CPP Problem Design Example

Subject: N Dim Vector

Contributor: 陳泳峰, 陳宥潤, 范茗翔

Main testing concept: Class Implementation

Basics	Functions
■ C++ BASICS	□ SEPARATE COMPILATION AND NAMESPACES
■ FLOW OF CONTROL	□ STREAMS AND FILE I/O
□ FUNCTION BASICS	□ RECURSION
■ PARAMETERS AND OVERLOADING	□ INHERITANCE
□ ARRAYS	□ POLYMORPHISM AND VIRTUAL FUNCTIONS
■ STRUCTURES AND CLASSES	□ TEMPLATES
■ CONSTRUCTORS AND OTHER TOOLS	□ LINKED DATA STRUCTURES
■ OPERATOR OVERLOADING, FRIENDS, AND REFERENCES	□ EXCEPTION HANDLING
□ STRINGS	□ STANDARD TEMPLATE LIBRARY
□ POINTERS AND DYNAMIC ARRAYS	□ PATTERNS AND UML

Description:

Define a class named VecNf as a n-dimensional float vector. Please implement the following functions and overload the following operations for VecNf.

The data inside VecNf should NOT share with others. So make sure any copy/assign operations would make a hard copy.

1. Some proper constructor
   i. Construct one-dimensional null/zero vector in default.
   ii. Using (float *S, int N) as parameter, construct using the first N elements of array S.
   iii. Any other constructor to make sure VecNf can be called by value.
   iv. Do not print anything when constructor called.
2. Assignment operator --- VecA = VecB (Assign a VecNf with VecNf), When assignment operator called, print "ASSIGNMENT!!!" in a line.
3. Subscript operator (int Index) --- Vec [Index] (Return the reference of the Index-th element of Vec. The first element is Vec[0])
4. Arithmetic operator ---
   i. VecA plus(+) VecB (vector addition)
   ii. VecA minus(-) VecB (vector subtraction)
   iii. VecA product(*) VecB (inner product)
   iv. float product(*) Vec (scale operation)
   v. Vec product(*) float (scale operation)
5. Size() --- return the dimensional of VecNf.

For any arithmetic operator, make sure the two VecNf have the same dimensional. If they are not, print "dimensions inconsistent" in a line and return 0.0f or 1-Dim Null/Zero vector.

Input:

The input is defined by the main function.
We will change the main function for testing.

Output:

The output is defined by the main function.
We will change the main function for testing.

Sample Input / Output：

Sample Input

```
#include<iostream>
#include "VecNf.h"
using namespace std;

void doNothing(VecNf tar) { return; }

int main()
{
  float a_value[] = { 3.0, 2.0 };
  float b_value[] = { 1, 2, 3 };
  float c_value[] = { 6, 5, 4 };
  VecNf A(a_value, 2);
  VecNf B(b_value, 3);
  VecNf C(c_value, 3);
  VecNf T;
  T = A; // Assignment
  for (int i = 0; i < T.Size(); i++) {
      cout << T[i] << “ ” ;
  } cout << endl;

  T = B; // Assignment
  for (int i = 0; i < T.Size(); i++) {
      cout << T[i] << “ ” ;
  } cout << endl;

  T = B + C; // Vector addition
  for (int i = 0; i < T.Size(); i++) {
      cout << T[i] << “ ” ;
  } cout << endl;

  doNothing(T); // call by value

  cout << C * B << endl; // Scale

  cout << A * C << endl; // Inconsistent

  system("pause");
  return 0;
}
```

Sample Output

```
ASSIGNMENT!!!
3 2
ASSIGNMENT!!!
1 2 3
ASSIGNMENT!!!
7 7 7
28
dimensions inconsistent
0
```

■ Eazy,Only basic programming syntax and structure are required.
□ Medium,Multiple programming grammars and structures are required.
□ Hard,Need to use multiple program structures or complex data types.

Expected solving time:

40 minutes
Other notes: