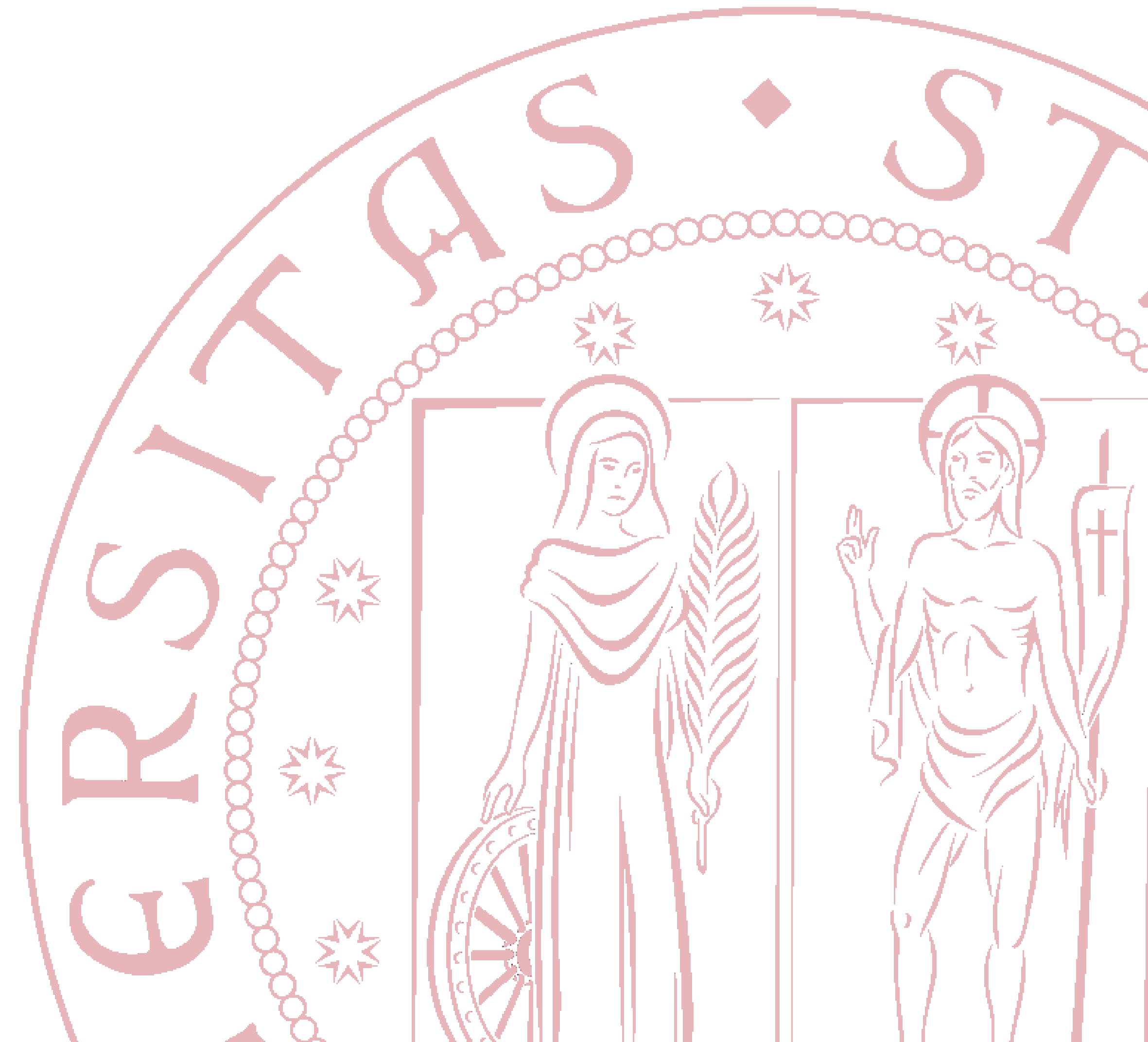


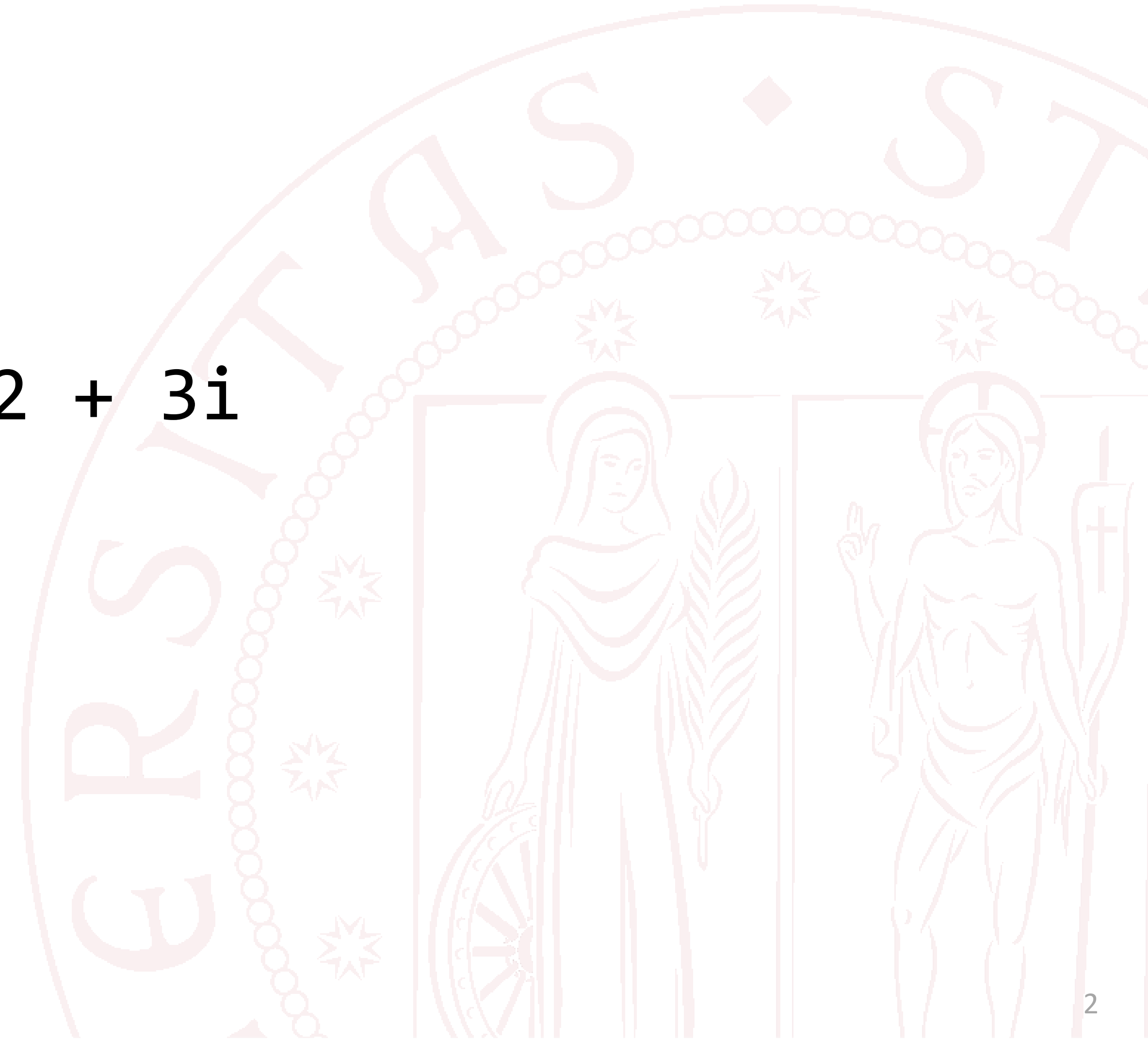
4.4 – Esempi di overloading



La classe Complex

- Vogliamo disegnare una classe Complex per la rappresentazione dei numeri complessi
- **Come la disegniamo?**

Numero complesso: $2 + 3i$



La classe Complex | Interfaccia

- L'interfaccia della classe (Complex.h):

```
class Complex {  
  
    public:  
        Complex(int real, int imag);  
  
        int real(void) const;  
        int imag(void) const;  
  
    private:  
        int r{0};  
        int i{0};  
  
};
```

La classe Complex | Implementazione

Complex.h

```
class Complex {  
  
    public:  
        Complex(int real, int imag);  
        Complex(int real);  
        Complex(void);  
  
        int real(void) const;  
        int imag(void) const;  
  
    private:  
        int r{0};  
        int i{0};  
  
};
```

Complex.cpp

```
Complex::Complex(int real, int imag) :  
    r{real}, i{imag} {}  
  
Complex::Complex(int real) : r{real} {}  
Complex::Complex(void) {}  
  
int Complex::real(void) const {  
    return r;  
}  
  
int Complex::imag(void) const {  
    return i;  
}
```

- Adesso vogliamo fornire l'overload degli operatori: <<, +, =

La classe Complex | Overload <<

- É una funzione membro o un helper function?

