1. **Problem Statement**

Program uses a newly made Vector class with different methods to be called in main, where the class is tested out with different values being passed in.

1. **Requirements**
   1. **Assumptions**
      1. We treat numbers as integers throughout the program
      2. Call declarations in main are appropriate to the class
   2. **Specifications**
      1. Write given methods from header file
      2. Check for errors
      3. Call vector class by passing different values
2. **Decomposition Diagram**

|  |  |  |
| --- | --- | --- |
| **Main** | | |
| **Input** | **Process** | **Output** |
| Data to be passed in to the methods of Vector class | Using the called function, initialize, make pointers, or organize data | Print the data in organized manner |
| Pass in size | Assign size to the private variable |  |
|  | Checks if any error exists | Displays error message if an error exists |

1. **Test Strategy**
   1. Valid Data
   2. Invalid Data
2. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid | 1 | Enter negative number |  |  |  |  |
| Valid | 1 | Input positive number |  |  |  |  |
| Valid | 2 | Empty vector being printed out |  |  |  |  |
| Valid | 3 | Vector with all zeroes printed out |  |  |  |  |

1. **Initial Algorithm**
   1. In main function
      1. Print out the data in each vector
      2. Set each vector to a specific data, according to test cases
      3. Print data again by calling print function
      4. Make and prove a deep copy
   2. In class Vector
      1. Make a constructor
      2. Add a function that sets the private variable of s
         1. This also makes all entries 0
      3. Copy constructor
         1. Makes a deep copy
      4. Make a default destructor
      5. Add a function to print all the data in the vector in organized manner
      6. Add a function that stores a value at given position
         1. Check if position is between 0 and the size of vector.
      7. In private section, initialize the private data types for size and entries
         1. \*entries to point to array of integers with size entries.
2. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid | 1 | Enter negative number | c.set(0,-1) | Error message |  |  |
| Valid | 1 | Input positive number | c.set(1,0) | Prints c |  |  |
| Valid | 2 | Empty vector being printed out | Before adding anything, print the vector | [] |  |  |
| Valid | 3 | Vector with all zeroes printed out | Vector b(3)  b.print() | [ 0 0 0 ] |  |  |

1. **Code**

Vector.h

#include <iostream>

using namespace std;

#ifndef VECTOR\_H

#define VECTOR\_H

class Vector

{

public:

Vector();

Vector(int s);

Vector(const Vector & other);

~Vector();

void print();

void set(int val, int pos);

private:

int size;

int \*entries;

};

#endif

Vector.cpp

#include "Vector.h"

Vector::Vector()

{

}

Vector::~Vector()

{

}

Vector::Vector(int s)

{

size = s;

entries = new int[size];

for (int i = 0; i < size; i++)

{

entries[i] = 0;

}

}

Vector::Vector(const Vector & other)

{

size = other.size;

entries = new int[size];

for (int i = 0; i < size; i++)

{

entries[i] = other.entries[i];

}

}

void Vector::print()

{

cout << "[";

for (int i = 0; i < size; i++)

{

cout << entries[i] << " ";

}

cout << "]" << endl;

}

void Vector::set(int val, int pos)

{

if (pos < 0)

{

cout << "Invalid: Position is negative." << endl;

}

else if (pos > size)

{

cout << "Invalid: Position is greater than size" << endl;

}

else {

entries[pos] = val;

}

}

Main()

// CIS200\_Lab\_4\_Home.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "Vector.h"

int main()

{

Vector a, b(3), c(3);

cout << "a.print()" << endl;

a.print();

cout << endl << "b.print()" << endl;

b.print();

cout << endl << "c.set(0, -1)" << endl;

c.set(0, -1);

cout << endl << "c.set(1, 0)" << endl;

c.set(1, 0);

cout << endl << "c.set(2, 1)" << endl;

c.set(2, 1);

cout << endl << "c.set(3, 2)" << endl;

c.set(3, 2);

cout << endl << "c.set(4, 3)" << endl;

c.set(4, 3);

cout << endl << "c.print()" << endl;

c.print();

cout << endl << "Vector d(c) ---- d.print()" << endl;

Vector d(c);

d.print();

cout << endl << "d.set(0, 1)" << endl;

d.set(0, 1);

cout << endl << "d.print()" << endl;

d.print();

cout << endl << "c.print()" << endl;

c.print();

system("pause");

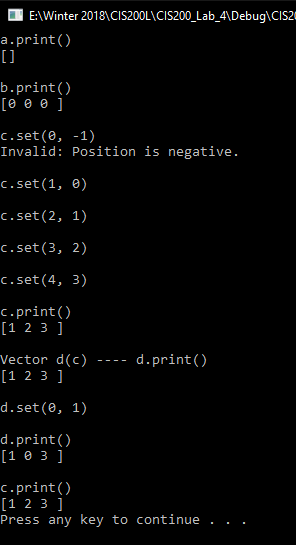
return 0;

}

1. **Updated Algorithm**
   1. In main function
      1. For each test and changes done to vectors in the main, print out the command so it is easy to read the outputs
      2. Print out the data in each vector
      3. Set each vector to a specific data, according to test cases
      4. Print data again by calling print function
      5. Make and prove a deep copy
   2. In class Vector
      1. Make a constructor
      2. Add a function that sets the private variable of s
         1. This also makes all entries 0
      3. Copy constructor
         1. Makes a deep copy
      4. Make a default destructor
      5. Add a function to print all the data in the vector in organized manner
      6. Add a function that stores a value at given position
         1. Check if position is between 0 and the size of vector.
      7. In private section, initialize the private data types for size and entries
         1. \*entries to point to array of integers with size entries.
2. **Test Plan Version 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid | 1 | Enter negative number | c.set(0,-1) | Error message | Error message | Pass |
| Valid | 1 | Input positive number | c.set(1,0) | Prints c | Prints c | Pass |
| Valid | 2 | Empty vector being printed out | Before adding anything, print the vector | [] | [] | Pass |
| Valid | 3 | Vector with all zeroes printed out | Vector b(3)  b.print() | [ 0 0 0 ] | [ 0 0 0 ] | Pass |

1. **Screenshots**



1. **Status**

Program runs good with assumptions and test cases in mind