Program – algoritmus úlohy zapísaný v jazyku, ktorému „rozumie“ počítač (v programovacom jazyku)
Programovanie – činnosť - riešenie úlohy pomocou počítača
Syntax programovacieho jazyka – presný opis, ktoré symboly vytvárajú abecedu programovacieho jazyka (písmená anglickej abecedy, číslice 0 až 9, vyhradené slová begin, end, if, while, const, var, array, procedure,...) a ktoré reťazce reprezentujú správne zapísané konštanty, premenné, výrazy, príkazy, deklarácie či programy (napr. identifikátor, definovanie konštanty, tvar príkazu, unitu, procedúry,...). Rôzne programovacie jazyky majú rôznu syntax.
Sémantika – presné určenie významu vyššie uvedených prvkov.

Testovanie a ladenie programu

Po napísaní programu musíme vhodnými prostriedkami overiť jeho funkčnosť a správnosť. Táto etapa sa nazýva testovanie a ladenie. **Testovaním** zisťujeme, či nie sú v danom programe chyby. **Ladením** ich odstraňujeme, t.j. predovšetkým lokalizujeme (určíme miesto chyby) a špecifikujeme (určíme chybu).

Chyby môžu byť:
- syntaktické (syntax errors), *zistené pri kompilácii* (preklade)
  napr.: preklep v príkaze, nedeklarovaná premenná, chybný počet parametrov a pod.
- logické (chyby v algoritme), *zistené počas alebo po skončení behu programu*
  - vedúce k predčasnému zastaveniu programu s chybovou správou (run time errors)
    napr.: delenie nulou, súbor na otvorenie nenájdený a pod.
  - vedúce k chybným výsledkom, zacykleniu programu (zastavenie vykonávania programu: Run – Program Reset) a pod.

Zásady testovania:
- poznať správny výsledok
- pri konečnej množine vstupných údajov úplné otestovanie programu
- testovanie všetkých ciest (vetiev)
- testovanie hraničných a „problémových“ hodnôt

Testovanie a ladenie v Delphi:

Spustenie programu
Run, F9
Ukončenie behu programu
Run – Program Reset, Ctrl+F2
Zastavenie behu programu
Run – Program Pause
Krokovanie (trasovanie, tracing) programu
Run – Trace Info, F7
Krokovanie programu bez krokovania v podprogramoch
Run – Step Over, F8
Zastavenie na riadku s kurzorom
Run – Run to Cursor, F4
Zastavenie na určenom mieste (Breakpoints)
Run – Add Breakpoint – Source Breakpoint... (Condition – zastavovacia podmienka, hodnota premennej rovná... a pod.; Pass Count – počet priechodov bez zastavenia)
Sledovanie obsahu premenných (watching)
Run – Add Watch..., Ctrl+F5 (pozri kontextovú ponuku vo Watch List)
Zmena obsahu premenných
Run – Evaluate/Modify..., Ctrl+F7 (pozri nástroje Evaluate, Modify, Watch)

Životný cyklus programu (etapy tvorby programu):

(rovnaké kroky treba vykonať pri každej výraznejšej zmene programu)

1. **rozbor problému**
   sformulujeme zadanie problému a požiadavky na vznikajúci program
   zodpovieme na otázku **čo** treba robiť
2. **návrh riešenia**
   hľadáme riešenie zvážením poznatkov z danej oblasti (napr. výberom najvhodnejšieho algoritmu), zvážením prostriedkov na riešenie a navrhnutím vhodného spôsobu organizácie údajov - výsledkom je algoritmus
   zodpovieme na otázku **ako** sa dá daný problém riešiť
3. **realizácia**
   prepíšeme navrhnutý algoritmus do vhodného programovacieho jazyka; sem patrí aj príprava obrázkov, zvukových efektov, hudby do pozadia a pod.
4. **údržba**
   používanie softvéru a s ním súvisiace odhaľovanie a oprava skrytých chýb, prispôsobovanie softvéru meniacim sa požiadavkám používateľov, vývoj novších verzií atď.

