Practice!

Give assignment statements for calculating the following values, given the value of x (assume that the value of x is in the proper range of values for the calculations).

1. coth x	2. sech 3 x
3. csch 4 x	4. acots 6 x
5. acosh x	6. acsch x

Character Functions

The Standard C++ library contains many pre-defined functions for use with character data. These functions fall into two categories: one set of functions is used to convert characters between uppercase and lowercase, and the other set is used to perform character comparisons. The following preprocessor directive should be used in programs referencing these character functions:

```
#include <cctype>
```

The following statement converts the lowercase letter stored in the object `ch` to an uppercase character and stores the result in the character object `ch_upper`:

```
ch_upper = toupper(ch);
```

If `ch` is a lowercase letter, the function `toupper()` returns the corresponding uppercase letter; otherwise, the function returns `ch`. **Note that no change is made to the object** `ch`.

The character comparison functions return a nonzero value if the comparison is true; otherwise they return zero. The following statement calls the function `isdigit()`. The function `isdigit()` will return a nonzero result if the value of `ch` is a digit (0–9) or the value 0 (false) if `ch` is not a digit:

```
digit = isdigit(ch);
```

A list of these functions along with a brief explanation is given in Appendix A.

2.8 Problem Solving Applied: Velocity Computation

In this section, we perform computations in an application related to vehicle performance. Turboprop engines, which have been in use for decades, combine the power and reliability of jet engines with the efficiency of propellers. They are a significant improvement over earlier piston-powered propeller engines. Their application has been limited to smaller commuter-type aircraft, however, because they are not as fast or powerful as the fanjet engines used on larger airliners. The unducted fan (UDF) engine employs significant advancements in propeller technology, which narrow the performance gap between turboprops and fanjets. New materials, blade shapes, and higher rotation speeds enable UDF-powered aircraft to fly almost as fast as fanjets, and with greater fuel efficiency. The UDF is also significantly quieter than the conventional turboprop.

During a test flight of a UDF-powered aircraft, the test pilot has set the engine power level at 40,000 N (newtons), which causes the 20,000-kg aircraft to attain a cruise speed of 180 m/s (meters/second). The engine throttles are then set to a power level of 60,000 N and the aircraft begins to accelerate. As the speed of the plane increases, the aerodynamic drag increases in proportion to the square of the airspeed. Eventually, the aircraft reaches a new cruise speed where the thrust from the UDF engines is just offset by the drag. The equations used to estimate the velocity and acceleration of the aircraft from the time that the throttle is reset until the plane reaches its new cruise speed (at approximately 120 s) are the following:

$$\text{Velocity} = 0.00001 * \text{time}^3 - 0.00488 * \text{time}^2 + 0.75795 * \text{time} + 181.3566;$$

$$\text{Acceleration} = 3 - 0.000062 * \text{velocity}^2.$$

Plots of these functions are shown in Figure 2.8. Note that the acceleration approaches zero as the velocity approaches its new cruise speed.

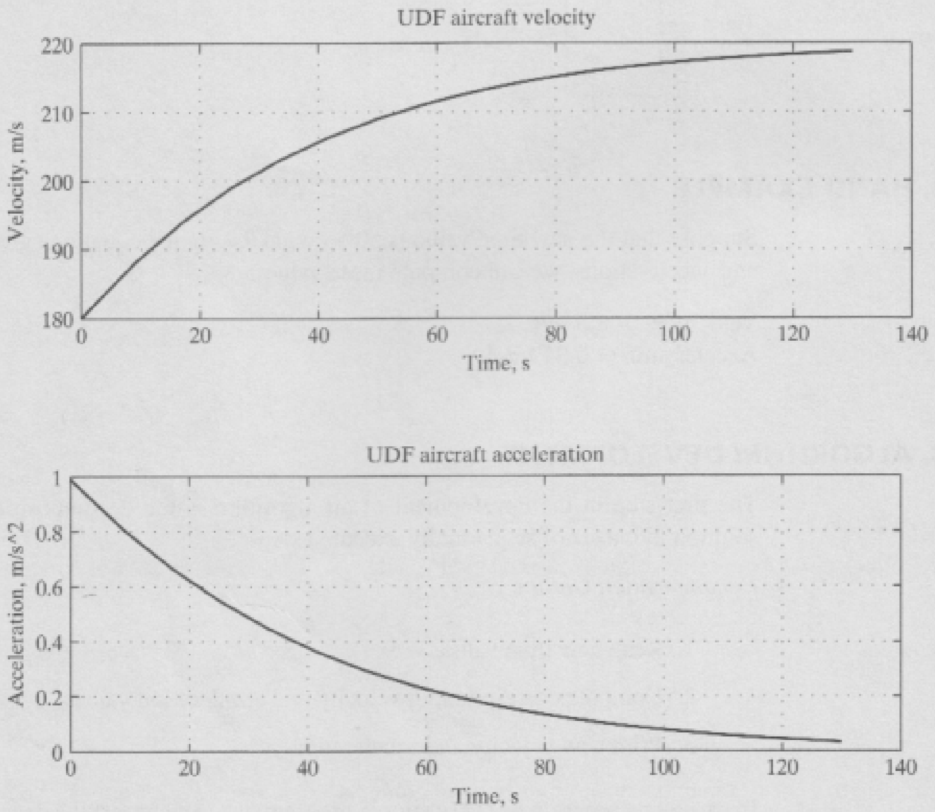


Figure 2.8 UDF aircraft velocity and acceleration.

