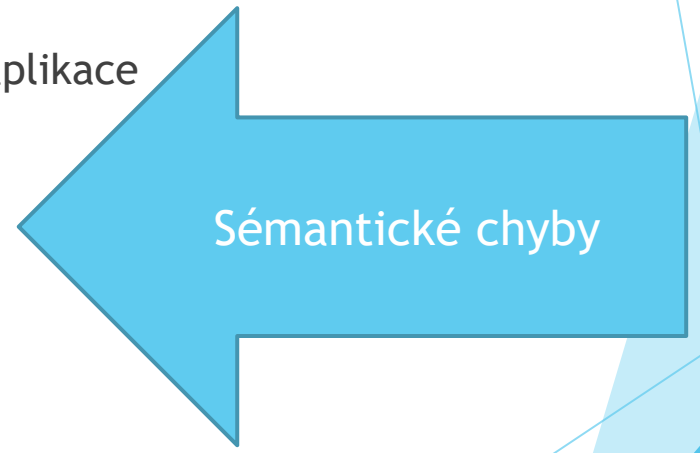
Programování

Ladění, Debugging, Testování aplikací

Chyby v programu

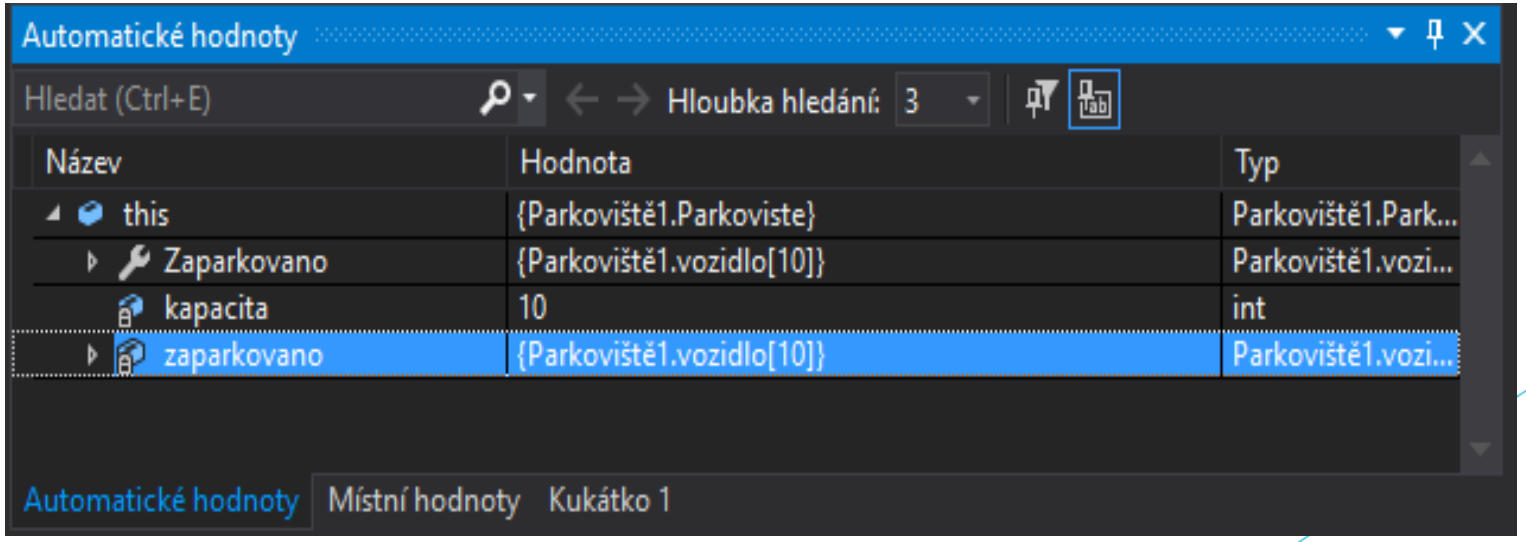
- **Syntaktické**
  - Překlepy v kódu
  - Chybějící středníky
  - Neukončené příkazy
- **Logické**
  - Program se tváří, že funguje správně, nedochází k pádu aplikace
  - Způsobeny špatným návrhem algoritmu
  - Odhalit je lze pomocí **ladění programu**
- **Běhové**
  - Program se dostává do **chybového stavu**
  - Často vzniká výjimka z běhu programu
  - Ošetření pomocí try-catch



Ladění programu (debugging)

