Тема 9.

Масиви от указатели към обекти.
Move-семантики - плъзги, lvalue, rvalue, move констр./оп. =, std::move
Пример за клас String с move семантика

Масиви от указатели към обекти

→ няма нужда от default-ен конструктор
→ позволяват се празни позиции (nullptr)
→ бързи размени (само пренасочваме указателите)
→ resize не създава нови обекти от А (не прави копия)

→ три масиви от обекти има locality, използваме това, че са подредени последователно в паметта за по-бързо четене и итериране (губим locality при колекциите)

Move семантики - не се правят излишни копия, а се "крадат" данните на друг обект, пести време и памет

Типове данни

1. prvalue - литерали (7, false, nullptr), извикване на функция, която връща копие
2. xvalue - обекти, чийто жизнен цикъл е към края си

prvalue + xvalue = rvalue

3. lvalue - име на съществуваща променлива/функция, извикване на функция, която връща референция

```
f(int a)  // lvalue и rvalue
f(int& a) // lvalue
f(const int& a) // lvalue и rvalue
f(int&& a) → rvalue референция - не прави копие, директно краде данните с метсру
```

Move конструктор и оператор =

→ местим ресурси от временни обекти, вместо да правим копия
→ приемат rvalue ref., която не е const

std::move → преобразува lvalue в rvalue
→ декларираме, че от тук-нататък lvalue можем да "крадем", защото няма да се използва след функцията

→ A&& се третира като lvalue, докато не му приложим std::move

Голямата 4 + move констр. + move op= ⇒ Голямата шестица