NUS WebMail IVLE LIBRARY MAPS

Search search for... in NUS Websites GO

CodeCrunch

Home | My Courses | Browse Tutorials | Browse Tasks | Search | My Submissions | Logout

CS1010E 16/17S1 Trapezoidal Rule (Question)

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Trapezoidal Rule

Topic Coverage

- Assignment and expressions
- Nested control statements
- Functions and procedures

Problem Description

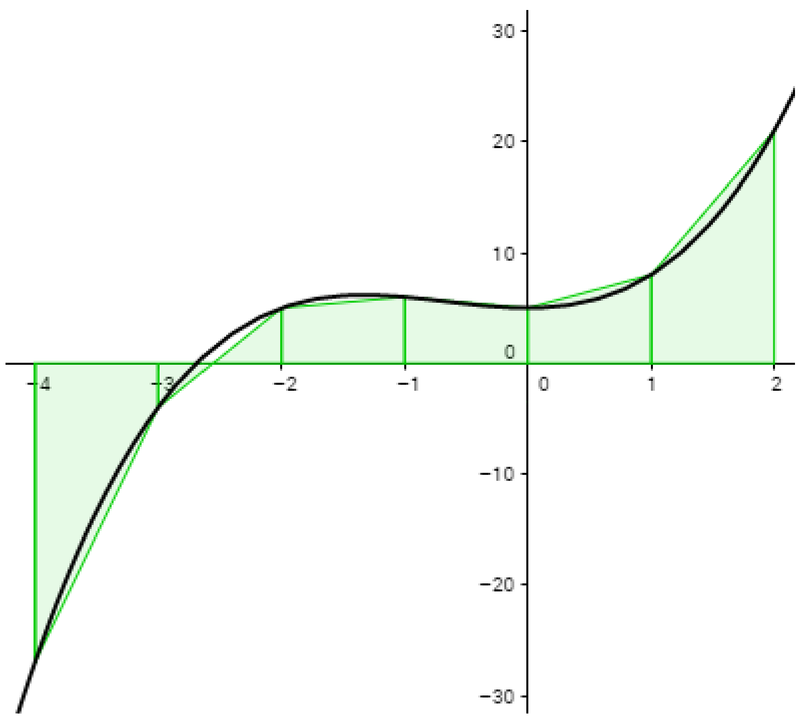
Given any polynomial, the integral of the polynomial is simply the sum of integrals of its terms.

Each term of the polynomial, written as cx^n, has an associated indefinite integral of the form $cx^{n+1}/(n+1) + k$ where `k` is a constant.

Using the example of $p(x) = x^3 + 2x^2 + 5$, the indefinite integral of `p(x)` is $P(x) = x^4/4 + 2x^3/3 + 5x + C$ where `C` is a constant.

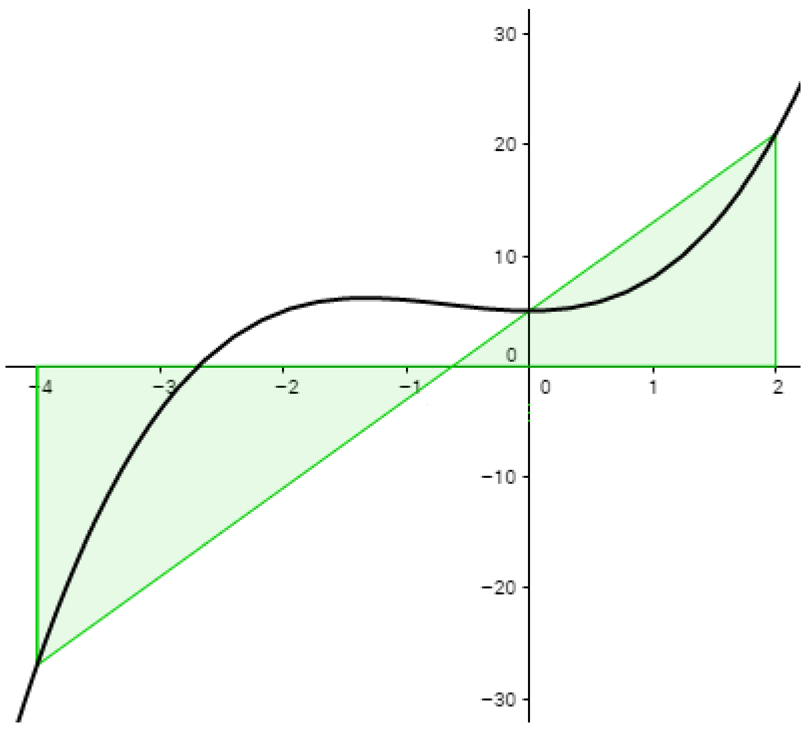
The **definite integral** of `p(x)` over the limits of the integral `[a,b]` is given by `P(b)-P(a)`. Notice that the constant C has been removed via the subtraction. In particular, using the limits `[-4,2]` the definite integral is `P(2)-P(-4) = 18`.

