Императивная парадигма программирования

Файл main.cpp

```
#include <iostream>
int main()
{
    using namespace std;
    double a(1.2), b(2.3);
    cout << a + b << endl;    // 3.5
    cout << 1 + 2 << endl;    // 3
    return 0;
}
```

Файл main.cpp

```
#include <iostream>
double add(double a, double b)
{
    return a + b;
}
int main()
{
    using namespace std;
    double a(1.2), b(2.3);
    cout << add(a, b) << endl;             // 3.5
    cout << add(1, 2) << endl;             // 3
    cout << fixed << add(1, 2) << endl;    // 3.000000
    return 0;
}
```

Файл main.cpp

```
#include <iostream>
double add(double a, double b)
{
    return a + b;
}
long int add(long int a, long int b)
{
    return a + b;
}
int main()
{
    using namespace std;
    double a(1.2), b(2.3);
    long int c(1), d(2);
    cout << add(a, b) << endl;    // 3.5
    cout << add(c, d) << endl;    // 3
    return 0;
}
```

Модульная парадигма программирования

Файл operations.h

```
double add(double a, double b)
{
    return a + b;
}
```

Файл main.cpp

```
#include <iostream>
#include ”operations.h”
int main()
{
    using namespace std;
    double a(1.2), b(2.3);
    cout << add(a, b) << endl;    // 3.5
    cout << add(1, 2) << endl;    // 3
    return 0;
}
```

Файл operations.h

```
namespace DoubleOperations
{
    double add(double a, double b)
    {
        return a + b;
    }
}
namespace IntegerOperations
{
    long int add(long int a, long int b)
    {
        return a + b;
    }
}
```

Файл main.cpp

```
#include <iostream>
#include ”operations.h”
int main()
{
    using namespace std;
    cout << IntegerOperations::add(1, 2) << endl;      // 3
    cout << DoubleOperations::add(1.2, 2.3) << endl;   // 3.5
    return 0;
}
```

Файл main.cpp

```
#include <iostream>
using namespace std;
class Double
{
    double number;
    public:
    Double(double d = 0.0) : number(d) {}
    Double add(Double, Double);
    friend ostream& operator<<(ostream&, Double);
};
Double Double::add(Double a, Double b)
{
    Double temporary;
    temporary.number = a.number + b.number;
    return temporary;
}
ostream& operator<<(ostream& stream, Double d)
{
    return stream << d.number;
}
int main()
{
    Double a(1.2), b(2.3), c;
    cout << a << endl;                // 1.2
    cout << b << endl;                // 2.3
    cout << c << endl;                // 0
    cout << a.add(a, b) << endl;      // 3.5
    cout << b.add(a, b) << endl;      // 3.5
    cout << c.add(a, b) << endl;      // 3.5
    return 0;
}
```

Файл main.cpp

```
#include <iostream>
using namespace std;
class Double
{
    double number;
    public:
    Double(double d = 0.0) : number(d) {}
    friend Double operator+(Double, Double);
    friend ostream& operator<<(ostream&, Double);
};
Double operator+(Double a, Double b)
{
```

```
    Double temporary;
    temporary.number = a.number + b.number;
    return temporary;
}
ostream& operator<<(ostream& stream, Double d)
{
    return stream << d.number;
}
int main()
{
    Double a(1.2), b(2.3), c(a + b);
    cout << a << endl;                    // 1.2
    cout << b << endl;                    // 2.3
    cout << c << endl;                    // 3.5
    cout << a + b << endl;                // 3.5
    return 0;
}
```

Файл main.cpp

```
#include <iostream>
using namespace std;
class Double
{
    double number;
    public:
    Double(double d = 0.0) : number(d) {}
    Double operator+(Double);
    friend ostream& operator<<(ostream&, Double);
};
Double Double::operator+(Double a)
{
    Double temporary(*this);
    temporary.number += a.number;
    return temporary;
}
ostream& operator<<(ostream& stream, Double d)
{
    return stream << d.number;
}
int main()
{
    Double a(1.2), b(2.3), c(a + b);
    cout << a << endl;                    // 1.2
    cout << b << endl;                    // 2.3
    cout << c << endl;                    // 3.5
    cout << a + b << endl;                // 3.5
    return 0;
}
```

Обобщенная парадигма программирования

Файл main.cpp

```
#include <iostream>
template<class T>
T add(T a, T b)
{
    return a + b;
}
int main()
{
    using namespace std;
    cout << add(1.2, 2.3) << endl;             // 3.5
    cout << add<double>(1.2, 2.3) << endl;     // 3.5
    cout << add(1, 2) << endl;                 // 3
    cout << add<int>(1, 2) << endl;            // 3
    return 0;
}
```

Файл main.cpp

```
#include <iostream>
using namespace std;
template<class T> class Number
{
    T number;
    public:
    Number(T d = 0) : number(d) {}
    friend Number<T> operator+(Number<T>, Number<T>);
    friend ostream& operator<<(ostream&, Number<T>);
};
template<class T>
Number<T> operator+(Number<T> a, Number<T> b)
{
    Number<T> temporary;
    temporary.number = a.number + b.number;
    return temporary;
}
template<class T>
ostream& operator<<(ostream& stream, Number<T> d)
{
    return stream << d.number;
}
int main()
{
    Number<double> a(1.2), b(2.3), c(a + b);
    cout << a << endl;                              // 1.2
    cout << b << endl;                              // 2.3
    cout << c << endl;                              // 3.5
    cout << a + b << endl;                          // 3.5
```

```
    Number<int> d(1), e(2), f(d + e);
    cout << d << endl;                              // 1
    cout << e << endl;                              // 2
    cout << f << endl;                              // 3
    cout << d + e << endl;                          // 3
    return 0;
}
```

Файл main.cpp

```
#include <iostream>
using namespace std;
template<class T> class Number
{
    T number;
    public:
    Number(T d = 0) : number(d) {}
    Number<T> operator+(Number<T>);
    friend ostream& operator<<(ostream&, Number<T>);
};
template<class T>
Number<T> Number<T>::operator+(Number<T> a)
{
    Number<T> temporary(*this);
    temporary.number += a.number;
    return temporary;
}
template<class T>
ostream& operator<<(ostream& stream, Number<T> d)
{
    return stream << d.number;
}
int main()
{
    Number<double> a(1.2), b(2.3), c(a + b);
    cout << a << endl;                              // 1.2
    cout << b << endl;                              // 2.3
    cout << c << endl;                              // 3.5
    cout << a + b << endl;                          // 3.5
    Number<int> d(1), e(2), f(d + e);
    cout << d << endl;                              // 1
    cout << e << endl;                              // 2
    cout << f << endl;                              // 3
    cout << d + e << endl;                          // 3
    return 0;
}
```