課題12

Q01: FloatArray クラスの設計（1 次元配列クラス）

（プログラム名：**ex12_1.cpp**）

C++ を用いて、float 型の 1 次元配列を表すクラス `FloatArray` を定義し、以下のメンバや操作を実装しなさい。イメージとしては、簡易版の「NumPy の 1 次元配列」のようなクラスとする。

【要件説明：クラス **FloatArray**】

- プライベートデータメンバ
  - `int size;  // 配列の要素数`
  - `float* data;  // 要素を保持する動的配列;`

- コンストラクタ・デストラクタ
  - `FloatArray(int n);`
    - コンストラクタ：要素数 n の配列を動的に確保し、すべての要素を **0.0f** で初期化する。
  - `FloatArray(const FloatArray& x);`
    - コピーコンストラクタ：x と同じサイズの動的配列を新しく確保し、全ての要素をコピーして初期化する。
  - `~FloatArray();`
    - 動的に確保した配列を `delete[]` で解放する。

- メンバ関数・演算子のオーバーロード
  - `int length() const;`
    - 配列の要素数 size を返す。
  - `float& operator[](int index);`
    - インデックス index 番目の要素への参照を返す。
  - `FloatArray& operator=(const FloatArray& x);`
    - 代入演算子 =：自分自身と x が同じサイズ・同じ内容になるように要素をコピーする。
    - すでに動的配列を確保している場合は、必要に応じて古い配列を解放してから新しい配列を確保し、要素をコピーすること。
    - 自己代入（`this == &x`）の場合にも正しく動作するように実装すること。
  - `FloatArray operator+(float value) const;`
    - 加算（配列 + 数）：各要素に value を加えた新しい FloatArray を返す。
  - `FloatArray operator+(const FloatArray& x) const;`
    - 配列同士の加算（要素ごとの和）：自分自身と x の要素ごとの和を表す新しい FloatArray を返す。
    - 基本的には、this->size と x.size が同じであると仮定してよい。

- `friend std::ostream& operator<<(std::ostream& os, const FloatArray& a);`
  - 挿入演算子 << のオーバーロード:`[a0, a1, a2, ..., a(n-1)]`のような形式で出力ストリーム os に書き出す。

【プログラム】

今回の課題では、コードを複数のファイルに分割する必要はありません。
下に示したサンプルコードを参考にして、同じファイル内に必要な処理を追記し、プログラムを完成させてください。

```C/C++
#include <iostream>

class FloatArray
{
private:
    int size;    // 配列の要素数
    float *data; // 要素を保持する動的配列

public:
    // ↓ ここを補完

    // コンストラクタ：要素数 n の配列を確保し、0.0f で初期化
    FloatArray(int n)
        : size(n), data(nullptr)
    {
        // ...
    }

    // コピーコンストラクタ
    FloatArray(const FloatArray& x)
       : size(x.size), data(nullptr)
    {
        // ...
    }

    // デストラクタ：確保した動的配列を解放
    ~FloatArray()
    {
        // ...
    }

    // 配列の要素数を返す
    int length() const
    {
        // ...
```

```
    }

    // 添字演算子
    float &operator[](int index)
    {
        // ...
    }

    // 代入演算子 =
    FloatArray &operator=(const FloatArray &x)
    {
        // ...
    }

    // 配列 + スカラ（各要素に value を加えた新しい配列を返す）
    FloatArray operator+(float value) const
    {
        // ...
    }

    // 配列 + 配列（要素ごとの和）
    FloatArray operator+(const FloatArray &x) const
    {
        // ...
    }

    // 挿入演算子 << のオーバーロード
    friend std::ostream &operator<<(std::ostream &os, const FloatArray &a)
    {
        // ...
    }
};

// 動作確認用 main、この部分は変更しないでください
using namespace std;
int main()
{
    // 要素数 5 の配列を作成
    FloatArray a(5);
    for (int i = 0; i < a.length(); ++i)
        a[i] = i * 0.3f;
    cout << "a = " << a << endl;

    FloatArray b, c;

    // 配列 + スカラのテスト
    b = a + 2.0f; // 各要素に 2.0 を加える
    cout << "b = " << b << endl;
```

```
    // 配列 + 配列のテスト
    c = a + b;
    cout << "c = " << c << endl;
}
```

【実行例】

```
a = [0, 0.3, 0.6, 0.9, 1.2]
b = [2, 2.3, 2.6, 2.9, 3.2]
c = [2, 2.6, 3.2, 3.8, 4.4]
```