Complex.h

```
#include <iostream>
```

```
class Complex {
```

```
    public:
```

```
        Complex(int real, int imag);
```

```
        Complex(int real);
```

```
        Complex(void);
```

```
        int real(void) const;
```

```
        int imag(void) const;
```

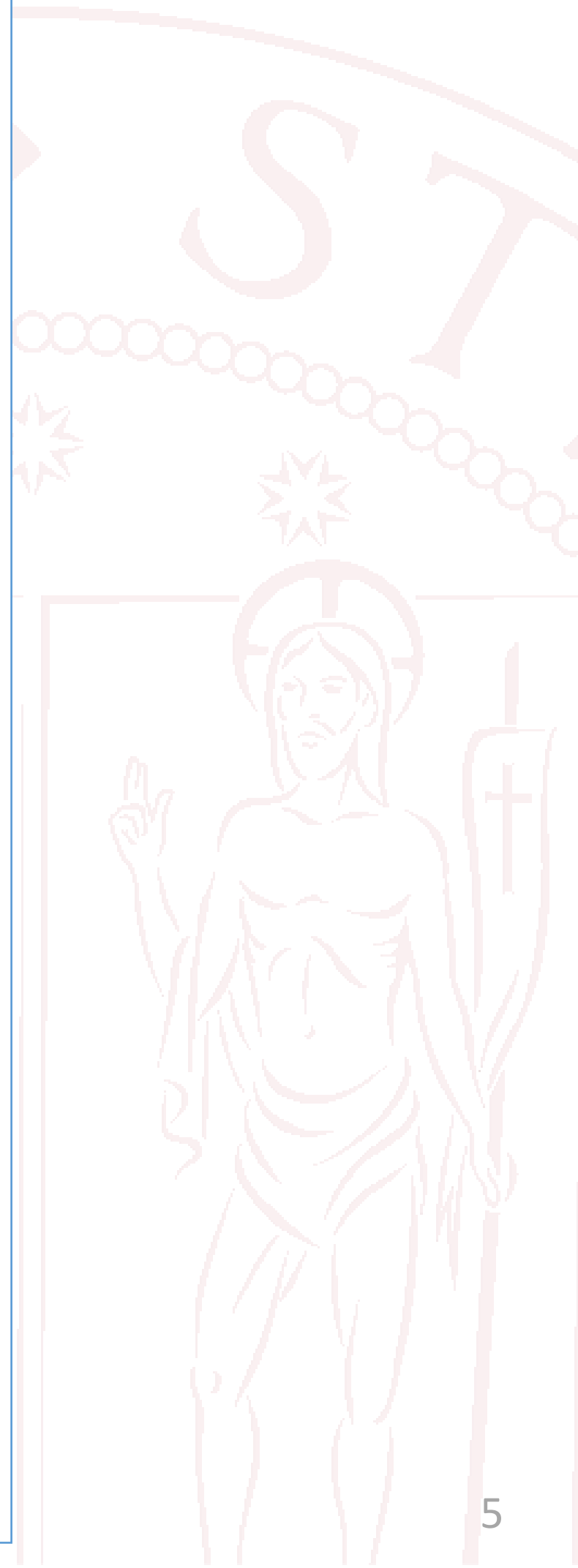
```
    private:
```

```
        int r{0};
```

```
        int i{0};
```

```
};
```

```
std::ostream& operator<<(std::ostream& out, const Complex& a);
```



La classe Complex | Overload <<

Prendo la reference a uno stream in ingresso e ritorno uno stream in uscita

Complex.cpp

```
//...  
std::ostream& operator<<(std::ostream& out, const Complex& a) {  
    return out << a.real() << "+" <<a.imag() <<"i";  
}  
//...
```

Uso le funzioni membro pubbliche per accedere alla rappresentazione

É costante → non viene modificato all'interno della funzione

La classe Complex | Overload +

- É una funzione membro o un helper function?

