1. Prvocisla: Kratky ukazkovy priklad na demonstraci baliku WEB . Nasledujici program slouzi pouze jako ukazka nekterych moznosti a sluzeb, ktere poskytuje programovaci nastroj WEB. Jak jiz vime, WEB je specialni nastroj pro tvorbu dobre dokumentovanych programu. Toto programovani nazyvame literarni programovani.

Tento ukazkovy priklad mel za vzor Knuthuv priklad, ktery pomoci baliku WEB literarne naprogramoval program na vypsani prvnich 1000 prvocisel. Zatimco Knuth naprogramoval opravdu kvalitni program, ktery prevzal od Edsgera Dijkstra z jeho knihy *Notes on Structured Programming*, ja zde predkladam pouze jednoduche priklady na toto tema, ktere kazdy jiste sam a mozna, ze i lepe naprogramuje. Programy budu rozepisovat a strukturovat vice nez pri programovani funkcnich, ale ne ukazkovych programu.

Muj ukazkovy priklad budou vlastne priklady dva: jeden na urceni, zda dane cislo je prvocislo a druhy vypise vsechna prvocisla mensi nez p. Budeme jim rikat podprogramy.

⟨ Ukazkovy program na urcovani prvocisel 2 ⟩

2. Jeden z podprogramu ma vstup a druhy ne. A cely program potrebuje na zacatku prace jednu vstupni hodnotu, podle ktere pozna, ktery z podprogramu ma pouzit pro vypocty. Musime si proto nadeklarovat konstanty a promenne.

⟨ Ukazkovy program na urcovani prvocisel 2 ⟩ ≡

```
program prvocisl;
  const p = 30;
  var ⟨ Promenne programu 4 ⟩;
    begin ⟨ Vetveni programu na dva podprogramy 3 ⟩;
    end.
```

This code is used in section 1.

3. Vyber jednoho z podprogramu. Jedna z prvnich veci, ktera se musi udelat, je rozhodnout, kterou vetvi programu se budeme dale ubirat. Jak jiz jsem se zminil, sklada se priklad ze dvou podprogramu a zalezi na uzivateli, jaky program chce zrovna pouzivat.

```
⟨ Vetveni programu na dva podprogramy 3 ⟩ ≡
  ⟨ Vstupni hodnota pro urceni vyberu podprogramu 5 ⟩;
  ⟨ Vyber podprogramu podle vstupni hodnoty 6 ⟩
```

This code is used in section 2.

4. Nadefinujeme si pomocnou promennou, ktera nam poslouzi pro nacteni vstupu a dale podle ni pozname, ktery z podprogramu se ma dale pouzit.

Pomocna promenna *vyber* bude typu word.

```
⟨ Promenne programu 4 ⟩ ≡
vyber: word;
```

See also sections 8, 10, 14, and 18.

This code is used in section 2.

5. Nebudeme nijak testovat vstupni hodnotu. Tzn. pri chybne vstupni hodnote se beh programu zastavi. Budeme predpokladat ze uzivatel vi, co chce a, nebude nijak zkouset dalsi vlastnosti programu.

```
⟨ Vstupni hodnota pro urceni vyberu podprogramu 5 ⟩ ≡
  writeln('Program na demonstraci prvocisel');
  writeln('Muzes si vybrat: zjisti zda zadane cislo je prvocislo...1');
  writeln('                 vypis prvocisel mensich nez ', p, '.........2');
  write('Tva volba: '); readln(vyber)
```

This code is used in section 3.

6. Pomoci uzivatelem zadane vstupni hodnoty program pozna, kde bude dale pokracovat ve vypoctu.

```
⟨ Vyber podprogramu podle vstupni hodnoty 6 ⟩ ≡
  case vyber of
  1: ⟨ Zjisti zda zadane cislo je prvocislo 7 ⟩2: ⟨ Vypis prvocisel mensich nez 13 ⟩
  end
```

This code is used in section 3.

