```cpp
//This is the implementation file: pfarray.cpp.
//This is the implementation of the template class PFArray.
//The interface for the template class PFArray is in the file pfarray.h.

#include "pfarray.h"
#include <iostream>
using std::cout;

namespace PFArraySavitch {
template<class T>
PFArray<T>::PFArray() :
        capacity(50), used(0) {
    a = new T[capacity];
}

template<class T>
PFArray<T>::PFArray(int size) :
        capacity(size), used(0) {
    a = new T[capacity];
}

template<class T>
PFArray<T>::PFArray(const PFArray<T>& pfaObject) :
        capacity(pfaObject.getCapacity()), used(pfaObject.getNumberUsed()) {
    a = new T[capacity];
    for (int i = 0; i < used; i++)
        a[i] = pfaObject.a[i];
}

template<class T>
void PFArray<T>::addElement(const T& element) {
    if (used >= capacity) {
        cout << "Attempt to exceed capacity in PFArray.\n";
        exit(0);
    }
    a[used] = element;
    used++;
}

template<class T>
bool PFArray<T>::full() const {
    return (capacity == used);
}

template<class T>
int PFArray<T>::getCapacity() const {
    return capacity;
}

template<class T>
int PFArray<T>::getNumberUsed() const {
    return used;
}

template<class T>
void PFArray<T>::emptyArray() {
    used = 0;
}

template<class T>
T& PFArray<T>::operator[](int index) {
    if (index >= used) {
        cout << "Illegal index in PFArray.\n";
        exit(0);
    }

    return a[index];
}

template<class T>
PFArray<T>& PFArray<T>::operator =(const PFArray<T>& rightSide) {
    if (capacity != rightSide.capacity) {
        delete[] a;
        a = new T[rightSide.capacity];
    }

    capacity = rightSide.capacity;
    used = rightSide.used;
    for (int i = 0; i < used; i++)
        a[i] = rightSide.a[i];

    return *this;
}

template<class T>
PFArray<T>::~PFArray() {
    delete[] a;
}
} // PFArraySavitch
```