1

КЛАСОВЕ И ОБЕКТИ

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

2

1. Декларация на клас

```
class <име_клас>
{
  public:    // секция на глобалните данни
     // декларации на данни и функции

  protected:   // секция на данните с ограничен достъп
     // декларации на данни и функции

  private:     // секция на локалните данни
     // декларации на данни и функции
};
```


3

1. Декларация на клас

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


4

1. Декларация на клас

```
class ComplexNum
{
   public:
     ..............
   private:
       double real;
       double imag;
};
```


5

1. Декларация на клас

```
class ComplexNum
{
public:
     void inputComplex( void );
     ComplexNum addComplex( ComplexNum, ComplexNum );
     ComplexNum multiComplex( ComplexNum, ComplexNum );
     double modulComplex( ComplexNum );
     void displayComplex( void );
     double getReal( void )       // извличане на реалната част
           { return real; }
     double getImag( void )       // извличане на имагинерната част
           { return imag; }
private:
     double real;
     double imag;
     };
```


6

1. Декларация на клас

```
void ComplexNum::inputComplex( void )
{
   cout << "Vavedete Real: ";
   cin >> real;

   cout << "Vavedete Imag: ";
   cin >> imag;
}
```

7

1. Декларация на клас

```
// събиране на две комплексни числа

ComplexNum ComplexNum::addComplex(ComplexNum x, ComplexNum y)
{
   ComplexNum z;    // работен обект
   z.real = x.real + y.real;
   z.imag = x.imag + y.imag;
   return z;
}
```


8

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.>);

9

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.**

10

2. Обекти

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


11

3. Указатели към обекти

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

Person george;
ptr = &george;
```


12

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";
```

13

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;
}
```


14

3. Указатели към обекти

Масивът е подреден набор от еднотипни елементи.

Масивите от обекти (масиви, чиито елементи са обекти) се дефинират по същия начин, както и масиви от основни типове.

<тип_на_класа> <име_на_масива>[<брой_елементи>];

15

3. Указатели към обекти

Компонентите на класовете могат да бъдат не само променливи и функции, но и обекти на други класове.

```
class PersonNew
{
  public:
     Person x;
     void getAddr( void )
        { cout << "Въведете адреса"; cin >> addr; }
     void displayAddr( void )
        { cout << "Address: " << addr << "\n"; }
  private:
     char addr[10];
};

PersonNew y;
```


16

3. Указатели към обекти

Езикът C++ не допуска рекурсивни декларации,
т.е. в един клас не могат да бъдат декларирани компоненти, които са обекти на същия клас.

По отношение на указателите обаче, такова ограничение няма.

Пример:

```
class IntItem
{
  public:
     ..............
     IntItem item;   // Грешка!! Рекурсивна декларация
     ..............
     IntItem *next;  // Допустимо - next е указател
};
```


17

4. Функции-членове (методи) на клас

Методите условно се делят на:

- ***Управляващи;***
- ***Инструментални;***
- ***Помощни;***
- ***методи за достъп.***

18

4.1. Управляващи методи

Управляващи са методите на класа, които инициализират и присвояват обекти, преобразуват типове и управляват паметта.

Обикновено те се извикват автоматично от компилатора.

19

4.2. Инструментални методи

Те реализират потенциалните възможности на класа.

20

4.2. Инструментални методи

```
// умножение на две комплексни числа

ComplexNum ComplexNum::multiComplex(ComplexNum x, ComplexNum y)
{
    ComplexNum z;     // работен обект
    z.real = x.real * y.real – x.imag*y.imag;
    z.imag = x.real * y.imag + x.imag * y.real;
    return z;
}
```


21

4.2. Инструментални методи

```
// модул на комплексно число

double ComplexNum::modulComplex(ComplexNum x)
{
   return sqrt( x.real*x.real + x.imag*x.imag );
}
```


22

4.3. Помощни методи

- **Те реализират спомагателни действия.**
- **Обикновено не се използват директно от потребителя, а от другите методи на класа.**
- **В общия случай помощните методи се декларират в секция private.**

23

4.4. Методи за достъп

- **Скриването на информация капсулира вътрешното представяне на класа, като по този начин го предпазва от неконтролирани промени.**
- **Данните на класа се променят от едно малко множество функции.**
- **При възникване на грешка, тя се търси само в тези функции. Така се улеснява поддържането на програмите.**

24

4.4. Методи за достъп

```
void Person::setData(char *buf)
{
    name = new char[strlen(buf)+1];
    strcpy(name,buf);
    cout << "Age = ";
    cin >> age;
}
```

4.5. Константни методи

25

- При дефиниране на класа трябва явно да се укаже кои функции са безопасни.
- За 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.

4.5. Константни методи

26

```
class Screen
{
   public:
     char get ( int x, int y );
     char get ( int x, int y ) const;
   // ..................
};
```

Кой от двата метода ще бъде избран зависи от самия обект:

```
   const Screen cs;
   Screen s;
```