The goal of this problem set is to extend the solution of tutorial 3. In particular, we wish to add methods to calculate a polynomial and to determine a polynomial's indefinite and definite integral.

Start with the solution of tutorial 3 (see Blackboard). Do not edit the provided files. Rather create a new .cpp unit, called `PolynomialPS1.cpp`, to implement the new methods. Please note that in C++ the implementation of a class does not need to occur in just one compilation unit (i.e., a .cpp file). As long as the compiler and the linker see all definitions, the build process should succeed, if the program does not contain any errors.

Remember, a polynomial can be expressed concisely by using summation notation:

$$\sum_{i=0}^{n} a_i x^i$$

That is, a polynomial can be written as the sum of a finite number of terms $a_i x^i$. Each term consists of the product of a number a_i, called the coefficient, and a variable x raised to integer powers – x^i. The exponent i in x^i is called the degree of the term $a_i x^i$. The degree of a polynomial is the largest degree of any one term with a non-zero coefficient.

To calculate a polynomial, we need to provide a value for the variable x. Mathematically, we can write

$$f(x) = \sum_{i=0}^{n} a_i x^i$$

to denote the result of calculating a polynomial for a given value of x. Again, this allows for a straightforward mapping to a for-loop is C++. To compute the x^i, we need to use the function `pow`, called *raise-to-power*, which is defined in `cmath`. For more details and example uses of `pow`, see http://www.cplusplus.com/reference/cmath/pow/.

Polynomials appear in many areas of mathematics and science. They are used, for example, in calculus, numerical analysis, and algebraic geometry. We can use polynomials to describe non-regular shaped objects and to determine, for instance, their volume.

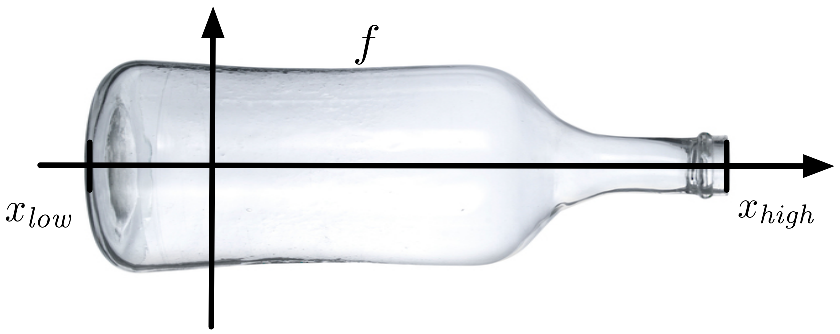


Figure 1: A curvy bottle.

To compute the volume of an object whose shape is given by a polynomial, we need to compute its *area under the curve*. There exist numerical methods to approximate the area. However, the most precise way of finding the area is to calculate the integral of the polynomial.

The *indefinite integral* of a polynomial is the sum of the integrals of its terms. A general term of a polynomial can written ax^n, read the term at nth degree, and the indefinite integral of that term is

$$\int ax^n \, dx = \frac{a}{n+1} x^{n+1}$$

In other words, we obtain the indefinite integral of the nth degree polynomial term by raising it to the power $n+1$ and dividing it by $n+1$. Using the summation notation, we have

$$\int \sum_{i=0}^{n} a_i x^n \, dx = \sum_{i=0}^{n} \frac{a_i}{i+1} x^{i+1}$$

A key observation here is that the terms in the indefinite integral move one position to the right – the next higher degree. The term at index 0 becomes 0. For example,

$$\int -0.25x + 4.0 \, dx = -0.125x^2 + 4.0x^1 + 0$$

To implement indefinite integrals for polynomials, the corresponding method has to create a new `Polynomial` object, assign to it `fDegree+1`, and run a for-loop from `fDegree` to `0` including (a top-down-loop) to set the polynomial coefficients according to the above rule.

The *definite integral* of a polynomial is a function with two arguments: x_{low} and x_{high} – the limits of the integral. We can write, without loss of generality,

$$\int_{x_{low}}^{x_{high}} \sum_{i=0}^{n} a_i x^n \, dx = \sum_{i=0}^{n} \frac{a_i}{i+1} {x_{high}}^{i+1} - \sum_{i=0}^{n} \frac{a_i}{i+1} {x_{low}}^{i+1}$$

That is, to compute the definite integral of a polynomial between x_{low} and x_{high}, we need to construct its indefinite integral first, calculate it for x_{low} and x_{high}, and subtract the result for x_{low} from that obtained for x_{high}.

The extended specification of class Polynomial is shown below. You only need to implement the last three methods. The other features (i.e., constructor and operators) are given as part of the solution for tutorial 3. In the .cpp file for the new methods you need to include `cmath` that contains the definition of `pow` – raise to power.