Complex.h

```
#include <iostream>
```

```
class Complex {
```

```
    public:
```

```
        Complex(int real, int imag);
```

```
        Complex(int real);
```

```
        Complex(void);
```

```
        int real(void) const;
```

```
        int imag(void) const;
```

```
    private:
```

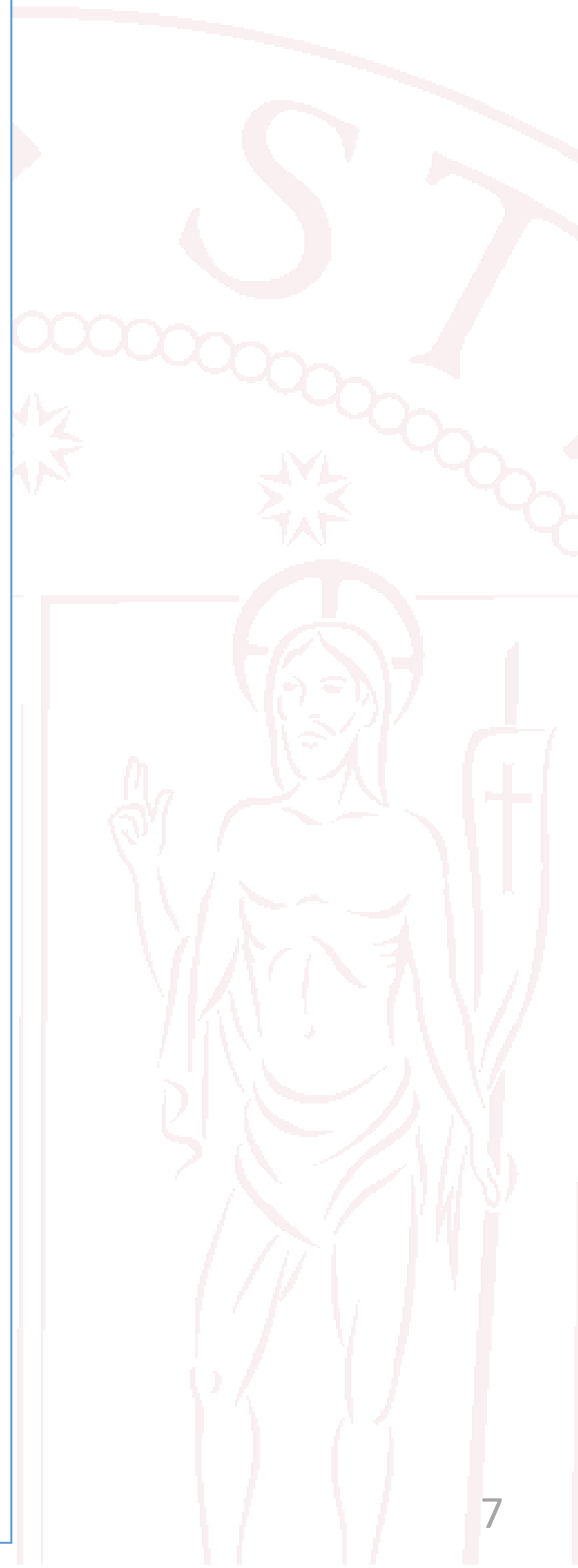
```
        int r{0};
```

```
        int i{0};
```

```
};
```

```
// ...
```

```
Complex operator+(const Complex& a, const Complex& b);
```



La classe Complex | Overload +

Complex.cpp

//...

```
Complex operator+(const Complex& a, const Complex& b) {
```

```
    int realp, imagp;
```

```
    realp = a.real() + b.real();
```

```
    imagp = a.imag() + b.imag ();
```

```
    return Complex(realp, imagp);
```

```
}
```

//...

Prendo la reference a due Complex in ingresso e ritorno un Complex

Sono costanti → non vengono modificati all'interno della funzione

Ritorno un nuovo oggetto Complex e sfrutto il costruttore

Cosa succede se sommo un Complex e un intero?

La classe Complex | Overload + [Complex e int]

Complex.h

```
#include <iostream>

class Complex {

    public:
        Complex(int real, int imag);
        Complex(int real);
        Complex(void);

        int real(void) const;
        int imag(void) const;

    private:
        int r{0};
        int i{0};

};
// ...
Complex operator+(const Complex& a, const Complex& b);
Complex operator+(const Complex& a, int b);
```

La classe Complex | Overload + [Complex e int]

Prendo la reference a un Complex e un int in ingresso e ritorno un Complex

Complex.cpp

```
//...  
Complex operator+(const Complex& a, int b) {  
    return a + Complex(b);  
}  
//...
```

Sfrutto il costruttore Complex(int real)

La classe Complex | Overload =

- É una funzione membro o un helper function?

