Abstrakcija un iekapsulēšana

Klase

```
class CoordPoint {
   private:
      int X;
      int Y;
   public:
      CoordPoint();                  // konstruktors
      CoordPoint(int, int);          // konstruktors
       ...
};
```

Objekti

```
CoordPoint CP1, CP2(1,2), CP3 = CoordPoint(3, 4),
   *CP4 = new CoordPoint(5, 6), *CP5;
CP5 = new CoordPoint(7, 8);
```

CP1, CP2, CP3 – *statiskie* objekti. CP4, CP5 – *dinamiskie* objekti

Dinamisku objektu iznīcināšana

```
delete CP4; //destruktora izsaukums
delete CP5; //destruktora izsaukums
```

Destruktors. Objektorientētā izvade

```
~CoordPoint() {
   cout << "Message from the \"CoordPoint\" - destroyed!" <<
      endl;   // endl ir '\n'
}
```

Metožu izsaukumi statiskos un dinamiskos objektos

```
CP1.Print();        //izsaukums no statiska objekta
(*CP4).Print();     //izsaukums no dinamiska objekta
CP4->Print();       //izsaukums no dinamiska objekta
```

Metožu izsaukumi (paplašinātā sintakse)

```
CP1.CoordPoint::Print();     // <objekts>.<klase>::<metode>
(*CP4).CoordPoint::Print();  // <objekts>.<klase>::<metode>
CP5->CoordPoint::Print();    // <objekts>.<klase>::<metode>
```

Inicializatoru izmantošana

```
CoordPoint::CoordPoint() : X(0), Y(0) {
}
```

Norāde uz sevi

```
void SetX(int X) {
   this->X = X; // atribūta un parametra vārds sakrīt
}
```

Iegultās (inline) funkcijas

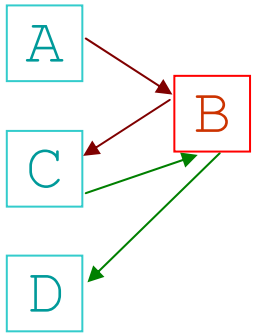
```
class CoordPoint {
   ...
   void SetX(int X) {          // funkcija definēta
      this->X = X;             // klases CoordPoint iekšā
   }
};

inline void CoordPoint::SetY(int Y) { // "inline"
   this->Y = Y;                       // definīcijā
}
```

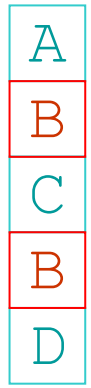
Iegulto funkciju efekts

Lai A, B, C, D ir koda bloki. Ir darbību secība: A, B, C, B, D.

B ir *"parasta"* funkcija.



B ir *"iegultā"* funkcija.



Atribūtu aizsardzība no izmaiņām

```
int GetX() const {
    // Y = 2; // Kompilācijas kļūda ! Y ir atribūts
    return X;
}
```

Funkcija netiks noformētā kā iegultā, ja...

1. Funkcijā ir *cikli*.

```
inline void ClearBuffer() {
   while (kbhit())
      getch();
}
```

```
Warning TEST.CPP 43: Functions containing while are not expanded inline
```

2. Funkcijā ir operators *goto*.

```
inline void GetChar() {
   char c=getch();
   if (c=='n')
      goto n;
   ...
}
```

```
Warning TEST.CPP 45: Functions containing goto are not expanded inline
```

3. Dažos citos gadījumos.

Darbs ar rakstzīmju virknēm

```
#include <string.h>
#include <cstring.h>
...
char* S1;
char S2[20];
string S3;     //klase no cstring.h

...
S1 = "C++";
strcpy(S2, "C++");
S3 = "C++";

...
cout << "TEXT:   " << S1 << " " << S2 << " " <<
   S3 << endl;
cout << "LENGTH: " << strlen(S1) << "    " <<
   strlen(S2) << "    " << S3.length() << endl;
...
```