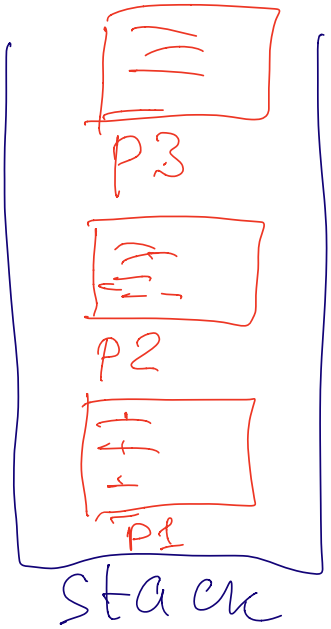
Compiler מייצר "אוטומטית"

```
class Person {
    char name[60];
    int id;

public:
    Person() {}                          def ctor
    ~Person() {}                         dtor
    Person(const Person & other)         cctor
    {
        id = other.id;
        strcpy(name, other.name);
    }
    ...
};

int main() {
Person p1;
Person p2(p1);
Person p3 = p1;
...
}
```

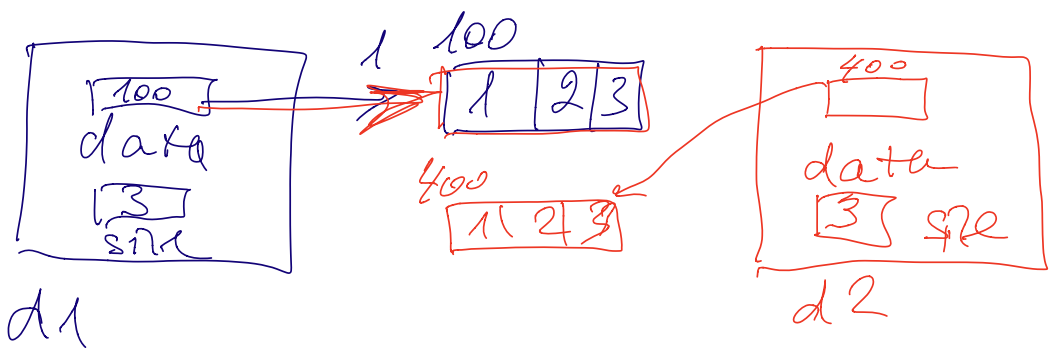


```
class DynArray {
    int * data;
    int size;
public:
    ...
    DynArray(const DynArray & sec) {
        size = sec.size;
        data = new int[size];
        - - -
    }
    ~DynArray() {delete[] data;}
};
```

Copy Ctor

```
Name(const Name &);
```

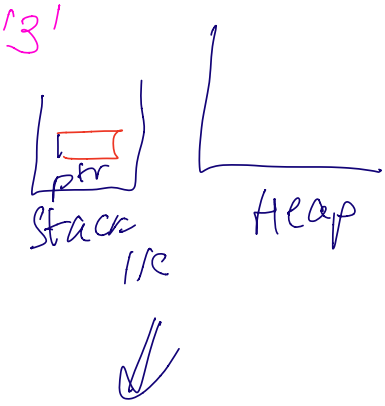
```
DynArray d1(...), d2(d1);
```



~~dual pointing~~
~~הצבעה כפולה~~

יצירה אוטומטית פונים

```
{
Point p1, p2(1,4);
Point * ptr;
ptr = new Point(1,5);
}
p1.Print();
(*ptr).Print();
delete ptr;
```

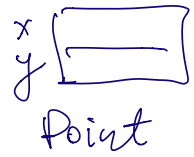


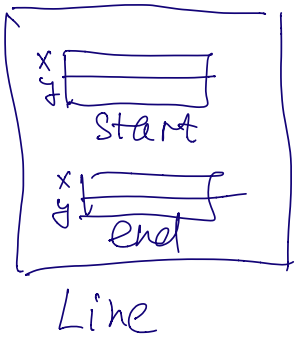

```
Point * ptr= new Point(1,5);
// ptr->Print();
```

Composition

Line **has** a point

```
class Point { int x, y; ...}
class Line {
    Point start, end;
    ⋮
};
```



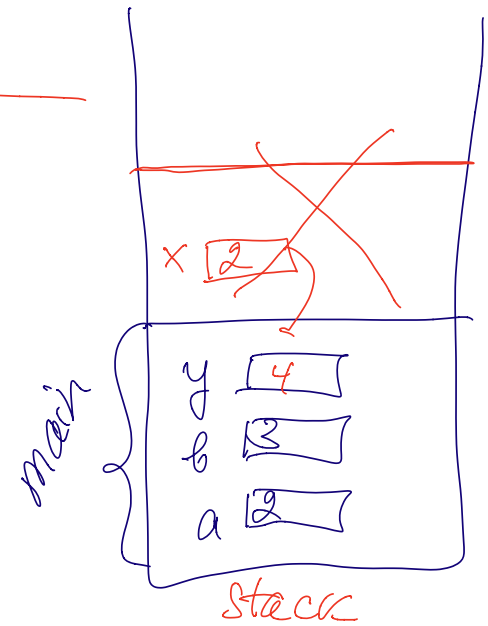


inline

בקשה מה- Compiler
להעתיק קוד הפונקציה במקום לקרוא
לה ישירות (דרך המחסנית)

```
int square (int x)
{     return  x * x; }
```

```
int main () {
    int a=2, b=3, y;
✓   y= square (a);
✓   y= square (b);
}
```




```
inline int square(int x)
{ return  x * x; }
int main () {
    y= square (a);  ⇒  y = a * a;
    y= square (b);  ⇒  y = b * b;
}
```

```
class CAccount {
    .
    .
    .
    float getBalance()
    { return balance; }        //inline implicit
};

inline void CAccount::getBalance()
{ return balance; }            //inline explicit
```

מיבים לכתוב אותם ב.h כדי
שה-Compiler יצא להחליף (לפרוש)
את הפונקציה במקום קריאה אליה
בשלב הקומפילציה.

```
    .
    .
    .
CAccount c1, c2(...);
cout << c1.getBalance();
```

⇓

```
cout << c1.balance;
```