Scalar Variables

When a variable is declared with the statement int num; memory is allocated for one integer. If several numbers are needed, they can be declared as int num1, num2, num3; The program below reads in three numbers, sorts them and prints them out in order:

```
 0 // Read in 3 numbers, sort, print
 1 #include<iostream.h>
 2 void swap(double&, double&);
 3 int main()
 4 { double num1, num2, num3;
 5   cout<<"Enter a number: ";
 6   cin>>num1;
 7   cout<<"Enter a number: ";
 8   cin>>num2;
 9   cout<<"Enter a number: ";
10   cin>>num3;
11   if (num1 > num2) swap(num1, num2);
12   if (num2 > num3) swap(num2,num3);
13   if (num1 > num2) swap(num1, num2);
14   cout<<"The numbers are: "<<num1<<" "<<num2<<" "<<num3<<"\n";
15   return 0;
16 } //main
17
18 void swap(double &n1, double &n2)
19 //receives n1 and n2 by reference, swaps them
20 { double temp;
21   temp=n1;
22   n1=n2;
23   n2=temp;
24 } //swap
```

Arrays

In the example above, num1, num2 and num3 are used in exactly the same way: all three are read in; the three numbers are sorted and the three numbers are printed out. When variables have the same function in a program, it is often more efficient to use an array. The program below does the same thing as the program above but uses an array. The statement double num[3]; allocates 3 memory for three decimal numbers. The three numbers are referred to as num[0], num[1], and num[2]. The number inside brackets is the subscript or index. The subscripts always start with 0. Once a variable is declared as an array, the subscript must be included to refer to the elements. One of the advantages of an array is that the subscript can be a variable. A **`for`** loop can be used to read in an array of numbers or print them out.

```
 0 // Read 3 numbers into an array, sort, print
 1 #include<iostream.h>
 2 void swap(double&, double&);
 3 int main()
 4 { double num[3];
 5   int n;
 6   for(n=0; n<3; n++)
 7   {  cout<<"Enter a number: ";
 8      cin>>num[n];
```

```
 9    } //n loop
10    if (num[0] > num[1]) swap(num[0], num[1]);
11    if (num[1] > num[2]) swap(num[1],num[2]);
12    if (num[0] > num[1]) swap(num[0], num[1]);
13    cout<<"The numbers are: "<<num[0]<<
14      " "<<num[1]<<" "<<num[2]<<"\n";
15    return 0;
16  } //main
17
18  void swap(double &n1, double &n2)
19  //receives n1 and n2 by reference, swaps them
20  { double temp;
21    temp=n1;
22    n1=n2;
23    n2=temp;
24  } //swap
```

The next example uses an array to store the number of days in each month. The user can enter the number of a month and find out how many days are in the month. The array is declare to have 13 elements. They will be number 0 to 12. We won't be using element 0, but it is easier to understand if January is element 1 and December is element 12.

```
 0  // Months: print name of month and number of days.
 1  #include<iostream.h>
 2  int main()
 3  { int days[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
 4    char month[13][10]={"","January", "February","March", "April",
 5    "May","June","July","August","September","October",
 6    "November", "December"};
 7    int mth; //to store user input
 8    cout<<"Enter the number of a month:";
 9    cin>>mth;
10    if (mth<1 || mth>12)
11       cout<<mth<<" is not a valid month.\n";
12    else
13       cout<<month[mth]<<" has "<<days[mth]<<" days.\n";
14    return 0;
15  } //main
```

Explanation:

3: An array can be given initial values. A something must be put in [0] in oreder for the other numbers to be in the right place.

4-6: An array of strings is a 2 dimensional array, it is easy to use an array with pre-defined values. The first dimension is the number of elements. The second dimension is the maximum length plus 1. Strings are discussed in more detail in the next lesson.

7: Not everything is an array! Sometimes students learn about arrays and start making everything an array.

10: It is the programmers responsibility to make sure that no attempt is made to use a subscript that is out of bounds.

