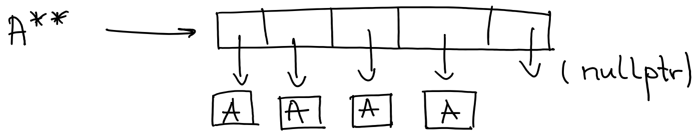
Тема 9

Tuesday, July 2, 2024 11:48 AM

Тема 9

9	Масиви от указатели към обекти. Move семантики - ползи, lvalue, rvalue, move конструктор/оп=, std::move. Пример за клас стринг с move семантика.

Масиви от указатели към обекти



Плюсове: + Не ни трябва def. к-р на A
+ Позволяват се празни позиции (с nullptr)
+ Бързи swap-ове
+ resize не създава нови обекти

Въпреки това по-често се използват масиви от обекти (A*) заради locality - обектите са на съседни адреси (итерацията и четенето стават по-бързо)

Move семантики

- открадваме данните вместо да ги копираме => пестим време и памет

Типове данни:

lvalue - име на съществуваща променлива | функция
- извикване на ф-я, която връща реф.

rvalue = prvalue + xvalue

prvalue: литерали, извикване на ф-я, връщаща копие

xvalue: expiring value - обект, който е накрая на жизнения си цикъл

f(int a) - lvalue + rvalue
f(int& a) - lvalue
f(cons int &a) - lvalue + rvalue
f(int&& a) - rvalue   && - rvalue ref

Когато подаваме обект по rvalue референцията, функцията очаква подадената променлива | константа да не се използва повече след ф-ята което значи че те свободно

Когато подаваме обект по rvalue референцията, функцията очаква подадената променлива/константа да не се използва повече след ф-ята, което значи, че тя свободно може да открадне данните ѝ

std::move – преобразува lvalue в rvalue, за да можем да крадем от вече създадени обекти
– деклариране, че от подаденото lvalue няма да се използва след ф-ята

```
MoveFrom( A&& other) {
        data = other.data;
        other.data = nullptr;
  }
```

move конструктор

```
A( A && other) noexcept {
   moveFrom( std::move(other));
  }
```

move оператор =

```
A& operator =( A&& other) noexcept {
      if( this != &other)
        {
         free();
         moveFrom(std::move(other));
        }

      return *this;
    }
```