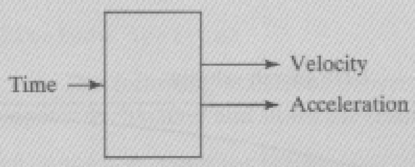
Write a program that asks the user to enter a time value that represents the time elapsed (in seconds) since the power level was increased. Compute and print the corresponding acceleration and velocity of the aircraft at the new time value.

1. PROBLEM STATEMENT

Compute the new velocity and acceleration of the aircraft after a change in power level.

2. INPUT/OUTPUT DESCRIPTION

The following diagram shows that the input to the program is a time value, and that the output of the program is the pair of new velocity and acceleration values. The built-in data type `double` can be used to represent these values.



3. HAND EXAMPLE

Suppose that the new time value is 50 seconds. Using the equations given for the velocity and accelerations, we can compute these values:

Velocity = 208.3 m/s;
Acceleration = 0.31 m/s^2.

4. ALGORITHM DEVELOPMENT

The first step in the development of an algorithm is the decomposition of the problem solution into a set of sequentially executed steps:

Decomposition Outline

1. Read new time value.
2. Compute corresponding velocity and acceleration values.
3. Print new velocity and acceleration.

Because this program is a very simple program, we can convert the decomposition directly to C++.

ts the time elapsed
rresponding accel-

ange in power

e, and that the
he built-in data

r the velocity

the problem

tion directly

```
/*--------------------------------------------------------*/
/*  Program chapter2_6                                    */
/*                                                        */
/*  This program estimates new velocity and               */
/*  acceleration values for a specified time.             */

#include<iostream>  //Required for cin,cout
#include<iomanip>   //Required for setprecision(), setw()
#include<cmath>     //Required for pow()
using namespace std;

int main()
{
   // Declare objects.
   double time, velocity, acceleration;

   // Get time value from the keyboard.
   cout << "Enter new time value in seconds: \n";
   cin >> time;

   // Compute velocity and acceleration.
   velocity = 0.00001*pow(time,3) - 0.00488*pow(time,2)
              + 0.75795*time + 181.3566;
   acceleration = 3 - 0.000062*velocity*velocity;
   // Print velocity and acceleration.
   cout << fixed << setprecision(3);
   cout << "Velocity = " << setw(10)
        << velocity << " m/s" << endl;
   cout << "Acceleration = " << setw( 14)
        << acceleration << "m/s^2" << endl;

   // Exit program.
   return 0;
}
/*--------------------------------------------------------*/
```

5. TESTING

We first test the program using the data from the hand example. This generates the following interaction:

```
Enter new time value in seconds:
50
Velocity = 208.304 m/s
Acceleration =   0.310 m/s^2
```

Because the values computed match the hand example, we can then test the program with other time values. If the values had not matched the hand example, we would need to determine whether the error is in the hand example or in the program.

Modify!

These problems relate to the program developed in this section for computing velocity and acceleration values.

1. Enter different values of time until you find one that gives a velocity between 210 m/s and 211 m/s.
2. Enter different values of time until you find one that gives an acceleration between 0.5 m/s and 0.6 m/s.
3. Modify the program so that the input values are entered in minutes instead of seconds. Remember that the equations will still assume that the time values are in seconds.
4. Modify the program so that the output values are printed in feet per second, and feet per second^2. (Recall that 1 meter = 39.37 inches.)

2.9 System Limitations

In Section 2.2, we presented a table that contained the maximum values for the various types of integers and floating-point values for the Microsoft Visual C++ 6.0 compiler. To print a similar table for your system, use the program presented next. Note that the program includes three header files. The `iostream` header file is necessary because the program uses `cout`; the `climits` header file is necessary because it contains information relative to the ranges of integer types; and the `cfloat` header file is necessary because it contains information relative to the ranges of floating-point types. Appendix A contains more information on the constants and limits that are system dependent.

```
/*--------------------------------------------------*/
/*  Program chapter2_7                              */
/*                                                  */
/*  This program prints the system limitations.     */

#include<iostream>
#include<climits>
#include<cfloat>
using namespace std;
int main()
{
   //  Print integer type maxima. /
   cout << "short maximum: " << sizeof(short) << endl;
   cout << "int maximum: " << sizeof(int) << endl;
   cout << "long maximum: " << sizeof(long) << endl << endl;
```