Metódy programovania: jazykové konštrukcie pre riadenie

1. Programovací jazyk C
   - charakteristika a vlastnosti
   - podporované metódy programovania
2. Príkazy jazyka C
   - deklaračné a definičné
   - riadiace príkazy
     - jednoduché
       - štrukturované (vetvenia, cykly, funkčné a procedurálne volania)
       - neštrukturované (skoky)
     - zložené
3. Podprogramy v jazyku C - funkcie
   - definícia, deklarácia, použitie

Programovací jazyk C charakteristika a vlastnosti

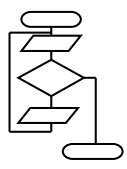
univerzálny
procedurálny
kompilačný

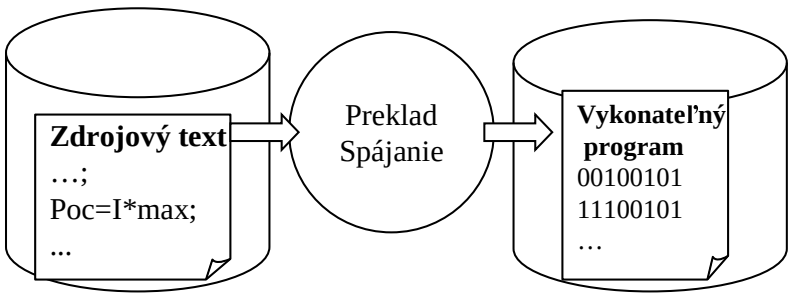
jazyk vyššej úrovne
jazyk strojovej úrovne
vysoká prenositeľnosť
kompilátora
aplikácií

Vedecko-technické výpočty
Databázové aplikácie
Systémové programy
Ovládače zariadení
Programové vybavenie v špecializovaných zariadenieach (embeded systems)
...

Procedúry a funkcie

...nutnosť definovania krokov – postupu – algoritmu...





Programovací jazyk C charakteristika a vlastnosti

univerzálny
procedurálny
kompilačný
jazyk vyššej úrovne
jazyk strojovej úrovne
vysoká prenositeľnosť
kompilátora
aplikácií

Premenné
Typy
Preddefinované
Definovateľné
Jednoduché
Štruktúrované
Výrazy
Aritmetické
Logické
Relačné
Príkazy
Podprogramy

```
#define PORT1 0x2A

struct port_str {
  unsignet int hlava:1;
  unsignet int sektor:4;
  unsignet int pozicia:3;
} pom;

void setPort (void) {
  pom.hlava = 1;
  pom.sektor = 12;
  pom.pozicia = 4;
  outportb (PORT1, pom);
  pom.pozicia = 3;
  outportb (PORT1, pom);
}
```

Jazyk bez príkazov pre:
V/V, prácu so súbormi
dynamické prideľovanie OP
vytváranie a synchronizáciu procesov

Strojovo závislé prvky a väzba na operačný systém sú realizované
..... **knižnicami**

C-program

Triviálny C-program

```
main() { }
```

pr0.c

Preklad:
cc pr0.c

Odštartovanie:
a.out

Príklad 0

```
main() {
    printf (“hello, world\n“);
}
```

funkčné

Príklad 1

```
#include <stdio.h>
main() {
    printf (“hello, world\n“);
}
```

funkčné a **koncepčne lepšie...**

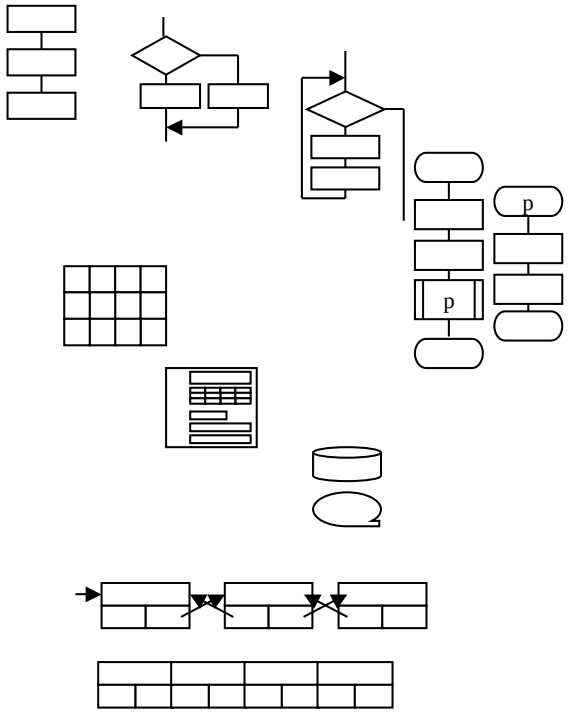
vloženie hlavičkového súboru

hlavná funkcia (vykonáva sa vždy ako prvá)