7. Urci zda zadane cislo je ci neni prvocislo. Tento podprogram muzeme rozdelit na dve samostatne casti. V jedne zjistime cislo, ktere budeme dale zkoumat, a v druhe budeme provadet vlastni vypocet.

```
⟨ Zjisti zda zadane cislo je prvocislo 7 ⟩ ≡
  begin ⟨ Vstupni hodnota zkoumaneho cisla 9 ⟩;
  ⟨ Testovani zkoumaneho cisla 11 ⟩;
  ⟨ Vypis vysledku testu 12 ⟩;
  end;
```

This code is used in section 6.

8. V teto casti chceme nacist do pomocne promenne hodnotu testovaneho cisla, s kterou budeme dale pracovat.

Po cislu budeme pozadovat, aby bylo cele a kladne. Zvolime ho typu integer.

```
⟨ Promenne programu 4 ⟩ +≡
n: integer;
```

9. Vstupni hodnotu cisla nebudeme nijak testovat, predpokladame, ze je cele a kladne.

```
⟨ Vstupni hodnota zkoumaneho cisla 9 ⟩ ≡
  writeln('Program urci zda zadane cislo je prvocislo');
  write('Zadej cele cislo vetsi nez 1: '); readln(n)
```

This code is used in section 7.

10. Pri urcovani zda dane cislo je/neni prvocislo pouzijeme tuto myslenku: budeme postupovat od 2 k odmocnine z n a po otestovani cisla 2 testujeme pouze licha cisla.

Musime si dodefinovat dalsi pomocne promenne. Dve ciselne a jednu logickou.

```
⟨ Promenne programu 4 ⟩ +≡
delitel, odmocnina: integer;
jeprvocislo: boolean;
```

11. Pro vypocet budeme pouzivat ciselne pomocne promenne *delitel* a *odmocnina* a logickou hodnotu *jeprvocislo*. *delitel* bude nabyvat lichych hodnot az do *odmocnina*. Budeme jim delit zadane cislo n a pomoci zbytku po celociselnem deleni pozname, jestli je n nasobkem *delitel* a nebo neni. Pokud se *delitel* rovna *odmocnina* a zbytek po celociselnem deleni je roven 0, pak zadane cislo je prvocislo, tzn. *jeprvocislo* je *true*.

```
⟨ Testovani zkoumaneho cisla 11 ⟩ ≡
  if ((n = 2) ∨ (n = 3)) then jeprvocislo ← true
  else if odd(n) then
      begin odmocnina ← round(sqrt(n)); delitel ← 3;
      while (n mod delitel ≠ 0) ∧ (delitel < odmocnina) do delitel ← delitel + 2;
      jeprvocislo ← n mod delitel ≠ 0;
      end
    else jeprvocislo ← false
```

This code is used in section 7.

12. Nyni jiz staci jen vypsat vysledek.

```
⟨ Vypis vysledku testu 12 ⟩ ≡
  writeln;
  if jeprvocislo then writeln('Zadane cislo n = ', n, ' je prvocislo.')
  else writeln('Zadane cislo n = ', n, ' neni prvocislo.')
```

This code is used in section 7.

13. Vypis prvocisel mensich nez p. Tento podprogram na rozdil od prvniho netestuje zda zadane cislo je nebo neni prvocislo, ale vypisuje prvocisla, ktera jsou mensi nez zadane p.

Take se nyni nemusime zabyvat zadnymi vstupnimi udaji od uzivatele. Pouze mu predame vysledek.

K vypoctu pouzijeme algoritmus, ktery se take nazyva Erastotenovo sito . Pouzivame zde dve pomocne ciselne mnoziny. Jedna na zacatku obsahuje vsechna cisla od 2 do p, oznacme ji *sito* a druha je prazdna, oznacme ji *prvocisla*. Da se rict, ze z mnoziny *sito* nam do mnoziny *prvocisla* propadavaji pres sito pouze prvocisla. A budeme presivat tak dlouho, dokud mnozinu *sito* nevyprazdnime, pak v mnozine *prvocisla* budou jen sama prvocisla mensi nez p.

```
⟨ Vypis prvocisel mensich nez 13 ⟩ ≡
  begin ⟨ Inicializace pomocnych mnozin 15 ⟩;
  repeat ⟨ Hledani dalsiho prvocisla 16 ⟩;
    ⟨ Eliminace nasobku prvocisla ze sita 17 ⟩;
  until sito = [ ];
  ⟨ Vypis sita 19 ⟩;
  end
```

This code is used in section 6.