Julian Date

The next program asks the user for a date and loops until a valid date is entered. It uses several functions: A function to get the date, a function to determine if it is leap year, and another function to calculate the Julian date. The Julian date is the day of the year: January 1st is the 1st day, February 1st is the 32nd day. The date is checked for valid year, month and day.

```
 0 // Finds Julian date for a valid month
 1 #include<iostream.h>
 2 void getDate(int& mth, int& day, int& year);
 3 bool validDate(int mth, int day, int year);
 4 int julian(int month, int day);  //finds day of year
 5 void getDate(int& mth, int& day, int& year);
 6 //Array is 13: 0..12, so Jan is [1] Dec. is [12], [0] not used
 7 int days[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
 8 int main()
 9 { int mth, day, year;
10   getDate(mth, day, year);
11   if (validDate(mth, day, year))
12     cout<<"Julian date="<<julian(mth,day)<<"\n";
13   else
14     cout<<"NOT a valid date\n";
15   return 0;
16 } //main
17
18 void getDate(int& mth, int& day, int& year)
19 { char slash;
20   cout<<"Enter a date as month/day/year:";
21   cin>>mth>>slash>>day>>slash>>year;
22 } //getDate
23
24 bool leapYear(int year) //returns true if leap year
25 { bool leap=true;
26   if (year%4) leap=false; //if year not div. by 4, not leap year
27   if ((year%100==0) && (year%400>0)) leap=false;
28   return leap;
29 } //leapyear
30 bool validDate(int mth, int day, int year)
31 { bool valid=true;
32   if (leapYear(year)) days[2]=29; else days[2]=28;
33   if (mth<1 || mth>12) valid=false;
34   else if(day<0 || day> days[mth]) valid=false;
35   return valid;
36 } //validate
37
38 int julian(int month, int day)  //finds day of year
39 { int total, m;
40   total=day; //May 21 is 21 + days in Jan. – April.
41   for (m=0;m<month-1;m++)
42         total+=days[m];
43   return total;
44 } // julian
```

If you would like to add a loop to read in many dates, only main needs to be changed:

```
 0 int main()
 1 { int mth, day, year;
 2   cout<<"Enter 0/0/0 to end loop\n";
```

```
  getDate(mth, day, year); //get first date
  while (mth>0)
  { if (validDate(mth, day, year))
      cout<<"Julian date="<<julian(mth,day)<<"\n";
    else
       cout<<"NOT valid\n";
    getDate(mth, day, year);//next date at end of loop
  } //while mth>0
  return 0;
} //main
```

Experiment: The program below is a menu-driven program that has just 1 choice! Brainstorm! Try adding each of the following functions:

- Raise all the prices
- Take an order for several items
- Check the price of an item
- What is the cheapest item?
- What is the most expensive?

```
// Fast food
#include<iostream.h>
char
food[9][7]={"","hotdog","fries","soda","cookie","pizza","burger",
"taco","shake"};
double price[9]={0,1.75,1.50,0.90,1.25,3.00,3.50,2.00,1.75};
void printAll(); //prints menu
void instructions(); //display choices
void process(char answer); //call correct function

void main()
{ char answer;
  instructions();
  cin>>answer;
  while ((answer != 'q') && (answer != 'Q'))
  { process(answer);
    instructions();
    cin>>answer;
  } //loop until they enter a Q
} //main

void instructions() //display choices
{ cout<<"C++ Fast Food\n";
  cout<<"Enter M to print the menu\n";
  cout<<"Enter Q to quit\n";
  cout<<"Your choice:";
} //instructions

void process(char answer) //call correct function
{  switch (answer)
   { case 'M': case 'm': printAll(); break;
     case 'Q': case 'q': break; //don't go to default
     default: cout<<"Invalid choice\n";
```

```
33     } //switch
34  } //process
35
36  void printAll()
37  { cout<<"Item\tFood\tPrice\n";
38    cout.precision(2);
39    cout.setf(ios::fixed);
40    for (int i=1; i<9; i++)
41      cout<<i<<"\t"<<food[i]<<"\t"<<price[i]<<"\n";
    }//printAll
```