volanie knižničnej funkcie

Podporované metódy programovania

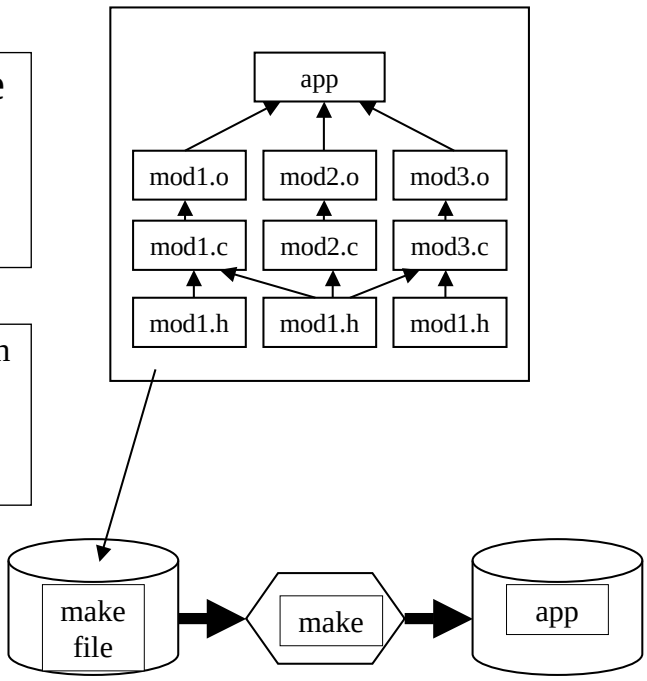
- **štruktúrované programovanie**
  - **riadiace štruktúry**
    - sekvencie
    - vetvenia
    - cykly
    - definície a volania podprogramov
  - **údajové štruktúry**
    - štandardné
      - polia
      - záznamy
      - súbory
    - neštandardné
      - zoznamy
        - spojkové
        - sekvenčné



Podporované metódy programovania

- **modulárne programovanie**
  - samostatne kompilovateľné moduly
  - export a import dát a funkcií

- vývoj a ladenie aplikácie po moduloch (inkrementálne)
- zakrývanie implementácie
- sprístupňovanie rozhraní



PRÍKAZY JAZYKA C

- deklaračné
  - deklarácia premenných — extern int a;
  - deklarácia typov — typedef char BYTE;
  - deklarácia funkcií — float skalar_sucin(float [], int);
- definičné
  - definícia premenných — int a, v[10];
  - definícia funkcií — void prompt(void) { printf("\n>"); }
  - (definícia konštánt a makier - príkazy predprocesora)
- riadiace - definícia toku riadenia
  - stavebné kamene pre definovanie algoritmu

Riadiace príkazy

- jednoduché (....syntax, ukončovací znak)
  - štruktúrované
    - vetvenia — **if, if-else, if-else if-else, switch**
    - cykly — **while, do-while, for**
    - funkčné a procedurálne volania
  - neštruktúrované
    - výrazové príkazy
    - skoky — **break, continue, goto**
- zložené - blok {p1,…,pn}
  - podmienky radenia a vnárania blokov

Príkaz vetvenia IF

- if (vyraz)
  prikaz
- if (vyraz)
  prikaz1 **dvojcestné**
  else **vetvenie**
  prikaz2
- if (vyraz1)
  prikaz1
  else if (vyraz2)
  prikaz2 **viaccestné**
  …. **vetvenie**
  else
  prikaz

VNÁRANIE PRÍKAZOV IF

ELSE sa viaže k najbližšiemu predošlému IF !!!

```
//načítanie hodnoty do
//premennej x - <0,99>
if (x >= 0)
        if ( x > 99 ) {
                x = 99;
        }
else
        x=0;
```

Čo sa vykoná.

