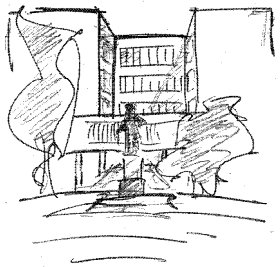
Развој софтвера 7



Саша Малков

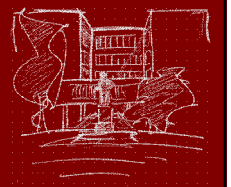
Универзитет у Београду
Математички факултет

2023/2024

[P290]

Развој софтвера

Саша Малков



Тема 11

Параметарско полиморфизам

Полиморфизам

- Елемент програма је ***полиморфан*** ако може да се односи на више различитих типова
  - елемент програма може да буде израз, вредност, функција, метод, класа, библиотекам, образац пројекта...
- Пример:
  - a = b + c;
  - Који је тип сваке од променљивих?
  - Који је тип сваког од оператора?
  - Који је тип целог израза?

Универзитет у Београду - Математички факултет

Проверавање типова

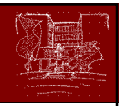
- Ако посматрамо израз: a = b + c
- Шта знамо о њему?
  - постоји неки бинарни оператор „+“, типа: $T1 \times T2 \rightarrow T3$;
  - име *b* је типа *Tb*, за који важи да је подтип типа *T1*, или постоји имплицитна конверзија из типа *Tb* у тип *T1*;
  - име *c* је типа *Tc*, за који важи да је подтип типа *T2*, или постоји имплицитна конверзија из типа *Tc* у тип *T2*;
  - постоји бинарни оператор додељивања, који променљивој типа *TV* додељује вредност типа *TE*;
  - *T3* је подтип типа *TE* или постоји имплицитна конверзија типа *T3* у *TE*;
  - име *a* је типа *Ta*, који је подтип типа *TV*
    - зависно од језика, можда је неопходно да је *a* типа *TV*

Универзитет у Београду - Математички факултет

Проверавање типова (2)

- Из претходног примера видимо да проверавање типова може бити сложен посао чак и за веома једноставне изразе
- Ствари се додатно компликују у контексту полиморфизма

Карактеристике полиморфизма



- Однос између универзалног и променљивог дела
- Начин и тренутак проверавања сагласности типова
- Статичко и динамичко везивање
- Технички аспект имплементирања

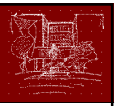
Карактеристике полиморфизма (2)

- **Однос између универзалног и променљивог дела**
  - универзалан део је основ полиморфизма
    - оно што је исто за све типове
  - променљиви део је основ разликовања
    - оно што се разликује за неке или сваки од типова
- Начин и тренутак проверавања сагласности типова
- Статичко и динамичко везивање
- Технички аспект имплементирања

Карактеристике полиморфизма (3)

- Однос између универзалног и променљивог дела
- **Начин и тренутак проверавања сагласности типова**
  - код слабо типизираних језика се одвија у фази извршавања
  - код строго типизираних језика се одвија у фази превођења
  - може и подељено – при превођењу се провери исправност, а при извршавању се одређују конкретни типови
- Статичко и динамичко везивање
- Технички аспект имплементирања

Карактеристике полиморфизма (4)

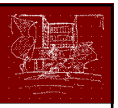


- Однос између универзалног и променљивог дела
- Начин и тренутак проверавања сагласности типова
- **Статичко и динамичко везивање**
  - статичко је одабир имплементације (метода) у фази превођења
  - динамичко је одабир имплементације у фази извршавања
    - подразумева да се употреба преводи статички, а одабир динамички
- Технички аспект имплементирања

Карактеристике полиморфизма (5)

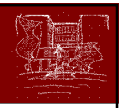
- Однос између универзалног и променљивог дела
- Начин и тренутак проверавања сагласности типова
- Статичко и динамичко везивање
- **Технички аспект имплементирања**
  - свака врста полиморфизма има своје карактеристике
  - утиче и врста и начин имплементирања програмског језика

Врсте полиморфизма



- Хијерархијски полиморфизам
- Параметарски полиморфизам
- Имплицитни полиморфизама
- Ад-хок полиморфизам