- Pomocí ladícího nástroje (= debugger) odhalujeme logické chyby
- Během ladění si můžeme krokovat jednotlivé příkazy - krokování
- Debugger nám umožňuje sledovat aktuální hodnoty v používaných proměnných
- Ve většine IDE si můžeme nastavit místo (řádek), od kterého sledujeme změny v programu
  - Umisťujeme **BREAKPOINT** – lze přidat i podmínku ke spuštění krokování
- Alternativou k ladění je nechávat si vypisovat na výstup současné hodnoty
  - Vhodné pouze pro sledování malého množství proměnných

```
53          }
            – odkazy
54          public void VypisVozidel()
55          {
56              string parkovist = "";
57              for (int i = 0; i < kapacita; i++)
58              {
59                  if (zaparkovano[i] != null)
60                  {
61                      parkovist += zaparkovano[i].ToString() + "\n";
62                  }
63              }
64              Console.WriteLine(parkovist);
65          }
            – odkazy
```



Testování aplikací

- Ověřování správného chování aplikací podle zadané specifikace
- Ověřujeme zda se setkávají požadavky s výsledným výtvorem
- Nezbytná součást při vývoji aplikací, systémů a celkově SW

- Testování aplikací probíhá na několika úrovních
- Na nejnižší úrovní provádí testování vývojáři
- Poslední úrovní testování je uživatelské testování
  - Poskytuje nám zpětnou vazbu na naši práci a odhaluje slabá místa

Proč vůbec testovat?

- Hledání defektů, chyb v aplikací
  - V závislosti na závažnosti chyby je nutné ji prioritně řešit
- Zvyšujeme kvalitu samotného produktu a jeho atraktivnost
  - Systém s mnoha chybami si nenalezne tolik zákazníků
- Poskytuje informace pro důležitá rozhodnutí
  - Obvykle majitel případně vedoucí vývoje – kam aplikaci posunout
- Prevence chybového chování
  - Definice standardů, specifikace požadavků

Úrovně testování

- Vývojové testování
  - **Unit testing**
  - Integrační testování
  - Systémové testování
- Akceptační testování
- Údržbové testování
- Operační testování

Unit testing

- Testování na nejnižší úrovní
- Tvorbu jednotlivých unit testů zajišťuje samotný vývojář
- Testujeme jednotlivé izolované části aplikace
  - Obvykle testujeme samostatnou třídu
- Testujeme funkcionalitu jednotlivých funkcí třídy včetně vytvoření nového objektu
- V testech nesmíme opomenout i případy, kdy uživatel vkládá neplatné hodnoty nebo se snaží aplikací rozbít
  - Negativní a pozitivní testování

Integrační testování

- Testování celkového návrhu aplikace a její architekturu
- O testování se obvykle stará pověřená osoba – tester
- Testujeme komunikaci mezi jednotlivými komponentami
  - Samotné chování komponenty je ošetřeno již v unit testech
- Vytváření uživatelských scénářů ověřující funkčnost
- Testování rozhraní aplikace
  - V případě webových aplikací například API testy

Systémové testování

- Kontrola systémových požadavků a specifikací
- E2E (End-TO-END) testování
- Sledujeme celkové chování systému při běžném používání
  - Simulace uživatelského chování
- Testujeme aplikaci v reálném prostředí, abychom zajistili její správné fungování ještě před dodání zákazníkovi
- Test funkčních a nefunkčních požadavků
  - Funkční požadavky jsou dány specifikací – co má aplikace umět
  - Nefunkční požadavky – nároky na HW, operační systém, …
- Analýza případných rizik

Akceptační testování

- Testování prováděné skutečnými uživately
- Testování může probíhat kdykoliv v průběhu vývojového cyklu
  - Metodika MVP – testujeme malé funkční části
- Cílem je posílit pozici produktu na trhu
- Často se můžete setkat s pojmy alfa testování, beta testování
  - Alfa – interní testování
  - Beta – externí testování
- V závislosti pro koho je aplikace určena, tak víme, kdo dává v akceptačním testování zelenou pro vydání

Údržbové testování

- Často označováno jako regresní testování
- Cílem je odhalovat chybové nově vzniklé chybové chování, které se v předešlé verzi nevyskytovalo
- V závislosti na závažnosti lze takovéto chyby dělit do několika úrovní tzv. severit
- **Blocker, Critical, Major, Minor, Trivial**
- Pro údržbové testování je výhodné mít řešenou úplnou nebo částečnou automatizaci testů
  - Při vložení nového kódu do vývojového repositáře se spouští test – pokud testy objeví chybu nelze kód sloučit s hlavní verzí

Test Driven Development - TDD

- Jedna z možností vyvíjení nové aplikace
- V praxi není často využívaná
  - Vyžaduje větší režii a vývoje se zdá pomalejší
- Před samotným vytváření aplikace se první píšou testy
- Vývojář píše testy, které odpovídají požadavkům vzniklých ze specifikací
- Na počátku tedy všechny testy neprochází
- Vývoj pokračuje až do momentu, kdy všechny vytvořené testy prochází
  - Jistota, že jsou dodrženy veškeré náležitosti, které byly požadovány
- V momentě, kdy nám všechny testy prochází přichází na řadu refaktoring a optimalizace