Programová dokumentácia

Dokumentácia k programu vysvetľuje, aký problém program rieši; aké vstupné údaje a akým spôsobom treba zadať, aké výstupné údaje možno očakávať, ako program funguje a ako ho používať. Slúži ako pomôcka pri odstraňovaní chýb, pri zmene či rozširovaní programu,...
Dokumentácia sa obyčajne robí dvoma spôsobmi:

- **komentáre** v programe
- **písomná dokumentácia**

Komentáre v programe sú vysvetľujúce poznámky priamo v programe a prekladač ich ignoruje (uvádzajú sa v množinových zátvorkách { } alebo jednoriadkové za „dvojlomítkami“ //).
Písomná dokumentácia sa nachádza mimo programu a môže byť v papierovej alebo elektronickej podobe. Môže to byť **manuál** alebo **používateľská príručka**. Manuál obsahuje informácie pre programátorov, príručka pre používateľov softvéru.

Dva spôsoby vykonávania programu

Po vytvorení programu, ktorý je „obyčajným“ textom,

- **prekladač** (**kompilátor**) preloží tento text do strojového (binárneho) kódu – tento proces sa nazýva preklad alebo kompilácia, počas ktorého sa každý príkaz zmení na niekoľko strojových inštrukcií. Preložený program sa stal postupnosťou čísel – strojových inštrukcií procesora, ktoré vie procesor veľmi rýchlo vykonať (pozri v Delphi po spustení programu a vyvolaní Run - Program Pause). Vznikla plnohodnotná aplikácia. Takto pracujú prekladače Pascalu, Delphi, C++ Builder,...
- **interpretér** „číta“ program a v texte rozpoznáva jednotlivé príkazy, ktoré hneď vykonáva (interpretuje). Vykonávanie programu interpretáciou je pomalšie (musia sa neustále rozpoznávať príkazy v programe). Interpretáciu využívajú napríklad programy zapísané v Basicu, Comenius Logo a Imagine.

Programovacie metódy:

- zhora nadol – rozkladáme problém na čiastkové podproblémy až na úroveň jednotlivých príkazov
- zdola nahor – napíšeme podprogramy a spojíme do výsledného programu
- lineárny prístup – pri jednoduchých problémoch rovno píšeme program príkaz po príkaze
- zvnútra von – napíšeme a odladíme jadro programu a potom „doprogramujeme okolie“ (napr. užívateľský vstup a výstup)

- kombinovaný...

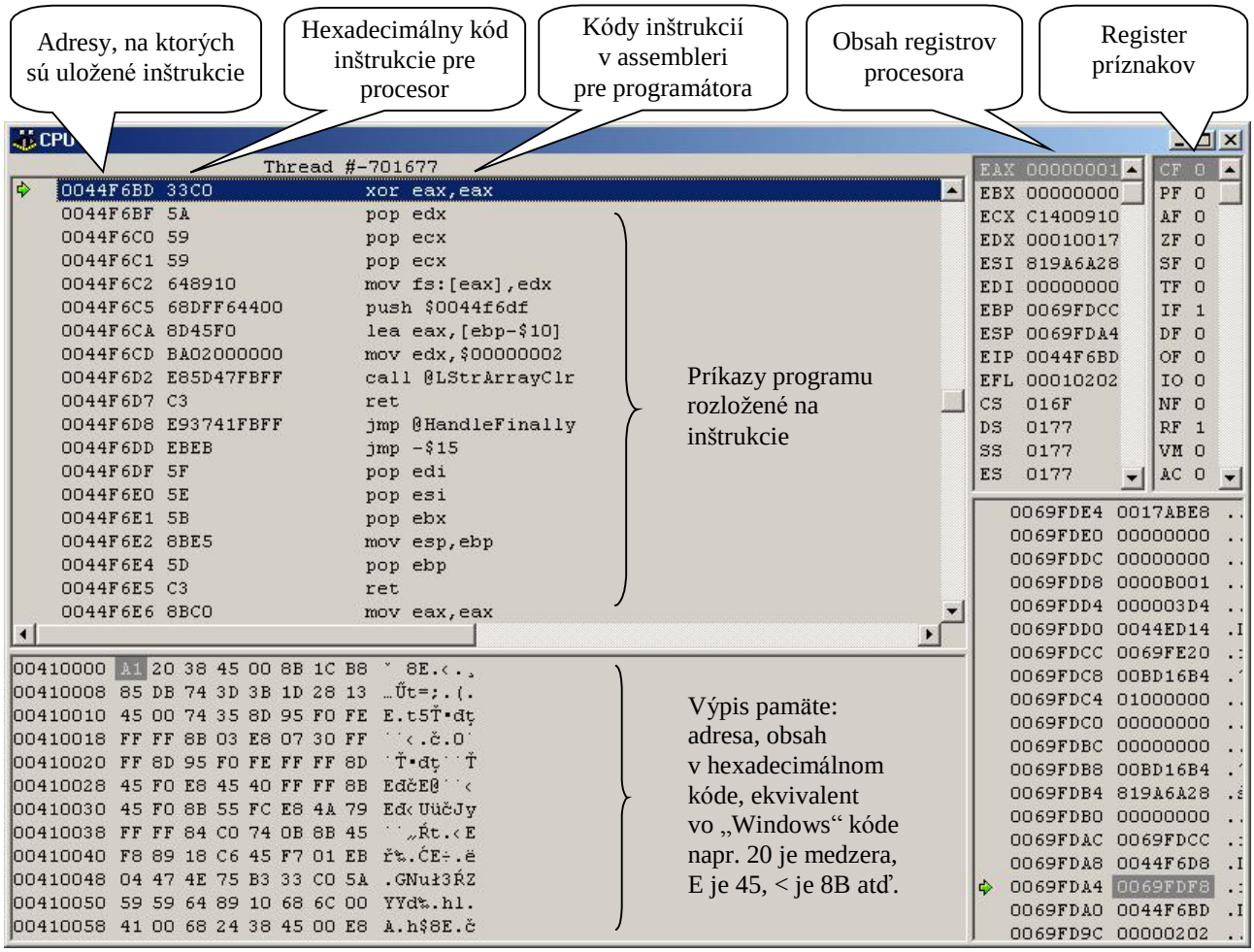
Programovacie techniky:

- štruktúrované programovanie:
  - v širšom slova zmysle:
    metóda „rozdeľ a panuj“ (rozdeľ do podprogramov a riaď ich spoluprácu)
    akceptuje najmä metóda zhora nadol
  - v užšom slova zmysle:
    obmedzenie programovacích konštrukcií len na tie, ktoré majú len jeden vstup a jeden výstup (spĺňajú všetky pascalovské konštrukcie – príkazy, okrem skoku)
- modulárne programovanie:
  - modul = unit (rozhranie - interface a implementácia t.j. realizovanie činnosti modulu)
  - rozdelenie programu na niekoľko nezávislých ale navzájom spolupracujúcich častí (jednotiek)
- objektovo orientované programovanie:
  - narábame s objektami (object) alebo triedami (class)
  - vlastnosti:
    - zapúzdrenie – spojenie dát (premenných) s obslužnými procedúrami
    - dedičnosť – objekt - potomok dedí vlastnosti predka
    - polymorfizmus – „chovanie“ procedúry sa mení podľa druhu daného objektu
- udalosťami riadené programovanie:
  - program reaguje – vykoná skupinu príkazov, na udalosť, ktorá nastala
    napr. na otvorenie formulára, kliknutie pravým – ľavým tlačidlom myši, presun myši nad komponentom a pod.
  - program najčastejšie obsahuje formulár s komponentmi u ktorých nastavujeme vlastnosti (properties) a programujeme, čo sa má vykonať pri vyvolaní určitej udalosti (events) - pri aktivovaní formulára, kliknutí na tlačidlo, zmene hodnoty v komponentovi a pod.
  - beh programu si môžeme predstaviť ako úvodnú inicializáciu (najčastejšie zobrazenie formulára s ovládacími prvkami) a čakanie na udalosť – pokyn, čo sa má vykonať ďalej
- vizuálne programovanie
  - základom aplikácie je formulár do ktorého vkladáme komponenty – objekty umiestnené v knižnici vizuálnych komponentov – Visual Component Library, sami teda fyzicky nevytvárame komponenty a ovládacie prvky, programujeme len vlastnú činnosť komponentov (event handlers – procedúry a funkcie ošetrujúce jednotlivé udalosti).
  - niektoré komponenty sú skryté počas behu aplikácie, napr. Timer.

Úlohy:
Otestujte a odlaďte priložený program.
Predveďte vytvorenie aplikácie nezávislej na prostredí Delphi.

Literatúra:
Informatika pre stredné školy (učebnica pre 1. ročník)

Príloha:
Pri zastavení behu programu v Delphi (Run – Program Pause) sa zobrazí okno zobrazujúce obsah procesora:



Príklad:
Časť výpisu pamäte počítača:

```
20706F6369746163     pocitac
312B333D322A323B    1+3=2*2;
4D41545552495441    MATURITA
                    Tic+= 23
```

Doplňte v štvrtom riadku chýbajúci hexadecimálny kód z výpisu pamäte.