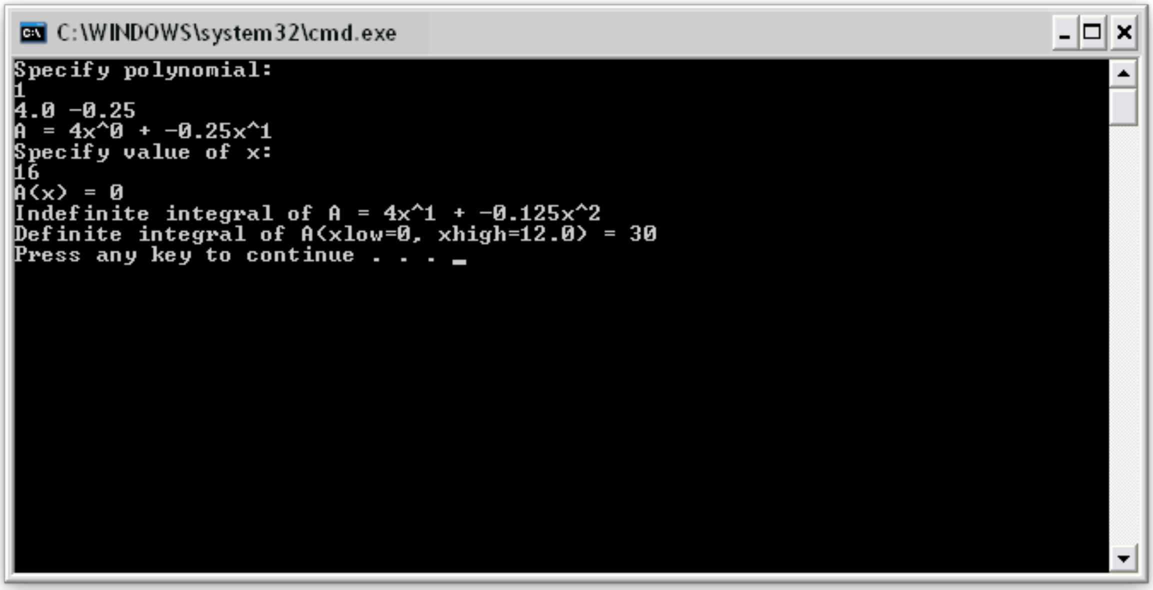
```
#pragma once

#include <iostream>

#define MAX_DEGREE 20+1+1     // max degree 0 to 20 plus 1 for integral

class Polynomial
{
private:
  int fDegree;                // the maximum degree of the polynomial
  double fCoeffs[MAX_DEGREE]; // the coefficients (0..10, 0..20, 0..21)

public:

  // tutorial 3 interface

  // the default constructor (initializes all member variables)
  Polynomial();

  // binary operator* to multiple to polynomials
  // arguments are read-only, signified by const
  // the operator* returns a fresh polynomial with degree i+j
  Polynomial operator*( const Polynomial& aRight ) const;

  // input operator for polynomials
  friend std::istream& operator>>( std::istream& aIStream,
                                   Polynomial& aObject );

  // output operator for polynomials
  friend std::ostream& operator<<( std::ostream& aOStream,
                                   const Polynomial& aObject );

  // new methods in problem set 1

  // calculate polynomial for a given x (i.e., parameter aX)
  double calculate( double aX ) const;

  // build indefinite integral
  // the indefinite integral is a fresh polynomial with degree fDegree+1
  // the method does not change the current object
  Polynomial buildIndefiniteIntegral() const;

  // build definite integral
  // the method does not change the current object
  // the method computes the indefinite integral and then calculates it
  // for xlow and xhigh and returns the difference
  double buildDefiniteIntegral( double aXLow, double aXHigh ) const;
};
```

Please note the changed value of `MAX_DEGREE`. Since we wish to build integrals, we need to reserve one more slot it the array for the coefficients of a polynomial. Our program supports polynomials up to the 10th degree. Multiplying them results in a polynomial up to the 20th degree, whose integral is up to the 21st degree. Hence, we need 22 entries in the array of coefficients (21+1).

Use as main program the following code:

```cpp
#include <iostream>

#include "Polynomial.h"

using namespace std;

int main()
{
  Polynomial A;
  cout << "Specify polynomial:" << endl;
  cin >> A;
  cout << "A = " << A << endl;

  double x;
  cout << "Specify value of x:" << endl;
  cin >> x;

  cout << "A(x) = " << A.calculate( x ) << endl;

  cout << "Indefinite integral of A = "
       << A.buildIndefiniteIntegral() << endl;

  cout << "Definite integral of A(xlow=0, xhigh=12.0) = "
       << A.buildDefiniteIntegral( 0, 12.0 ) << endl;

  return 0;
}
```

Using `-0.25x + 4.0` and `16` for the first calculation, the test code should produce the following result.



```
Specify polynomial:
1
4.0 -0.25
A = 4x^0 + -0.25x^1
Specify value of x:
16
A(x) = 0
Indefinite integral of A = 4x^1 + -0.125x^2
Definite integral of A(xlow=0, xhigh=12.0) = 30
Press any key to continue . . . _
```

The solution requires 20-40 lines of low density C++ code.