The definite integral can also be approximated using the trapezoidal rule as shown below.



Notice that the interval is fitted with `n` trapezoids (in the above, `n = 6`) within the limits `x=[-4,2]`. Each trapezoid is of equal width `h = (b-a)/n` and the area of each trapezoid can be computed as $(h/2)(p(a_i)+p(b_i))$ where a_i and b_i are limits with respect to the i^{th} trapezoid.

The sum of all trapezoids gives an approximation of the definite integral. Clearly the more trapezoids used, the more accurate is the approximation. As an another example, here is an approximation using only one trapezoid.



In this case, the area of the trapezoid is `(6/2)(p(-4)+p(2)) = -18`

Task

Write a program that asks the user to enter the integer coefficients (c_3, c_2, c_1, c_0) for a polynomial of degree 3:

$c_3x^3 + c_2x^2 + c_1x + c_0$.

It then asks for the interval limits `[a,b]` which are real numbers.

The program computes the exact value of the definite integral using the formula above and proceeds to approximate the definite integral using the trapezoidal rule starting with one trapezoid, then two, then three, etc. The approximation stops when the approximation is close to the actual value given by

| $P_{approx}(x)$ - P(x) | ≤ 0.001

Take note of the following:

- Assume that the user enters a polynomial function of at most degree 3.
- Use the **double** data types for real numbers.
- You may use the `pow` and `fabs` C Math library functions.
  `double pow(double x, double y);`
  Returns x raised to the power of y.

  `double fabs(double x);`
  Returns the absolute value of x (a negative value becomes positive, positive value is unchanged).

This task is divided into several levels. Read through all the levels (from first to last, then from last to first) to see how the different levels are related. **You may start from any level.**

Within each level, sample runs are provided for you to test your program. These sample runs represent the corner test cases for small inputs. You are encouraged to expand on these and come up with your own test cases.

Level 1

Name your program `trapz1.c`

Write a program that reads in four integer coefficients (c_3, c_2, c_1, c_0) followed by two floating-point values a and b representing limits `[a, b]`. Output the values of the coefficients and limits.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-4.0 2.0
1 2 0 5
a = -4.000000; b = 2.000000
```

Click here to submit to CodeCrunch.

Check the correctness of the output by typing the following Unix command

```
./a.out < trapz.in | diff - trapz1.out
```

To proceed to the next level (say level 2), copy your program by typing the Unix command

```
cp trapz1.c trapz2.c
```

Level 2

Name your program `trapz2.c`

Write a program that reads in four integer coefficients (c_3, c_2, c_1, c_0) followed by two floating-point values a and b representing limits `[a, b]`. Output the values of the coefficients, the limits as well as the values of the polynomial function evaluated at the two endpoints `p(a)` and `p(b)`.

You are encouraged to define the following function:

`double polynomial(int c3, int c2, int c1, int c0, double x);`
Returns the value of the polynomial evaluated at x.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-4.0 2.0
1 2 0 5
a = -4.000000; b = 2.000000
pa = -27.000000; pb = 21.000000
```

Click here to submit to CodeCrunch.

Check the correctness of the output by typing the following Unix command

```
./a.out < trapz.in | diff - trapz2.out
```

To proceed to the next level (say level 3), copy your program by typing the Unix command

```
cp trapz2.c trapz3.c
```

Level 3

Name your program `trapz3.c`

Write a program that reads in four integer coefficients (c_3, c_2, c_1, c_0) followed by two floating-point values a and b representing limits `[a, b]`. Output the values of the coefficients, the limits as well as the values of the polynomial function evaluated at the two endpoints `p(a)` and `p(b)`. In addition, output the exact value of the definite integral using the formula.

You are encouraged to define the following function:

`double polynomial(int c3, int c2, int c1, int c0, double x);`
Returns the value of the polynomial evaluated at x.

