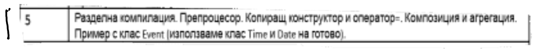
Тема 5

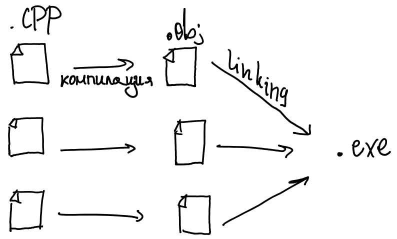
Tuesday, July 2, 2024 7:42 PM

Тема 5



5	Разделна компилация. Препроцесор. Копиращ конструктор и оператор=. Композиция и агрегация. Пример с клас Event (използваме клас Time и Date на готово).

Разделна компилация



- изходните файлове (.cpp) се компилират независимо един от друг
- в резултат на компилацията се получават .obj файлове, представляващи машинен код
- изпълнимия код на програмата (.exe) се получава след linking-а - свързване на обектните файлове от linker-а

- По този начин при промяна по един от файловете е нужно да се прекомпилира само съответния файл, а не целия проект

forward декларация - деклариране функция, при което "обещаваме", че компилаторът ще намери имплементацията ѝ при linking-а

#pragma once - казва на компилатора, че дадения header файл трябва да се обработи само веднъж

Стъпки при компилация

1) Препроцесор - "обработка на стрингове" - изпълняват се всички директиви, задаващи някаква текстова обработка

#include - замества реда със съдържанието на даден файл

#include " " - търси локално
#include< > - търси в стандартната библиотека

#define - замества един стринг с друг в целия код

макроси - замества код с друг код

2) синтактичен анализ
3) семантичен анализ
4) междинна оптимизация
5) assembly code
6) машинен код ——— край на компилационния процес

7) линкване

Композиция и агрегация

- взаимоотношение между обектите

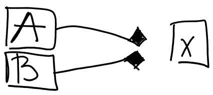
1. Композиция

class X {

Жизненият цикъл на A, B се контролира от X

1. Композиция

```
class X {
    obj;
  B obj;
};
```

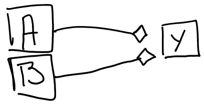


Жизненият цикъл на A, B се контролира от X

2. Агрегация

```
class Y {
  A* a;
  B& b;

};
```



A, B живеят извън рамките на Y

Копиращ конструктор - приема об т от същия клас и текущият става негово копие
Ако не го разпишем експлицитно, компилаторът създава такъв, независимо дали друг к-р е създаден.

Default-ният к.к.:
→ прави копие на примитивните член-данни.
→ извиква к.к. на инстанциите/обектите в класа

Извикване:

```
              f(A obj);
A obj
A obj2(obj); //k.k.
f (obj)      //k.k.
A obj3 = obj;
```

Синтаксис:

```
X {
 A obj1;
 B obj2;
}
```

```
X (const X& other):obj1(other.obj1),
                   obj2(other.obj2)
{ }
```

Оператор = - семантиката е същата като при к.к., но тук модифицираме вече съществуващ обект
Ако не го разпишем компилаторът автоматично създава такъв

1. изчиства текущи данни
2. копира

- оператор = извиква op= и на членовете му (щом текущият обект е съществуващ, то и членовете му са съществуващи

Извикване:

```
X obj1;
X obj2;
obj1 = obj2
```

Синтаксис:

```
X& operator = (const X& other)
{
    if(this != &other)
    {
        obj1 = other.obj1;
        obj2 = other.obj2;
```

```
{
    obj1 = other.obj1;
    obj2 = other.obj2;
}
 return this*;
```

- RVO - return value optimization - спестява правенето на копия

```
A create A()
 { return A(); }

   A obj = create A(); //не се вика копиращ к-р.
```

- NRVO - named RVO - когато обектът е създаден преди return-a