Enumerated Type Example

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

enum Work_Days {Monday, Tuesday, Wednesday, Thursday, Friday};

Work_Days Convert(string day);

int main()
{
   string day_string;
   Work_Days day;
   cout << "Please enter a day of the work week (Monday
           through Friday): " " << endl;
   cin >> day_string;
   day = Convert_work_days(day_string);
   cout << "The day is " << Convert_string(day) << endl;
}
```


Enumerated Type Example

```
Work_Days Convert_work_days(string day)
{
   Work_Days result;

   switch(toupper(day[0]))
   {
      case 'M': result = Monday;
                break;
      case 'T': switch(toupper(day[1]))
                {
                   case 'U': result = Tuesday; break;
                   case 'H': result = Thursday; break;
                }
                break;
      case 'W': result = Wednesday;
                break;
      case 'F': result = Friday;
                break;
   }
   return result;
}
```


Enumerated Type Example

```
string Convert_string(Work_Days day)
{
   string result;

   switch( day)
   {
      case Monday    : result = "Monday";
                       break;
      case Tuesday   : result = "Tuesday";
                       break;
      case Wednesday : result = "Wednesday";
                       break;
      case Thursday  : result = "Thursday";
                       break;
      case Friday    : result = "Friday";
                       break;
   }
   return result;
}
```

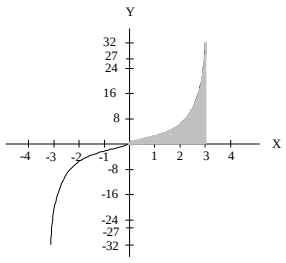

Problem-Solving Case Study: Finding the Area Under a Curve

- Problem: Find the area under the curve of the function x^3 over an interval specified by the user.
- Input: Two floating-point numbers specifying the interval over which to find the area, and an integer number of intervals to use in approximating the area.

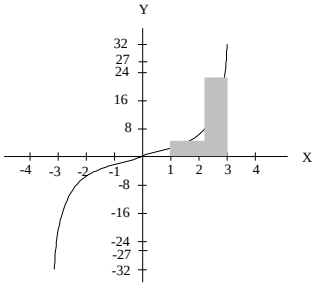
Problem-Solving Case Study: Finding the Area Under a Curve

- Output: The input data (echo print) and the value calculated for the area over the given interval.
- Discussion: Our approach is to compute an approximation by dividing the area into equal width rectangular strips. Smaller widths give a better answer. We can use a value-returning function to compute the area of each rectangle.

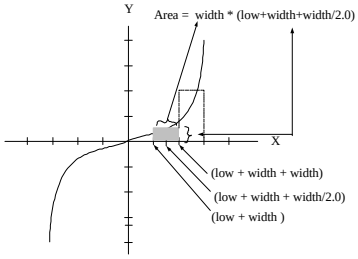
Problem-Solving Case Study: Finding the Area Under a Curve



Problem-Solving Case Study: Finding the Area Under a Curve



Problem-Solving Case Study: Finding the Area Under a Curve



Problem-Solving Case Study: Finding the Area Under a Curve

Main
- Get data
- Set width = (high – low)/divisions
- Set area = 0.0
- Set leftEdge = low
- For count going from 1 through divisions
  - Set area = area + RectArea(leftEdge, width)
  - Set leftEdge = leftEdge + width
- Print area

Problem-Solving Case Study: Finding the Area Under a Curve

RectArea (In: leftEdge, width) Out: Function value
- Return Funct(leftEdge + width/2.0) * width

Get Data (Out: low, high, divisions)
- Prompt for low and high
- Read low, high
- Prompt for divisions
- Read divisions
- Echo print input data

Funct (In: x) Out: Function value
- Return x*x*x

Problem-Solving Case Study: Finding the Area Under a Curve

```
float Funct( float );
void  GetData( float&, float&, int& );
float RectArea( float, float );
int main()
{
    float low; float high; float width; float leftEdge;
    float area; int divisions; int   count;

    GetData(low, high, divisions);
    width = (high - low) / float(divisions);
    area = 0.0; leftEdge = low;
    for (count = 1; count <= divisions; count++)
    {
        area = area + RectArea(leftEdge, width);
        leftEdge = leftEdge + width;
    }
    cout << "The result is equal to "
         << setprecision(7) << area << endl;
}
```


Problem-Solving Case Study: Finding the Area Under a Curve

```
void GetData( /* out */ float& low,          // Bottom of interval
              /* out */ float& high,         // Top of interval
              /* out */ int&   divisions )   // Division factor
// Postcondition:
//     All parameters (low, high, and divisions)
//     have been prompted for, input, and echo printed
{
    cout << "Enter low and high values of desired interval"
         << " (floating point)." << endl;
    cin >> low >> high;
    cout << "Enter the number of divisions to be used (integer)."
         << endl;
    cin >> divisions;
    cout << "The area is computed over the interval " << low
         << endl << "to " << high << "          with " << divisions
         << " subdivisions of the interval." << endl;
}
```

Problem-Solving Case Study: Finding the Area Under a Curve

```
float RectArea( /* in */ float leftEdge,    // Left edge of
                                            //   rectangle
                /* in */ float width )   // Width of rectangle
// Computes the area of a rectangle that starts at leftEdge and is
// "width" units wide. The height is given by the value
// computed by Funct at the horizontal midpoint of the rectangle
// Precondition:
//     leftEdge and width are assigned
// Postcondition:
//     Function value == area of specified rectangle
{
    return Funct(leftEdge + width / 2.0) * width;
}
```


Problem-Solving Case Study: Finding the Area Under a Curve

```
float Funct( /* in */ float x )    // Value to be cubed

// Computes x cubed.  You may replace this function with any
// single-variable function

// Precondition:
//     The absolute value of x cubed does not exceed the
//     machine's maximum float value
// Postcondition:
//     Function value == x cubed

{
    return x * x * x;
}
```