`double integralPoly(int c3, int c2, int c1, int c0, double x);`
Returns the value of the integral polynomial evaluated at x, ignoring any constant.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-4.0 2.0
1 2 0 5
```

```
a = -4.000000; b = 2.000000
pa = -27.000000; pb = 21.000000
exact = 18.000000
```

```
$ ./a.out
0 1 0 0
-2.0 2.0
0 1 0 0
a = -2.000000; b = 2.000000
pa = 4.000000; pb = 4.000000
exact = 5.333333
```

```
$ ./a.out
0 0 1 0
-2.0 2.0
0 0 1 0
a = -2.000000; b = 2.000000
pa = -2.000000; pb = 2.000000
exact = 0.000000
```

```
$ ./a.out
0 0 0 1
-2.0 2.0
0 0 0 1
a = -2.000000; b = 2.000000
pa = 1.000000; pb = 1.000000
exact = 4.000000
```

Click here to submit to CodeCrunch.

Check the correctness of the output by typing the following Unix command

```
./a.out < trapz.in | diff - trapz3.out
```

To proceed to the next level (say level 4), copy your program by typing the Unix command

```
cp trapz3.c trapz4.c
```

Level 4

Name your program `trapz4.c`

Write a program that reads in four integer coefficients (c_3, c_2, c_1, c_0) followed by two floating-point values a and b representing limits `[a, b]`. Output the values of the coefficients, the limits, as well as the exact value of the definite integral using the formula.

Output the area of **one** large trapezoid spanning the entire interval limits. This is the approximation of the definite integral using one trapezoid.

You are encouraged to define the following function:

`double polynomial(int c3, int c2, int c1, int c0, double x);`
Returns the value of the polynomial evaluated at x.

`double integralPoly(int c3, int c2, int c1, int c0, double x);`
Returns the value of the integral polynomial evaluated at x, ignoring any constant.

`double trapz(int c3, int c2, int c1, int c0, double a, double b, int n);`
Returns the trapezoidal rule approximation of the definite integral using n trapezoids.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-4.0 2.0
1 2 0 5
a = -4.000000; b = 2.000000
exact = 18.000000
approx = -18.000000
```

```
$ ./a.out
0 1 0 0
-2.0 2.0
0 1 0 0
a = -2.000000; b = 2.000000
exact = 5.333333
approx = 16.000000
```

```
$ ./a.out
0 0 1 0
-2.0 2.0
0 0 1 0
a = -2.000000; b = 2.000000
exact = 0.000000
approx = 0.000000
```

```
$ ./a.out
0 0 0 1
-2.0 2.0
0 0 0 1
a = -2.000000; b = 2.000000
exact = 4.000000
approx = 4.000000
```

Click here to submit to CodeCrunch.

Check the correctness of the output by typing the following Unix command

```
./a.out < trapz.in | diff - trapz4.out
```

To proceed to the next level (say level 5), copy your program by typing the Unix command

```
cp trapz4.c trapz5.c
```

Level 5

Name your program `trapz5.c`

Write a program that reads in four integer coefficients (c_3, c_2, c_1, c_0) followed by two floating-point values a and b representing limits `[a, b]`. Output the values of the coefficients, the limits, as well as the exact value of the definite integral using the formula.

Using **exactly five** trapezoids spanning the entire interval limits, output the individual sub-intervals and areas of the trapezoids.

You are encouraged to define the following function:

`double polynomial(int c3, int c2, int c1, int c0, double x);`
Returns the value of the polynomial evaluated at x.

`double integralPoly(int c3, int c2, int c1, int c0, double x);`
Returns the value of the integral polynomial evaluated at x, ignoring any constant.

`double trapz(int c3, int c2, int c1, int c0, double a, double b, int n);`
Returns the trapezoidal rule approximation of the definite integral using n trapezoids.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-4.0 2.0
1 2 0 5
a = -4.000000; b = 2.000000
exact = 18.000000
area within [-4.000000, -2.800000] = -16.963200
area within [-2.800000, -1.600000] = 2.851200
area within [-1.600000, -0.400000] = 6.768000
area within [-0.400000, 0.800000] = 7.228800
area within [0.800000, 2.000000] = 16.675200
```

```
$ ./a.out
0 1 0 0
-2.0 2.0
0 1 0 0
a = -2.000000; b = 2.000000
exact = 5.333333
area within [-2.000000, -1.200000] = 2.176000
area within [-1.200000, -0.400000] = 0.640000
area within [-0.400000, 0.400000] = 0.128000
area within [0.400000, 1.200000] = 0.640000
area within [1.200000, 2.000000] = 2.176000
```

```
$ ./a.out
0 0 1 0
-2.0 2.0
0 0 1 0
a = -2.000000; b = 2.000000
exact = 0.000000
area within [-2.000000, -1.200000] = -1.280000
area within [-1.200000, -0.400000] = -0.640000
area within [-0.400000, 0.400000] = 0.000000
area within [0.400000, 1.200000] = 0.640000
area within [1.200000, 2.000000] = 1.280000
```

```
$ ./a.out
0 0 0 1
-2.0 2.0
0 0 0 1
a = -2.000000; b = 2.000000
exact = 4.000000
area within [-2.000000, -1.200000] = 0.800000
area within [-1.200000, -0.400000] = 0.800000
area within [-0.400000, 0.400000] = 0.800000
area within [0.400000, 1.200000] = 0.800000
area within [1.200000, 2.000000] = 0.800000
```

Click here to submit to CodeCrunch.

Check the correctness of the output by typing the following Unix command