14. Musime si nadefinovat dve pomocne mnoziny kladnych celych cisel a pomocne ciselne promenne pro praci s cisly v mnozinach.

```
⟨ Promenne programu 4 ⟩ +≡
sito, prvocisla: set of 2 .. p;
dalsiprvocislo, nasobekprvocisla: word;
```

15. Dale si musime inicializovat obe pomocne ciselne mnoziny a jednu pomocnou ciselnou promennou.

```
⟨ Inicializace pomocnych mnozin 15 ⟩ ≡
  prvocisla ← [ ]; sito ← [2 .. p]; dalsiprvocislo ← 2
```

This code is used in section 13.

16. Z pomocne mnoziny *sito* budeme vybirat prvocisla a budeme je zapisovat do mnoziny *prvocisla*.

```
⟨ Hledani dalsiho prvocisla 16 ⟩ ≡
  while ¬(dalsiprvocislo ∈ sito) do dalsiprvocislo ← succ(dalsiprvocislo);
  prvocisla ← prvocisla + [dalsiprvocislo]
```

This code is used in section 13.

17. Pak uz jen staci z mnoziny *sito* odebrat vsechny nasobky posledniho prvocisla.

```
⟨ Eliminace nasobku prvocisla ze sita 17 ⟩ ≡
  nasobekprvocisla ← dalsiprvocislo;
  while nasobekprvocisla ≤ p do
    begin sito ← sito − [nasobekprvocisla]; nasobekprvocisla ← nasobekprvocisla + dalsiprvocislo;
    end
```

This code is used in section 13.

18. Tento postup opakujeme dokud neodebereme z mnoziny *sito* posledni cislo a pak uz mame v mnozine *prvocisla* vysledek, ktery muzeme vypsat.

K tomu pouzijeme novou pomocnou promennou i, ktera bude postupne nabyvat hodnot od 2 do p a pomoci ktere vypiseme prvocisla z mnoziny *prvocisla*.

```
⟨ Promenne programu 4 ⟩ +≡
i: integer
```

19. Nyni budeme testovat jestli cislo i je v mnozine *prvocisla*. Jestli ano, pak je vypiseme, jestli ne testujeme i o jednicku vyssi. Jako prvni prvocislo vypiseme cislo 1, ktere sem samozrejme take patri, ale kvuli vypoctu jsme ho do pomocne mnoziny *sito* nezahrnuli, jiste kazdy vi proc.

```
⟨ Vypis sita 19 ⟩ ≡
  writeln('1');
  for i ← 2 to p do
    if i ∈ prvocisla then writeln(i)
```

This code is used in section 13.

20. Index. Zde je rejstrik vsech pouzitych pojmu.

⟨ Eliminace nasobku prvocisla ze sita 17 ⟩ Used in section 13.
⟨ Hledani dalsiho prvocisla 16 ⟩ Used in section 13.
⟨ Inicializace pomocnych mnozin 15 ⟩ Used in section 13.
⟨ Promenne programu 4, 8, 10, 14, 18 ⟩ Used in section 2.
⟨ Testovani zkoumaneho cisla 11 ⟩ Used in section 7.
⟨ Ukazkovy program na urcovani prvocisel 2 ⟩ Used in section 1.
⟨ Vetveni programu na dva podprogramy 3 ⟩ Used in section 2.
⟨ Vstupni hodnota pro urceni vyberu podprogramu 5 ⟩ Used in section 3.
⟨ Vstupni hodnota zkoumaneho cisla 9 ⟩ Used in section 7.
⟨ Vyber podprogramu podle vstupni hodnoty 6 ⟩ Used in section 3.
⟨ Vypis prvocisel mensich nez 13 ⟩ Used in section 6.
⟨ Vypis sita 19 ⟩ Used in section 13.
⟨ Vypis vysledku testu 12 ⟩ Used in section 7.
⟨ Zjisti zda zadane cislo je prvocislo 7 ⟩ Used in section 6.