

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <cstring>
#include <sys/time.h>

#include "array.h"

template <typename T>
std::ostream& operator<<(std::ostream& s, const array_t<T>& rhs)
{
    for(int i=0;i < rhs.size(); i++)
        s << "[" << i << "] = " << rhs.arr[i] << std::endl;

    return s;
}

template <typename T>
const array_t<T>& array_t<T>::operator=(const array_t<T>& rhs)
{
    if(this != &rhs) { // standard alias test
        // copy all elements of rhs into this
        sz = rhs.sz;
        if(arr) delete arr;
        arr = new T [rhs.size()];
        for(int i=0; i<rhs.size(); i++) arr[i] = rhs.arr[i];
    }
    return *this;
}

template <typename T>
array_t<T>::array_t(int isz)
{
    // this constructor creates the array_t object given a size
    sz = isz;
    arr = new T [sz];

    memset(arr,0,sz*sizeof(T));
}

template <typename T>
array_t<T>::array_t(T *array, int isz)
{
    // this constructor creates the array_t object given a size
    // and source of data (namely, array)
    sz = isz;
    arr = new T [sz];

    for(int i=0; i < sz; ++i)
        arr[i] = array[i];
}

template <typename T>
array_t<T>::array_t(const array_t<T>& rhs)
{
    // this constructor creates the array_t object given a constant
    // reference to another array_t object (result is a copy of
    // the other object)
    sz = rhs.sz;
    arr = new T [sz];

    for(int i=0; i < sz; ++i)
```

```
    arr[i] = rhs.arr[i];
}

template <typename T>
void array_t<T>::randseed()
{
    struct timeval  tp;
    unsigned int   seed;

    // get time of day
    gettimeofday(&tp, NULL);

    // use microseconds as the seed
    seed = (unsigned int)tp.tv_usec;
    srand(seed);
    std::cerr << "seed = " << seed << std::endl;
}

template <typename T>
void array_t<T>::randfill()
{
    for(int i=0; i<sz; i++)
        arr[i] = static_cast<T>((float)rand()/(float)RAND_MAX*100.0);
}

template <typename T>
T array_t<T>::min()
{
    T    min = arr[0];

    // find the minimum value
    for(int i=0; i<sz; i++)
        if(arr[i] < min) min = arr[i];

    return min;
}

template <typename T>
T array_t<T>::max()
{
    T    max = arr[0];

    // find the maximum value
    for(int i=0; i<sz; i++)
        if(arr[i] > max) max = arr[i];

    return max;
}

template <typename T>
int array_t<T>::find(T value) const
{
    // if the value is found within the array,
    // return the index of its first occurrence
    // otherwise, return -1
    for(int i=0; i<sz; i++)
        if(arr[i] == value) return i;

    return -1;
}

////////// specializations //////////
```

```
template class array_t<int>;
template std::ostream& operator<<(std::ostream&, const array_t<int>&);

template class array_t<float>;
template std::ostream& operator<<(std::ostream&, const array_t<float>&);

template class array_t<double>;
template std::ostream& operator<<(std::ostream&, const array_t<double>&);
```

array

```
#include <iostream>

#include "array.h"

int main()
{
    int          seeknum;
    array_t<int>  iarr(10);
    array_t<float> farr(10);

    iarr.randseed();
    iarr.randfill();

    std::cout << "iarr:\n" << iarr << std::endl;

    std::cout << "min: " << iarr.min() << std::endl;
    std::cout << "max: " << iarr.max() << std::endl;
    std::cout << std::endl;

    std::cout << "Enter number to find: ";

    std::cin >> seeknum;
    std::cout << seeknum << " is at position " << iarr.find(seeknum) << std::endl;
}
```