```
./a.out < trapz.in | diff - trapz5.out
```

To proceed to the next level (say level 6), copy your program by typing the Unix command

```
cp trapz5.c trapz6.c
```

Level 6

Name your program `trapz6.c`

Write a program that reads in four integer coefficients (c_3, c_2, c_1, c_0) followed by two floating-point values a and b representing limits `[a, b]`. Output the values of the coefficients, the limits, as well as the exact value of the definite integral using the formula.

Using **exactly five** trapezoids spanning the entire interval limits, output the total area of the trapezoids.

You are encouraged to define the following function:

`double polynomial(int c3, int c2, int c1, int c0, double x);`
Returns the value of the polynomial evaluated at x.

`double integralPoly(int c3, int c2, int c1, int c0, double x);`
Returns the value of the integral polynomial evaluated at x, ignoring any constant.

`double trapz(int c3, int c2, int c1, int c0, double a, double b, int n);`
Returns the trapezoidal rule approximation of the definite integral using n trapezoids.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-4.0 2.0
```

```
1 2 0 5
a = -4.000000; b = 2.000000
exact = 18.000000
approx = 16.560000
```

```
$ ./a.out
0 1 0 0
-2.0 2.0
0 1 0 0
a = -2.000000; b = 2.000000
exact = 5.333333
approx = 5.760000
```

```
$ ./a.out
0 0 1 0
-2.0 2.0
0 0 1 0
a = -2.000000; b = 2.000000
exact = 0.000000
approx = 0.000000
```

```
$ ./a.out
0 0 0 1
-2.0 2.0
0 0 0 1
a = -2.000000; b = 2.000000
exact = 4.000000
approx = 4.000000
```

Click here to submit to CodeCrunch.

Check the correctness of the output by typing the following Unix command

```
./a.out < trapz.in | diff - trapz6.out
```

To proceed to the next level (say level 7), copy your program by typing the Unix command

```
cp trapz6.c trapz7.c
```

Level 7

Name your program `trapz7.c`

Write a program that reads in four integer coefficients (c_3, c_2, c_1, c_0) followed by two floating-point values a and b representing limits `[a, b]`. Output the values of the coefficients, the limits, as well as the exact value of the definite integral using the formula.

Output the number of trapezoids required to give an approximation of the definite integral that meets the following criteria:

$| P_{approx}(x) - P(x) | \le 0.001$

You are encouraged to define the following function:

`double polynomial(int c3, int c2, int c1, int c0, double x);`
Returns the value of the polynomial evaluated at x.

`double integralPoly(int c3, int c2, int c1, int c0, double x);`
Returns the value of the integral polynomial evaluated at x, ignoring any constant.

`double trapz(int c3, int c2, int c1, int c0, double a, double b, int n);`
Returns the trapezoidal rule approximation of the definite integral using n trapezoids.

The following is a sample run of the program. User input is underlined. Ensure that the last line of output is followed by a newline character.

```
$ ./a.out
1 2 0 5
-4.0 2.0
1 2 0 5
a = -4.000000; b = 2.000000
exact = 18.000000
approx using 190 trapz = 17.999003
```

```
$ ./a.out
0 1 0 0
-2.0 2.0
0 1 0 0
a = -2.000000; b = 2.000000
exact = 5.333333
approx using 104 trapz = 5.334320
```

```
$ ./a.out
0 0 1 0
-2.0 2.0
0 0 1 0
a = -2.000000; b = 2.000000
exact = 0.000000
approx using 1 trapz = 0.000000
```

```
$ ./a.out
0 0 0 1
-2.0 2.0
0 0 0 1
a = -2.000000; b = 2.000000
exact = 4.000000
approx using 1 trapz = 4.000000
```

```
$ ./a.out
1 2 0 5
-2.0 -1.0
1 2 0 5
a = -2.000000; b = -1.000000
exact = 5.916667
approx using 21 trapz = 5.915722
```

```
$ ./a.out
1 2 0 5
1.0 2.0
1 2 0 5
a = 1.000000; b = 2.000000
```

```
exact = 13.416667
approx using 33 trapz = 13.417661
```

```
$ ./a.out
1 2 0 5
-4.0 -3.0
1 2 0 5
a = -4.000000; b = -3.000000
exact = -14.083333
approx using 38 trapz = -14.084314
```

Click here to submit to CodeCrunch.

Check the correctness of the output by typing the following Unix command

```
./a.out < trapz.in | diff - trapz7.out
```