9. Масиви от указатели към обекти.
Move семантика - ползи, lvalue, rvalue, move cc, ОП =, std::move

---

Масиви от указатели

Има 3 начина за създаване на масиви:

I статичен масив

- Compile time се решава неговия размер
- високо locality => бързодействие
- размерът му не може да бъде променян

Пр.

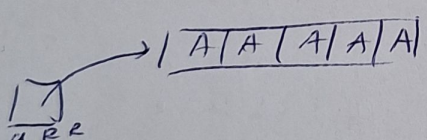
```
A arr[10];
```

II ~~Дина~~ динамичен масив от обекти (dynamic)

- размерът му може да се променя (delete, new)
- високо locality

Пр.

```
A* arr = new A[5];
```



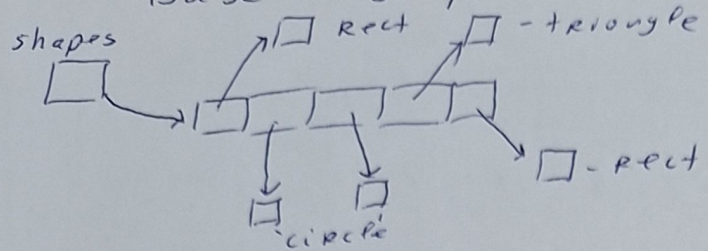
- трудно заемане на елементи
- при уголемяване се налага resize(), изтриване на цялия масив и създаване на нов

III Масив от указатели към обекти

- много лесно ~~[illegible]~~ swap-ване на елт.
- ниско locality => бавно последователно четене
- използване на функцията от def A(), подържа nullptr (празни елт.)

Пр:

Base** shapes = new Base*[10];



Move семантика

Полза:

Имаме обект, който е на края на своя жизнен цикъл. Искаме да преместим ресурсите му в друг обект и да го "откачим" от конструкцията.

lvalue
- референтни типове
- име на променливи, ф-ции, обекти
- променливи, които бихме поставили от лявата страна на равното

rvalue
prvalue
- неименувани променливи
- литерали - 73, "ABC", nullptr

xvalue
- expiring value
- обекти, които са на края на своя жизнен цикъл

rvalue = prvalue + xvalue;

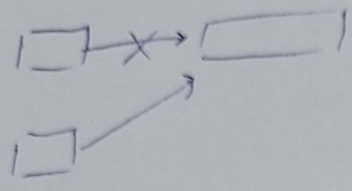
Пр: f(int&& n)
↑
rvalue ref

f(3); ✓
int n = 3;
f(n); ✗

f(int n) - lvalue, rvalue
f(int& n) - lvalue
f(const int&) - lvalue & rvalue

Move CC
- конструктор за откраднване на данните

Диаг.



Пр: ~~string& operator=(string&& oth)~~

```
string(string&&){
    moveFrom(std::move(oth));
}
void moveFrom(string&&){
    _data = oth._data;
    oth._data = nullptr;
}
```

Move OP=

Диаг:

```
string& operator=(string&& oth){
    if(this != &oth){
        free();
        moveFrom(std::move(oth));
    }
    return *this;
}
```

std::move
- преобразува lvalue във xvalue

Пр:

```
string s1 = "ABC";
string s2(s1);          // CC
string s3(std::move(s1)); // move CC
string s4(s1);          // undefined behaviour
```

⚠ Използване на std::move във ф-ции

f(A&& obj) {

...

}

във scope-а на ф-цията obj се третира като lvalue, тъй като не отвън му е подадено rvalue.

=> g(std::move(obj));

⚠ noexcept

~~при~~ runtime ако ф-цията throw-не извиква std::terminate

Пр:

A(A&&) noexcept;

A& operator=(A&&) noexcept;

- използва се за оптимизации от компилатора