```
#include <iostream>
#include <iomanip> // Untuk setw
#include <stdexcept> // Untuk out_of_range
using namespace std;

// Mendefinisikan ukuran maksimum array dengan #define
#define MAX_SIZE 5

// Template class untuk array statis
template <class T>
class Array1Dstatis {
public:
    Array1Dstatis();

    T& operator[](int index);
    const T& operator[](int index) const;

    // Friend function untuk operator overloading
    friend ostream& operator<<(ostream& os, const Array1Dstatis<T>& arr) {
        os << "Array Statis: ";
        for (int i = 0; i < MAX_SIZE; ++i) {
            os << arr.data[i] << " ";
        }
        os << endl;
        return os;
    }

    friend istream& operator>>(istream& is, Array1Dstatis<T>& arr) {
        cout << "Masukkan " << MAX_SIZE << " nilai untuk array statis: " << endl;
        for (int i = 0; i < MAX_SIZE; ++i) {
```

```
            cout << "Nilai elemen [" << i << "]: ";
            is >> arr.data[i];
        }
        return is;
    }

    static constexpr int Size() { return MAX_SIZE; }

private:
    T data[MAX_SIZE];
};

template <class T>
Array1Dstatis<T>::Array1Dstatis() {
    for (int i = 0; i < MAX_SIZE; ++i) {
        data[i] = T();
    }
}

template <class T>
T& Array1Dstatis<T>::operator[](int index) {
    if (index >= 0 && index < MAX_SIZE) {
        return data[index];
    } else {
        throw out_of_range("Index out of range");
    }
}

template <class T>
const T& Array1Dstatis<T>::operator[](int index) const {
```

```
    if (index >= 0 && index < MAX_SIZE) {
        return data[index];
    } else {
        throw out_of_range("Index out of range");
    }
}

int main() {
    // Membuat instance dari Array1Dstatis dengan ukuran MAX_SIZE
    Array1Dstatis<int> myArray;

    // Mengisi array dengan nilai
    cin >> myArray;

    // Menampilkan nilai array
    cout << "Array Statis: " << endl;
    cout << myArray;

    return 0;
}
```