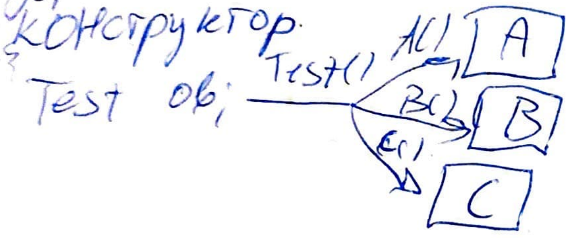
Тема 6 | Динамична памет в класовете. Голямата четворка.

Динамична памет в класовете (външни ресурси)

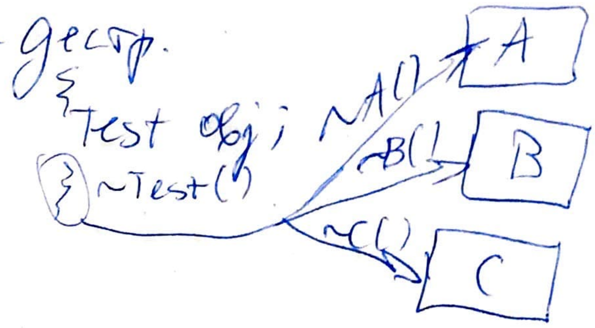
- външна за класа памет, композирана, класа да се грижи за създаване и изчистване
- голямата четворка се създава автоматично за ∀ клас

⚠ проблем с автоматично генерирана голяма 4-ка в клас с дин. памет.

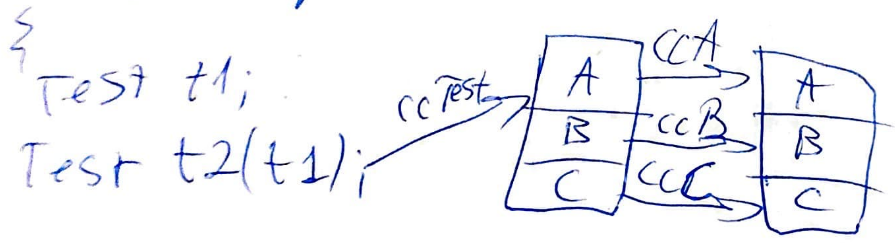
- ~~дефо~~ конструктор



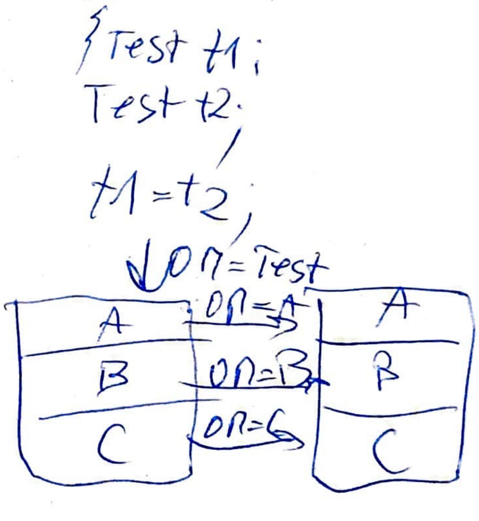
- дестр.



- копи. констр.



- OП=

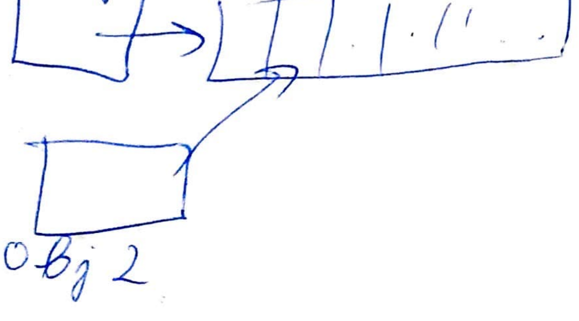


⚠ При к.к. и OП= има проблем

```
class A{
    char* str
}
```

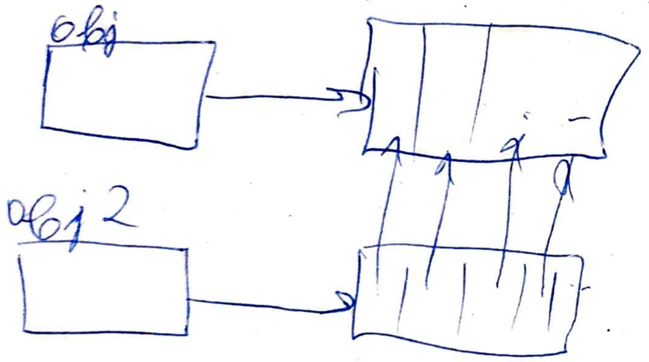
A obj
A obj2=obj;

