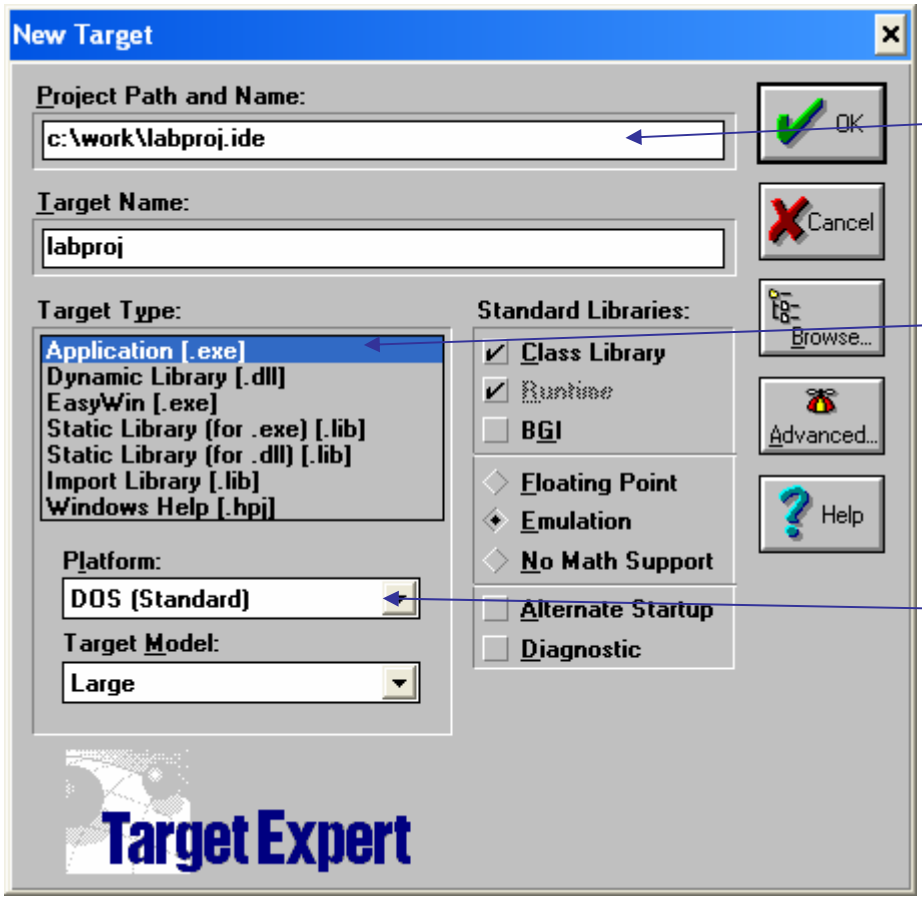
Moduļi un projekts. Draugi. Operatoru pārlāde