Vizuálne odsadenie:
„čo chcel programátor“

Príkaz vetvenia IF

- if (vyraz)
  prikaz
- if (vyraz)
  prikaz1 **dvojcestné**
  else **vetvenie**
  prikaz2
- if (vyraz1)
  prikaz1
  else if (vyraz2)
  prikaz2 **viaccestné**
  …. **vetvenie**
  else
  prikaz

VNÁRANIE PRÍKAZOV IF

ELSE sa viaže k najbližšiemu predošlému IF !!!

```
//načítanie hodnoty do
//premennej x - <0,99>
if (x >= 0) {
        if ( x > 99 ) {
                x = 99;
        }
}
else
        x = 0;
```

Príkaz SWITCH - prepínač

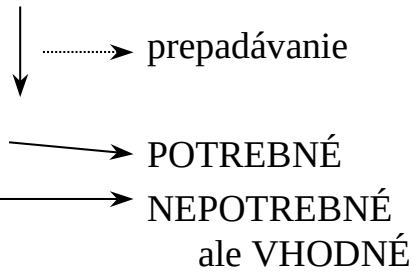
```
switch (celocíselny_vyraz) {
    case CHOD1: prikaz1
    case CHOD2: prikaz2
    …
    case CHODn: prikazN
    default: prikaz
}
```

- **Rozdiel v použití IF a SWITCH**
  - univerzálnosť IF
  - rýchlosť SWITCH
- **Význam vetvy DEFAULT**
- **Prepadávanie cez alternatívy**
- **Použitie príkazu BREAK**

Príklad použitia príkazu SWITCH

```
/* počítanie bielych a iných znakov */
#include <stdio.h>
main( ) {
  int znak, biele, ine;
  biele = ine = 0;

  while (( znak=getchar( )) != EOF) {
        switch ( znak ) {
              case ‘ ‘ :
              case ‘\t’ :
              case ‘\n’ : biele++; break;
              default : ine++; break;
        }
  }
  printf(“\nbiele=%d, ine=%d\n”, biele, ine);
}
```



Príkaz cyklu WHILE

while (vyraz)
prikaz

- výraz sa vyhodnotí vždy
- príkaz sa nemusí vykonať ani raz
- vykonávanie cyklu pokračuje, pokiaľ hodnota výrazu je NENULOVÁ

Príklad na použitie cyklu WHILE

```
/* preskoč biele znaky */
while ( (  c=getchar( ) ) == ‘ ‘ ||
           c == ‘\n’              ||
           c == ‘\t’
        ) ;
```

PRÁZDNE TELO CYKLU

```
/* čítanie zo stdin */
while ( EOF != (c=getchar( ) ) )
    {
    …….
    }
```

SPRÁVNY ALGORITMUS AJ VTEDY, KEĎ VSTUPI EOF

Cyklus do while

do príkaz
while (výraz);

- Cyklus vykonáva telo - *príkaz*, kým je *výraz* nenulový (to zodpovedá logickej hodnote true), najmenej však jedenkrát!

Príklad na použitie príkazov IF a DO WHILE

```
/* binar. vyhľadávanie */

int binschr(x,v,n)
int x,    /*hľadané číslo */
    v[ ], /*prehl.vektor   */
    n;    /*počet prvkov */
{
   int i, j, k;
   i = 0;
   j = n-1;
   do {
         k = (i + j)/2;
         if (x > a[k]) {
                  i = k + 1;
         } else {
                  j = k – 1;
         }
   } while (a[k]!=x && i<=j);
   if (a[k]==x)
         return k;
   else
         return –1;
}
```

Príkaz cyklu FOR

for ([V1]; [V2]; [V3] **)**
príkaz

1. ak existuje V1, vyhodnotí sa
2. ak existuje V2, vyhodnotí sa
3. ak V2 je nulové, koniec cyklu
4. ak V2 je nenulové.
   - vykoná sa príkaz
   - ak existuje V3, vyhodnotí sa
   - pokračuje sa krokom 2

význam

V1 - inicializácia

V2 - test konca

V3 - príprava ďalšieho kroku cyklu

- univerzálnosť FOR
- možné bočné efekty výrazov V1,V2,V3

Príklad použitia cyklu FOR

```
/* konverzia reťazca na cele číslo */
char s[ ] = “12345“; //pole znakov
int i, n;
n=0;
for (  i=0;
       s[i]>=‘0’ && s[i]<=‘9’ ;
       i++)
    n = 10*n + s[i] - ‘0’;
```

