right justified in a 10 byte field with 2 digits to the right of the decimal.

```
23434  Westall             23.12
31003  Smith               1.20
31024  Taylor             12.30
10230  Jones            1345.30
00302  Wilson            123.22
```

Step 3: Sorting

In CPSC 101 you saw how to make a single pass over an array of *n* elements swapping any adjacent pair that was out of order. A simple but not very efficient way to sort such an array is to just repeat this process *n* times. A nested loop structure is commonly used and in the example of CPSC 101 the result is:

```
     for (i = 0; i < count; i++)   // make count passes
     {
         for (j = 0; j < count - 1; j++) //  make a single pass
         {
             if (table[j] > table[j+1])
                   swap(table[j], table[j + 1])
         }
      }
```

In this exercise we will be sorting account records. We want to be able to use any field in the record as a sort key. A standard way to do this is to write "compare()" functions for each possible key and then just pass a pointer to the one you wish to use to the sort routine.

```
void   sorttab(account_t *tab, int (*comp)(account_t &, account_t &),
               int count);


int    compnums(account_t &, account_t &);
int    compnames(account_t &, account_t &);
int    compbals(account_t &, account_t &);
int    swap(account_t &, account_t &);
```

A pointer to the actual compare function to be used is passed in as a parameter

Here the main() function invokes the sort asking it to sort on names:

```
   sorttab(&accts[0], compnames, count);
```

A pointer to the compnames function is passed in as a parameter

Within the sort function, instead of directly comparing elements,

```
  if (table[j] > table[j+1])
```

the sorttab function asks correct compare function to decide whether or not to swap the elements:

```
 if ((*comp)(table[j], table[j + 1]) > 0)
```

When the sort is complete, remove the #if 0 in the main function and you should get:

```
00302  Wilson              123.22
10230  Jones              1345.30
23434  Westall              23.12
31003  Smith                 1.20
31024  Taylor               12.30

10230  Jones              1345.30
31003  Smith                 1.20
31024  Taylor               12.30
23434  Westall              23.12
00302  Wilson              123.22

31003  Smith                 1.20
31024  Taylor               12.30
23434  Westall              23.12
00302  Wilson              123.22
10230  Jones              1345.30
```

In this lab you will submit a single file, account.cpp that includes the specified functions.

sendlab.102.labsection# lab# account.cpp