1. Projekta radīšana: *Project → New project...*



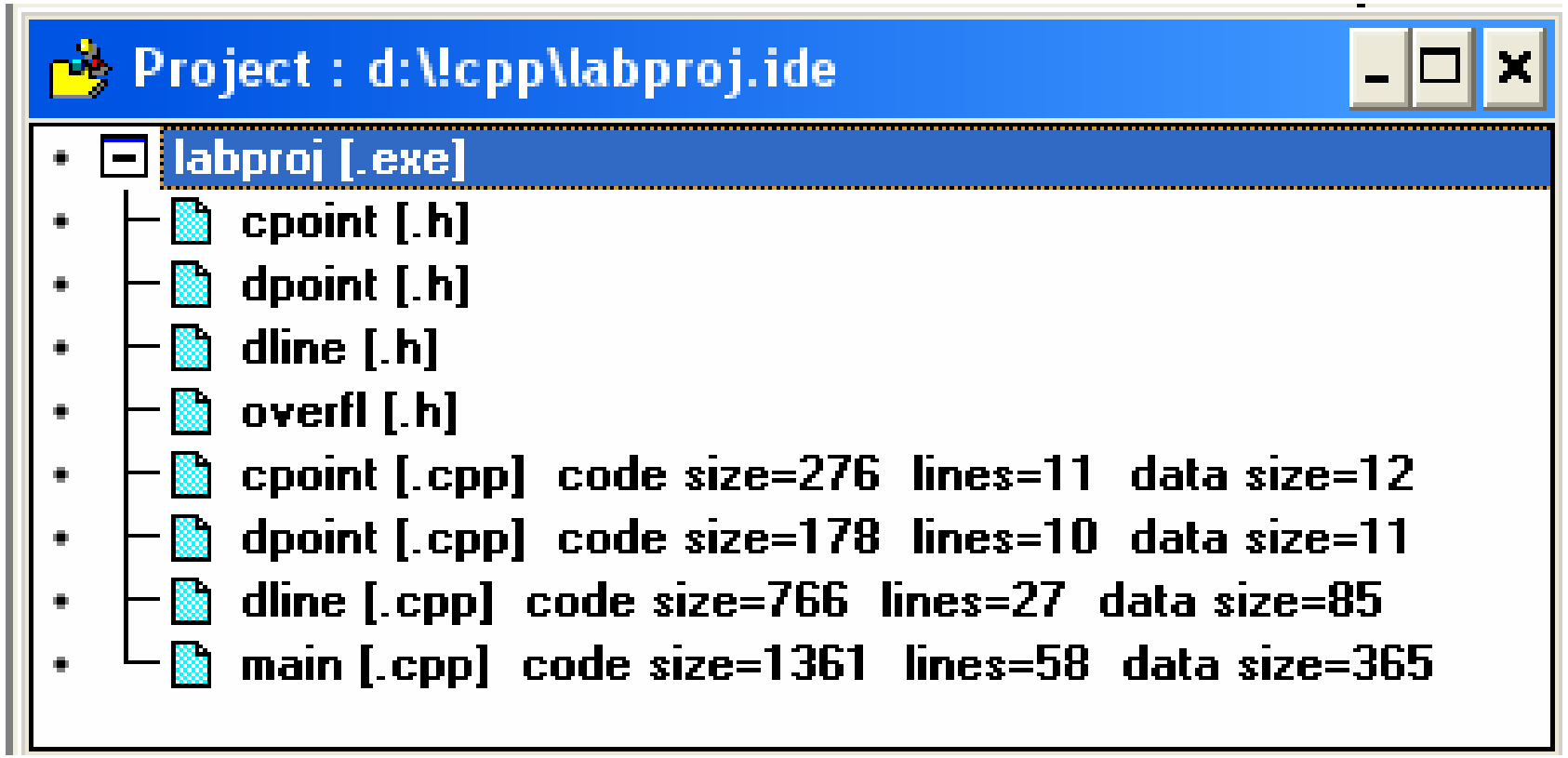
Projekta vārds

Rezultāts: izpildāmais fails

Projekta platforma

2. Projekta atvēršana: *Project → Open project...*

Atvērt failu ar paplašinājumu **.ide*. Piemēram, *LabProj.ide*.



Klases *CoordPoint* virsraksta fails *CPoint.h*.
Nav *Print()* metodes.

```
#ifndef CPointH
#define CPointH
class CoordPoint {
   ...
   friend ostream& operator << (ostream& O,
      const CoordPoint& CP);
};
#endif
```

Preprocesora direktīvas aizsargā programmu no *atkārtotas* moduļa pieslēgšanas. Elementārais piemērs:

```
#include “cpoint.h”
#include “cpoint.h”
```

Bez norādītas aizsardzības notiks kompilācijas kļūda

Komentāri klases *CoordPoint* deklarācijai

ostream ir *klase* (fails *<iostream.h>*)

cout ir **ostream** klases *objekts*

```
#include <iostream.h>
...
ostream& MyOut=cout;          //norāde uz "cout"
MyOut << "Hello, world!"; //Hello, world!
```

Operators-draugs ir *brīva* funkcija.

Iespējamie draugi:

1. Brīvas funkcijas.
2. Klases.
3. Klašu metodes.

Virsraksta faila pieslēgšana failā *CPoint.cpp*

```
#include <iostream.h>
#include "CPoint.h"
...
ostream& operator << (ostream& O,
   const CoordPoint& CP) {
      O << "X = " << CP.X <<
         ", Y = " << CP.Y;
      return O;
}
```

Bez *friend*-deklarācijas notiktu divas kompilācijas kļūdas: X un Y nav pieejami.

Alternatīvais risinājums: operatoru **ne**deklarē klases iekšā kā *friend*-funkciju.

```
O << "X = " << CP.GetX() <<
   ", Y = " << CP.GetY();
```

Operatora-drauga << deklarēšana *Print()* metodes vietā

Fails *DLine.h*

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

class DisplayBrokenLine {
    ...
    friend ostream& operator <<
       (ostream& O, const DisplayBrokenLine& DBL);
};
```

Pārlādētajā operatorā **<<** (*DisplayBrokenLine* izvadei) tiks pielietots cits pārlādētais operators **<<** (*DisplayPoint* izvadei).

Operatora-drauga << realizācija
Fails *DLine.cpp*

```
ostream& operator << (ostream& O,
    const DisplayBrokenLine& DBL) {
    O << "\nLine Color: " << DBL.LineColor <<
        ". Nodes:" << endl;
    for (int i=0; i<DBL.Length; i++) {
        O << (i+1) <<
        ". " << *(DBL.Nodes[i]) << endl;
    }
    return O;
}
```

Operatora-drauga << pielietošana
Fails *Main.cpp*

```
DisplayBrokenLine *DL =
    new DisplayBrokenLine(2, 3);
...
cout << *DL;
```

Rezultāti:

```
Line Color: 3. Nodes:
1. X = 10, Y = 11, Color = 12
2. X = 13, Y = 14, Color = 15
```