## 1. Декларация на клас 2. Обекти 3. Указатели към обекти 4. Функции-членове (методи) на клас

```
1. Декларация на клас

class Person
{
    public:
        void setData( void );
        void display( void );
    private:
        char*name;
        int age;
    };
```

```
1. Декларация на клас

class ComplexNum

(
public:
    void inputComplax( void );
    ComplexNum addComplex( ComplexNum, ComplexNum );
    ComplexNum multiComplex( ComplexNum, ComplexNum);
    double modulComplex( ComplexNum, ComplexNum);
    double modulComplex( ComplexNum);
    void displayComplex( void );
    double getReal( void ) // извличане на реализет част
    ( return real;)
    double getImag( void ) // извличане на знактинерната част
    ( return imag;)
    private;
    double real;
    double imag;
    );

**EXACCMM HOMETH
```

```
Provid ComplexNum::inputComplex( void )

{

cout << "Vavedete Real: ";

cin >> real;

cout << "Vavedete Imag: ";

cin >> imag;
}
```

```
2. Обекти

Създаването на обекти от даден клас е аналогично на дефинирането на променливи от даден тип.

<име_клас> <име_променлива>;

Обръщението към член-функция (метод) на даден обект има следния синтаксис:

<име_обект>.<име_функция>( <фактич. парам.>);
```

```
2. Обекти

Съществуват три начина за въвеждане на обекти от един и същ клас в тялото на член-функция:

чрез текущия обект, чрез който се извиква методът;

чрез предаване на обект като параметър на метода;

чрез дефиниране на локален обект в тялото на метода.
```

```
class Access
{
    public:
        void method( Access, int);
    private:
        int i,j;
};
```

```
3. Указатели към обекти

Person *ptr;
ComplexNum numObj;
ComplexNum *numPtr = &numObj;

Person george;
ptr = &george;
```

```
3. Указатели към обекти

// Деклариране на указател към клас ComplexNum

ComplexNum *ptrNum;

// Създаване на обект от клас ComplexNum

ptrNum = new ComplexNum;

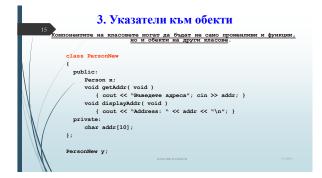
if ( !(ptrNum = new ComplexNum) )

cout << "No memory";
```

```
3. Указатели към обекти

void Person::setData(void)

{
    char buf[20];
    cout << endl << "Name= ";
    cin >> buf;
    name = new char[strlen(buf)+1];
    strcpy(name,buf);
    cout << "Age = ";
    cin >> age;
}
```





```
4. Функции-членове (методи) на клас

Методите условно се делят на:

Управляващи;

Инструментални;

Помощни;

методи за достъп.
```



## 4.2. Инструментални методи Те реализират потенциалните възможности на класа.

```
4.2. Инструментални методи

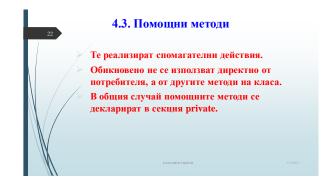
// умножение на две комплексни числа

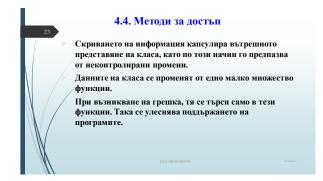
ComplexNum ComplexNum::multiComplex(ComplexNum x, ComplexNum y)

(
ComplexNum z; // работен oбект
z.real = x.real * y.real - x.imag*y.imag;
z.imag = x.real * y.real - x.imag * y.real;
return z;

}
```

## 4.2. Инструментални методи // модул на комплексно число double ComplexNum::modulComplex(ComplexNum x) { return sqrt( x.real\*x.real + x.imag\*x.imag ); }





```
void Person::setData(char *buf)

{

name = new char[strlen(buf)+1];

strcpy(name,buf);

cout << "Age = ";

cin >> age;
}
```

```
4.5. Константни методи

При дефиниране на класа трябва явно да се укаже кон функции са безопасии.

3а целта се използва ключовата дума const.

class Screen
{
    public:
        char get( void ) const
        { return *cursor; }
};

Константните обекти могат да се обръщат само към константните методи на класа.

Метод, който променя данните на класа не може да се дефинира като константен метод.
```