题14.7

**Array Class Template**

题目要求：

将fig,11.10~11.11的Array类改写成类模板。

解题思路：

将原来的数据成员中的int型的指针ptr改为template<typename T>中的T类型的指针，并且在成员函数和友元函数中做适当更改。

源代码：

//14.7.cpp

# include <iostream>

# include "Array.h"

using namespace std;

int main()

{

Array<int> arrayInt(10);

Array<double> arrayDouble(10);

Array<char> arrayChar(10);

for (int i = 0; i < 10; i++)

{

arrayInt[i] = i;

arrayDouble[i] = (double)i;

arrayChar[i] = (char)(i + 65);

}

cout << arrayInt;

cout << setprecision(2) << fixed << arrayDouble;

cout << arrayChar;

system("pause");

return 0;

}

//Array.h

# ifndef ARRAY\_H

# define ARRAY\_H

# include <iostream>

# include <iomanip>

# include <cstdlib>

using namespace std;

template<typename T>

class Array {

friend ostream &operator<<(ostream &output, const Array<T> &a)

{

int i;

for (i = 0; i < a.size; ++i)

{

output << setw(12) << a.ptr[i];

if ((i + 1) % 4 == 0)

output << endl;

}

if (i % 4 != 0)

output << endl;

return output;

}

friend istream &operator>>(istream &input, Array<T> &a)

{

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

input >> a.ptr[i];

return input;

}

public:

Array(int = 10);

Array(const Array &);

~Array();

int getSize() const;

const Array &operator=(const Array &);

bool operator==(const Array &) const;

bool operator!=(const Array &right) const

{

return !(\*this == right);

}

T &operator[](int);

T operator[](int) const;

private:

int size;

T \*ptr;

};

template<typename T>

Array<T>::Array(int arraySize)

{

if (arraySize > 0)

size = arraySize;

else

throw invalid\_argument("Array size must be greater than 0");

ptr = new T[size];

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

ptr[i] = 0;

}

template<typename T>

Array<T>::Array(const Array &arrayToCopy)

:size(arrayToCopy.size)

{

ptr = new T[size];

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

ptr[i] = arrayToCopy.ptr[i];

}

template<typename T>

Array<T>::~Array()

{

delete[] ptr;

}

template<typename T>

int Array<T>::getSize() const

{

return size;

}

template<typename T>

const Array<T> &Array<T>::operator=(const Array &right)

{

if (&right != this)

{

if (size != right.size)

{

delete[] ptr;

size = right.size;

ptr = new T[size];

}

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

ptr[i] = right.ptr[i];

}

return \*this;

}

template<typename T>

bool Array<T>::operator==(const Array &right) const

{

if (size != right.size)

return false;

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

if (ptr[i] != right.ptr[i])

return false;

return true;

}

template<typename T>

T &Array<T>::operator[](int subscript)

{

if (subscript < 0 || subscript >= size)

throw out\_of\_range("Subscript out of range");

return ptr[subscript];

}

template<typename T>

T Array<T>::operator[](int subscript) const

{

if (subscript < 0 || subscript >= size)

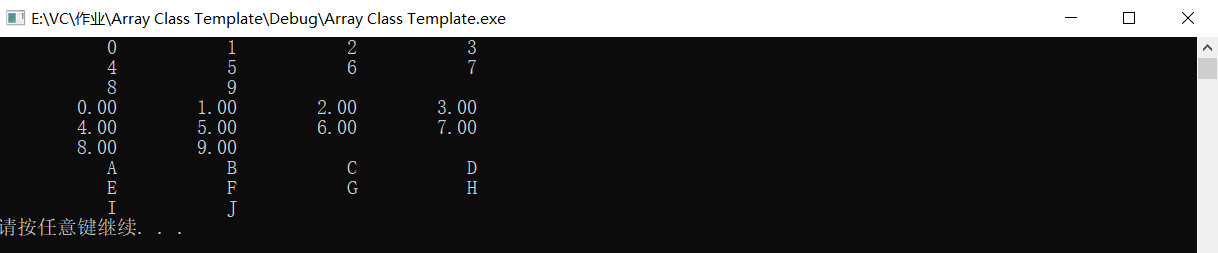
throw out\_of\_range("Subscript out of range");

return ptr[subscript];

}

#endif

运行结果：



题14.8

问题：

Distinguish between the terms “function template” and “function-template specialization.

回答：

函数模板是一个模板，不能直接使用，在使用之前必须进行特例化，特例化即对函数模板中的不确定类型的变量进行类型确定的过程。