Complex.h

```
#include <iostream>

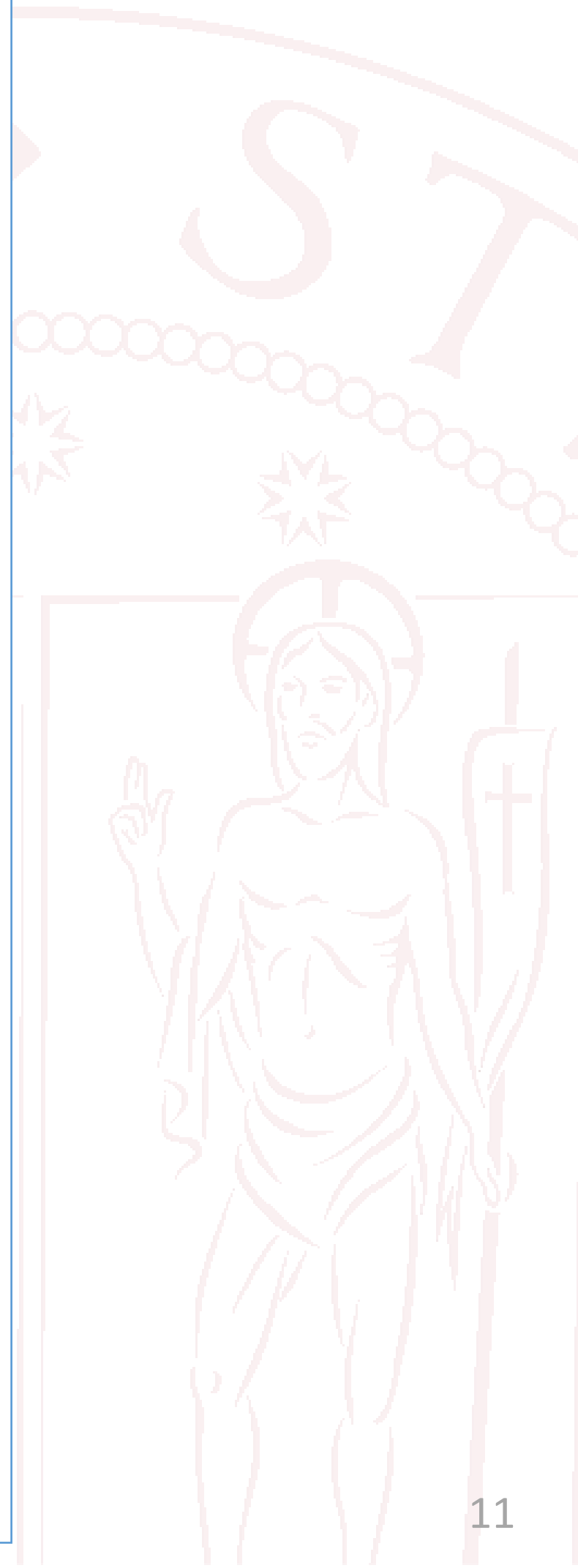
class Complex {

    public:
        Complex(int real, int imag);
        Complex(int real);
        Complex(void);

        int real(void) const;
        int imag(void) const;
        Complex& operator=(const Complex& a);

    private:
        int r{0};
        int i{0};

};
// ...
```



La classe Complex | Overload =

Complex.cpp

```
//...
```

```
Complex& Complex::operator=(const Complex& a) {
```

```
    r = a.real();
```

```
    i = a.imagine();
```

```
    return *this;
```

```
}
```

```
//...
```

Prendo la reference a un Complex in ingresso e ritorno la reference a un Complex

É funzione membro, può accedere ai membri privati r e i

Cosa devo ritornare?

L'oggetto stesso! Uso il puntatore this dereferenziato (con preposto *)

La classe Complex | Test

main.cpp

```
#include <iostream>
#include "Complex.h"

int main(void) {

    Complex c1(2, 1);           // Costruttore
    Complex c2;                 // Costruttore di default
    Complex c3;                 // Costruttore di default
    Complex c4(1, 1);           // Costruttore
    Complex c5, c6;             // Costruttore di default

    c3 = c1;                    // assegnamento
    c4 = c1 + c4;                // somma due Complex
    c5 = c1 + 3;                 // somma Complex e intero
    c6 = 3 + c1;                 // somma intero e Complex

    std::cout<<"Complex number c1: "<<c1<<std::endl;
    std::cout<<"Complex number c2: "<<c2<<std::endl;
    std::cout<<"Complex number c3: "<<c3<<std::endl;
    std::cout<<"Complex number c4: "<<c4<<std::endl;
    std::cout<<"Complex number c5: "<<c5<<std::endl;
    std::cout<<"Complex number c6: "<<c6<<std::endl;

    return 0;
}
```

