Ladění programů

- Testování a ladění základní součástí každodenní aktivity programátora.
- **Ladění** je proces **hledání a redukování chyb** a defektů v programech tak, aby se programy chovaly podle očekávání.
    - Víme o chybě, známe způsob spuštění programů pro projevení chyby.
    - Systematicky hledáme moment a místo v programu, které chybu způsobuje.
- **Testování** je systematický proces **zlepšování kvality** programů.
    - Např.: Tvoříme testovací sadu, která by ukázala, v jakých případech program funguje.
    - Předmět ITS (Testování a dynamická analýza), ATA (Automatizované testování)
- **Buffer overflow**
    - Chybové přepsaní hodnot, např.
    -
```
int n_lodi = 3;
char lode[3] = {'a', 'b', 'c'};

char *p = lode;

for (int index = 0; index <= 3; index++) {
    p[index] = 'a';
}
```

- **Chyba, bug, klasifikace**
    - Chyba/defekt/selhání - obecně bug (definice později, pro tuto přednášku si vystačíme s pojmem “chyba”).
    - Základní klasifikace podle obtížnosti odhalení:

	Důsledek	Doba opravení
Syntaktické chyby	selhání	zanedbatelná
Chyby sestavení	selhání	krátká
Základní sémantické chyby	chyba základní funkcionality, selhání	krátká
Účelu specifické sémantické chyby	Bez projevu až po selhání	střední až velmi dlouhá

    -
    - Praxe: “Existují jen dva druhy chyb. Triviální a velmi, velmi obtížné.”
    - **Klasifikace chyb**
        - **Syntaktické chyby:**
            - Měly by být odhalitelné překladačem; ne vždy na místě, na které se překladač odkazuje.
                - “měly by být” = i překladač může (a obsahuje) chyby; za svůj život se ale možná s žádnou chybou překladače nebo interpretu nesetkáte
        - **Chyby sestavení:**
            - Chyby z neaktualizovaných binárních souborů (nepřeložené nové zdrojové soubory, nekompatibilita knihoven)
        - **Základní sémantické chyby:**
            - Neinicializované položky, mrtvý kód, nekompatibilita datových typů, …
        - **Účelu specifické sémantické chyby:**
            - Chyby neodhalitelné překladačem – jsou syntakticky správné, ale logicky špatné (program neodpovídá zadání, které překladač nezná)

- Špatný operátor, špatné pořadí argumentů funkce, nekontrolované meze polí, neošetřené krajní případy, přetečení, chyby paměti, off-by-one, stack-frame, chyby v rekurzi, zaokrouhlovací chyby, statická vs. Dynamická data, ...

- **Koncept (fáze) ladění**
  - Proces ladění odpovídá konceptu:
    - **1. Reprodukce (hledání vstupních dat, bug report)**
    - **2. Diagnóza**
    - **3. Oprava (bug fix)**
    - **4. Intergrace (fix test, commit)**
  - Oprava a integrace není tématem přednášky.
  - **Reprodukce chyby (kdo: reportér)**
    - **Kontrola, že chyba již nebyla opravena**, tj. Projevuje se chyba na poslední verzi?
    - **Hledání vstupních dat, které vedou k projevu** chyby:
      - "Umíme chybu vyvolat opětovně (1) na všech počítačích a (2) pro každého vývojáře?"
    - **Izolování chyby:**
      - Minimalizace počtu kroků k reprodukci chyby.
      - Minimalizace počtu vnějších parametrů.
      - Odstranění nedeterminismu (pokud se chyba projevuje náhodně, najít a odstranit všechny "náhodné" vstupy).
    - **Automatizace chyby** (vede k tzv. Unit a regresním testům)
  - **Diagnóza chyby (kdo: vývojář)**
    - **Pochopení principu chyby**, tj. Hlavní příčiny (tzv. Root cause)
      - Uspěchaná oprava chyby vede k zavedení dalších chyb
        - Chyby mají tendenci se družit,
        - Projev chyby na jistém místě neznamená, že chyba je v tomto místě.
      - Základní check-list:
        - ✓ nesplést si projev a reálnou příčinu chyby
        - ✓ podobné chyby nejsou v jiných částech kódu
        - ✓ jedná se o chybu programování, nikoliv návrhu
    - **Prozkoumání chyby:**
      - Odpovídají vstupní podmínky správnému použití?
        - Ne? Je třeba opravit požadavky použití (tj. Dokumentaci programu) →
      - Není chyba ve vnějším prostředí?
        - Jsou soubory a data správná? Knihovny očekávané verze? Architektura počítače, operační systém ok? Dostatečné prostředky (paměť, volné místo, rychlost/spolehlivost sítě, ...)?
      - Chyba je v programu?
        - 1. Minimalizace ("ukrajování") kódu.
        - 2. Krokování nejmenšího kusu kódu, ve kterém se chyba projevuje
        - 3. Hledání místa, kde chyba nastala (nemusela se projevit)

- **Techniky ladění**
  - **Statická x dynamická analýza**
    - **Statická analýza** zkoumá program průchodem a vyšetřením zdrojových kódů bez jejich spouštění
    - **Dynamická analýza** zkoumá průběh a výsledky provádění programu nebo jeho částí.

- **Ladění jako statická analýza** = code review

  - Zahrnuje dva a více vývojářů
  - Code review = programování + diskuze (neplést s pair-programming)
- **Ladění jako dynamická analýza:**
  - Záznam sekvence událostí (log, logging, “logování”)
  - Interaktivní ladění (debugging)
- **Záznam, logování:**
  - Informování o aktuálním stavu programu
  - Záznam do jiného souboru (stderr, log)
  - Log = soubor určený pro hlášení o událostech (datum a čas, původ hlášení, typ hlášení, zpráva)

- **Podmíněný překlad**
  - Úprava zdrojových kódů na základě vstupních podmínek překladu.
  - Jazyk C definuje tento proces jako preprocessing (vykonává preprocesor).
  - Preprocesor je ovlivněn direktivy překladače:
    - #include, #define, #if, #ifdef, #error, ...
  - Makro (macro) = symbolické jméno za textovou náhradu ve zdrojovém kódu
    - Může být parametrické

- **Ladění pomocí ladicích nástrojů**
  - Ladicí nástroje = nástroje pro podporu dynamického ladění programů:
    - Překladač + debug info
    - Debugger
    - Object inspector
    - Disassembler
    - Sys/library tracer
  - Debugger = interaktivní nástroj pro lokalizaci chyb:
    - Typicky součást IDE
    - Sledování proměnných (watcher)
    - Možnosti záchytných bodů (break-point)
    - Analýza stopy volání funkcí (backtrace)
    - Debugger je interaktivní nástroj pro krokování programu a sledování jeho vnitřních stavů.
    - Musí mít přístup ke všem částem analyzovaného programu:
      - zaručeno buď podporou operačního systému
      - nebo úpravou kódu před spuštěním (tzv. instrumentací kódu).