Диаграма



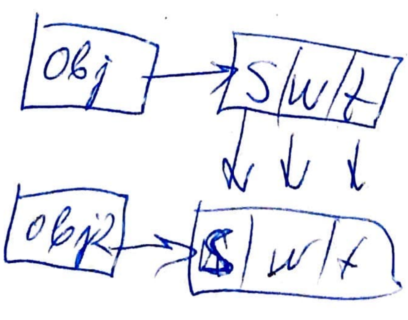
двата обекта използват една и съща външна памет, което?

{ A obj;
A obj2=obj;
③
↖ при ~A на obj2 ще хвърли runtime_error, защото паметта вече не съществува и няма как да бъде изтрита. Този проблем се нарича shallow copy.
Решение: Трябва да се направи deep copy на данните на obj в obj2, налага се да промени КК и ОМ=



Голяма четворка.
-разписваме я винаги когато имаме "външни" ресурси.

1) def()
-заделяне на външен ресурс
-инициализиране на променливи
2) KK()
-трябва да направи deepCopy на данните на външния обект и да ги запише в настоящия
-вика се memcpy на примитивни
{
A obj;
A obj2(obj)
}
-копиране



3)ОП=
- освобождаване (зачистване на сегашния обект) и копиране на данните на другия обект

4) ~Destr()
- трябва да се погрижим за паметта заделена от външни ресурси

[illegible]

Пример за голяма четворка:

GraduatedStudent.h

```
pragma once
class GraduatedStudent
{
    char* name=nullptr;
    int* grades=nullptr;
    size_t gradesCount=0;
    char quote[30+1]="";

    void copyFrom(const GraduatedStudent& other);
    void free();
  public:
    GraduatedStudent(const char* name, const int* grades,
                     size_t gradesCount, const
                                  char* quote);

    GS(const GS& other);
    GS& operator=(const GS& other);
    ~GS();
    void setName(const char* newName);
    void setGrades(const int* newGrades, size_t
                                 newGradesCount);
    void setQuote(const char* quote);
    const char* getName() const;
    const int* getGrades() const;
    unsigned getGradesCount() const;
    const char* getQuote() const;
};
```

```
include "GraduatedStudent.h"                GraduatedStudent.cpp
pragma warning(disable: 4996)
void GS::copyFrom(const GraduatedStudent& other)
{
    name = new char[strlen(other.name)+1];
    strcpy(name, other.name);
    gradesCount = other.gradesCount;
    grades = new int[gradesCount];
    for(sizet i=0; i< gradesCount; i++)
            grades[i] = other.grades[i];
strcpy(quote, other.quote);
}
void GS::free()
{
    delete[] name;
    delete[] grades;

    name = nullptr;
    grades = nullptr;
    gradesCount = 0;
    strcpy(quote, "");
}
GS::GS(const char* name, const int* grades, size_t grCount, const char* quote)
{
    setName(name);
    setGrades(grades, grCount);
    setQuote(quote);
}
GS::GS(const GS& other){
    copyFrom(other);
}
```

```
GS& GS::op=(const GS& other)
{
    if(this!= & other)
    {  free();
       copyFrom(other);
    }
    return *this;

}

GS::~GS()
{
   free();
}
void  setName(const char* newName)
    {  if(!newName || newName==name)
              return;
    delete[]name;
    name=new char[strlen(newName)+1];
    strcpy(name , newName);
}
     GS::
void setGrades(const int* newGrades, size_t newGrCount)
{
    if(!newGrade || newGrades==grades)
          return;
    delete[]grades;
    grCount=newGrCount;
    grades=new int[grCount];
    for(size_t i=0;i<grCount;i++)
        grades[i]=newGr[i];
```

```
oid GS::setQuote(const char* quote)    GraduateSt.cpp
                                       № 2
{
    if(!quote || strlen(~~new~~Quote) > 30)
        return;

    strcpy(this->quote, quote);
}
                GS::
const char* getName() const
{
    return name;
}
const int* GS::getGrades() const
{
    return grades;
}
const char* getQuote() const
{
    return quote;
}
(unsigned getGradesCount() const
{
    return grCount;
}
```