Врсте полиморфизма (1)



- **Хијерархијски полиморфизам**
  - основ ОО програмирања
  - универзалан део је интерфејс базне класе
  - променљив део је имплементација (виртуалних) метода
  - везивање метода мора да се изводи динамички
- Параметарски полиморфизам
- Имплицитни полиморфизама
- Ад-хок полиморфизам

Врсте полиморфизма (3)

- Хијерархијски полиморфизам
- **Параметарски полиморфизам**
  - почива на примени концептуалних интерфејса
    - исти су принципи, али не и конкретни типови
  - универзалан део је алгоритам, тј. употреба симболичких типова и вредности
  - променљиви делови су конкретни типови или вредности
  - променљиви делови су и потенцијално различите дефиниције за различите типове
    - практично се захтева да је подржан ад-хок полиморфизам
  - назива се и ***генеричко програмирање***
  - нема посебних предуслова у односу на начин проверавања типова и везивања
    - код строго типозираних језика уобичајено је да се преводи статички
- Имплицитни полиморфизама
- Ад-хок полиморфизам

Врсте полиморфизма (4)

- Хијерархијски полиморфизам
- Параметарски полиморфизам
- **Имплицитни полиморфизама**
  - у основи је уопштење параметарског
  - али нема експлицитне нотације типова
    - осим у делу где се захтевају различите дефиниције за различите типове
- Ад-хок полиморфизам

Врсте полиморфизма (5)

- Хијерархијски полиморфизам
- Параметарски полиморфизам
- Имплицитни полиморфизама
- **Ад-хок полиморфизам**
  - дефинише се више функција (метода, оператора) са истим именом али различитим типовима аргумената
  - преводилац према контексту бира дефиницију
  - може да се надопуњује са параметарским и имплицитним полиморфизмом
    - вид представљања променљивог дела понашања

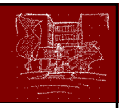
C++

- Јединствен по томе што подржава све видове полиморфизма
  - хијерархијски полиморфизам
    - ОО хијерархије
  - параметарски полиморфизам
    - шаблони функција и класа
  - имплицитни полиморфизам (делимично)
    - “тип” *auto*, макрои
  - ад-хок полиморфизам
    - вишезначна имена функција

Параметарски полиморфизам

- Суштински се разликује од хијерархијског
  - нуди много већу слободу
  - као да се праве посебан интерфејс и посебна хијерархија за сваки аспект понашања
- Нема егзактне него концептуалне интефејсе
  - на пример, концептуални интерфејс бинарне операције је да од два аргумента истог типа рачуна резултат истог типа
  - али тип ни на који начин није унапред ограничен
- Има сличности са хијерархијама које у основи имају апстрактан интерфејс и само непосредне наследнике

Шаблони функција и класа



- Имплементација параметарског полиморфизма
- Омогућавају писање функција и класа које могу да раде са великим бројем различитих типова

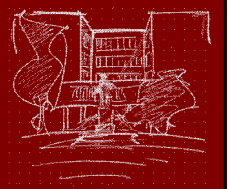
Концепти

- Основни проблем са шаблонима је њихово превођење
  - у суштини се преводе онда када се користе и за типове за које се користе
  - неке грешке се касно уочавају – тек при конкретој употреби
  - поруке о грешкама су криптичне
- То се решава *концептима*
  - Концепти имају улогу *интерфејса* за шаблоне, као што апстрактна класа представља интерфејс хијерархије
  - Пре превођења се проверава да ли параметри задовољавају услове концепта и одмах се стаје ако не задовољавају
  - Пре примене се проверава да ли тело шаблона задовољава услове концепта и пријављује се грешка пре конкретне примене

Литература

- *Stanley B. Lippman, Josee Layoie, Barbara Moo,* ***C++ Primer, 4th ed.****, Addison-Wesley, 2005.*
- Саша Малков, **ООП- C++ кроз примере**, Математички факултет, 2007.
- *"cplusplus.com"*
  - www.cplusplus.com
- *"cppreference.com"*
  - www.cppreference.com

Хвала на пажњи!



МАТФ
Универзитет у Београду
Математички факултет