INICIALIZÁCIA

TEST

PRÍPRAVA ĎALŠIEHO KROKU

TELO CYKLU

Príkazy skoku

- break
- continue
- return
- goto

Break, continue

- Príkaz *break* ukončuje najvnútornejšiu neukončenú slučku cyklu a opúšťa telo cyklu
  - spôsobuje skok z príkazu cyklu na ďalší príkaz
- Príkaz *continue* preruší vykonávanie najbližšej nadradenej štruktúrovanej konštrukcie cyklu (najvnútornejšiu neukončenú slučku cyklu)
  - spôsobí prechod na ďalší krok cyklu bez toho, aby sa predchádzajúci krok cyklu ukončil
  - nespôsobuje skok z tela cyklu von

Skokový príkaz CONTINUE

```
/* vynulovanie záporných
   čísel v poli */

   for (i=0; i<N; i++) {
      if(a[i] > 0) {
          continue;
      }
      a[i] = 0;
   }
```

- začatie ďalšej iterácie najvnútornejšieho cyklu(…skok v rámci štruktúry…)
- ak sa má vykonať inicializačný výraz cyklu a opakovať cyklus od 1.iterácie, nedá sa použiť continue

Return

- Príkaz *return* ukončí vykonávanie funkcie a môže vrátiť hodnotu
- Môže sa vyskytovať v jednom z tvarov
  - `return výraz;`
  - `return (výraz);`
  - `return;`

Goto

- Príkaz *goto* umožňuje vykonať skok v rámci jednej funkcie
- Je možné vykonať skok z bloku do bloku a von z cyklu.
- Všeobecne sa doporučuje používať príkaz goto čo najmenej a skutočne len vo výnimočných prípadoch.
- POUŽITIE: ukončenie sústavy vnorených cyklov.

Podprogramy v jazyku C - funkcie

- základ C-programu
- **Definícia funkcie**
  - popisuje funkciu z hľadiska jej vstupov, výstupov (t.j. rozhrania – *interface*), algoritmov a použitých údajových štruktúr
  - v danom programe práve jedna

- **Syntax definície funkcie:**

```
[<typ_návratovej hodnoty>] <meno_funkcie>
  (<zoznam_formálnych_parametrov>)
<deklarácia_formálnych_parametrov>
{
  <telo_funkcie>
}
```

Typ funkcie a argumentov

- implicitný typ návratovej hodnoty je ***int***

```
sucet_cisel (a,b)
int a,b;       //deklarácia formálnych parametrov
{return a+b;}  //funkcia implicitne vracia hodnotu
               //typu int
```

- funkcia bez návratovej hodnoty je typu ***void***

```
void vypis ()
{ printf ("Bola zavolana funkcia vypis!\n"); }
```

- ak funkcia nemá žiadne formálne parametre norma ANSI C odporúča na ich mieste použiť kľúčové slovo ***void***

```
void vypis (void)
{ printf ("Bola zavolana funkcia vypis!\n"); }
```


Deklarácia a prototyp funkcie

- **Deklarácia funkcie**

  - popisuje funkciu len z hľadiska jej rozhrania (meno funkcie a typ návratovej hodnoty)

  Syntax deklarácie funkcie:

```
<typ_návratovej hodnoty> <meno_funkcie> ();
```

```
int sucet_cisel();   //deklarácia funkcie
```

- **Prototyp funkcie**

  -norma ANSI C doporučuje používať deklaráciu funkcie v tvare tzv. *prototypu* (meno, typ návratovej hodnoty a typy parametrov)

- **Syntax prototypu funkcie:**

```
<typ_návratovej hodnoty> <meno_funkcie>
                 (<typ_param1>[,<typ_param2>]. . .);
```

```
int max3 (int, int, int);  //prototyp funkcie
```

Volanie funkcie

- **Syntax volania funkcie:**

  `<meno_funkcie> (<zoznam_skutočných_parametrov>)`

- **volanie funkcie** je rozdielne pre funkcie s a bez návratovej hodnoty
  - **s návratovou hodnotou** – volanie funkcie môže byť použité vo výraze ako operand

```
main()
{int x, y, z;   // deklarácia lokálnych
premenných
 . . .
printf („Sucet cisel %d a %d = %d\n", x, y,
sucet_cisel(x,y));
. . .
z = sucet_cisel(x,y);
. . .}
```

  - **bez návratovej hodnoty (void funkcia)** – volanie funkcie nesmie byť použité ako operand

```
main()
{. . .
vypis(); //funkcia vypis je
. . .}   // void funkcia
```

```
main()
{. . .
a = vypis();//CHYBA!!!
. . .}
```