ARRAY 2

Flexible Arrays

Delroy A. Brinkerhoff

THE Array CLASS MEMBERS

```
private:
    int  lower;
    int  upper;
    T*   array;
```

```
public:
    Array(int s, int e);
    ~Array() { delete[] array; }
    T& operator[](int index);
    T& at(int index);
```

THE Array CONSTRUCTOR

```
#include <stdexcept>
using namespace std;

template <class T>
Array<T>::Array(int l, int u) : lower(l), upper(u)
{
    if (upper < lower)
        throw invalid_argument("Upper must be >= lower");

    array = new T[upper - lower + 1]{};
}
```

Array INDEXING FUNCTIONS

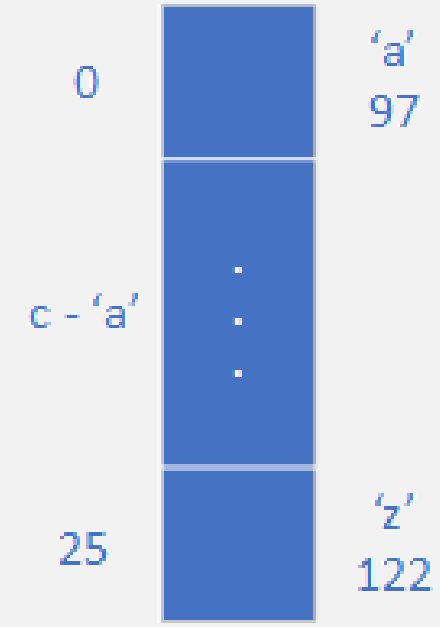
```
template <class T>
T& Array<T>::operator[](int index)
{
    return array[index - lower];
}

template <class T>
T& Array<T>::at(int index)
{
    if (index < lower || index > upper)
        throw out_of_range("Index out of bounds");

    return array[index - lower];
}
```

Array AND ANAGRAM

- Form an anagram by rearranging the letters of one phrase to form a second
- Ignore spaces, punctuation, letter case
- The two phrases have the same number of a's, b's, c's, etc.
- An anagram checker counts the occurrence of each letter



EXCEPTION HANDLING

```
try
{
    . . .
}
catch (invalid_argument ia)
{
    cerr << ia.what() << endl;
}
catch (out_of_range oor)
{
    cerr << oor.what() << endl;
}
```

THE TEST PHRASES

```
const char* p1 = "To be or not to be: that is the question, whether "
    "it's nobler in the mind to suffer the slings and arrows of "
    "outrageous fortune.";

const char* p2 = "In one of the Bard's best-thought-of tragedies, "
    "our insistent hero, Hamlet, queries on two fronts about how "
    "life turns rotten.";
```

COUNTING THE LETTER OCCURRENCES

```
Array<int> a1('a', 'z');
Array<int> a2('a', 'z');

for (size_t i = 0; i < strlen(p1); i++)
    if (isalpha(p1[i]))
        a1[tolower(p1[i])]++;

for (size_t i = 0; i < strlen(p2); i++)
    if (isalpha(p2[i]))
        a2[tolower(p2[i])]++;
```

COUNTING THE LETTER OCCURRENCES

```
Array<int> a1('a', 'z');
Array<int> a2('a', 'z');

for (size_t i = 0; i < strlen(p1); i++)
    if (isalpha(p1[i]))
        a1[tolower(p1[i])]++;

for (size_t i = 0; i < strlen(p2); i++)
    if (isalpha(p2[i]))
        a2[tolower(p2[i])]++;
```

COUNTING THE LETTER OCCURRENCES

```
Array<int> a1('a', 'z');
Array<int> a2('a', 'z');

for (size_t i = 0; i < strlen(p1); i++)
    if (isalpha(p1[i]))
        a1[tolower(p1[i])]++;

for (size_t i = 0; i < strlen(p2); i++)
    if (isalpha(p2[i]))
        a2[tolower(p2[i])]++;
```

COUNTING THE LETTER OCCURRENCES

```
Array<int> a1('a', 'z');
Array<int> a2('a', 'z');

for (size_t i = 0; i < strlen(p1); i++)
    if (isalpha(p1[i]))
        a1[tolower(p1[i])]++;

for (size_t i = 0; i < strlen(p2); i++)
    if (isalpha(p2[i]))
        a2[tolower(p2[i])]++;
```

COUNTING THE LETTER OCCURRENCES

```
Array<int> a1('a', 'z');
Array<int> a2('a', 'z');

for (size_t i = 0; i < strlen(p1); i++)
    if (isalpha(p1[i]))
        a1[tolower(p1[i])]++;

for (size_t i = 0; i < strlen(p2); i++)
    if (isalpha(p2[i]))
        a2[tolower(p2[i])]++;
```

VERIFYING OR REJECTING AN ANAGRAM

```
for (int i = 'a'; i <= 'z'; i++)
    if (a1[i] != a2[i])
    {
        cout << "NOT an anagram" << endl;
        exit(0);
    }

cout << "Valid anagram" << endl;
```