Algoritmizácia a programovanie
4. prednáška

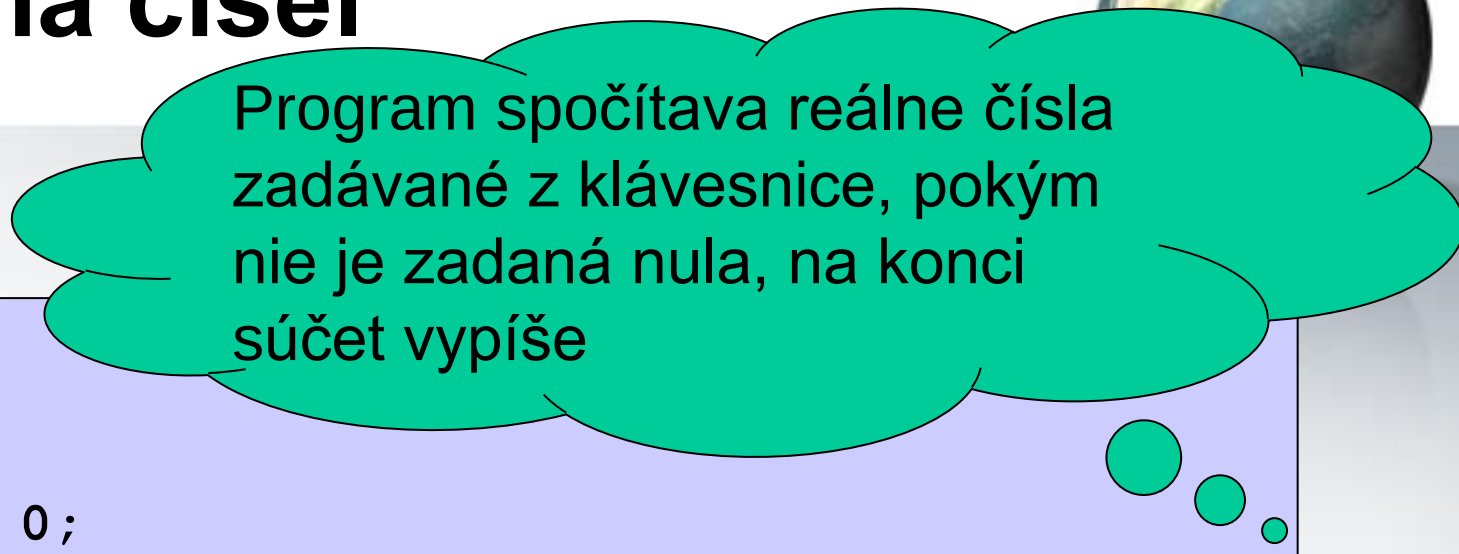
Ján Grman

Obsah

1. opakovanie
2. práca so súborom
3. príklady práce so súborom

Riadiace štruktúry - opakovanie v príkladoch

Príklad: suma čísel




```
#include <stdio.h>

int main() {
   float x, suma = 0;

   printf("Zadajte cisla (ukoncite 0)\n");

   do {
      scanf("%f", &x);
      suma += x;
   } while(x != 0.0);

   printf("Sucet je: %.3f", suma);
   return 0;
}
```

Príklad: minimum, maximum

```
#include <stdio.h>

int main() {
   int i, n;
   float x, min, max;

   scanf("%d", &n);
   if (n > 0) {
      scanf("%f", &x);
      min = max = x;

      for( i=2; i<=n; i++ ) {
         scanf("%f", &x);
         if(x > max)
            max = x;
         else if(x < min)
            min = x;
      }
      printf("Minimum: %.2f\n", min);
      printf("Maximum: %.2f\n", max);
   }
   return 0;
}
```

Program načíta **n** čísel a vypíše minimum a maximum (príklad z cvičení)

Príklady: hviezdičkovanie do trojuholníka

```
#include <stdio.h>

int main() {
   int i, j, n, r;

   scanf("%d", &n);
   for(i=1; i<=n; i++) {
      for(j=1; j<=n; j++)
         if(i < j)
            putchar('*');
         else
            printf("%d", i%10);
      putchar('\n');
   }
   return 0;
```

Pre daný počet riadkov vykreslite obrázok

```
1*******
22******
333*****
4444****
55555***
666666**
7777777*
88888888
99999999
```

```c
#include <stdio.h>

int main() {
    int i, j, n, r;

    scanf("%d", &n);
    if(n < 1 || n > 15) return 0;
    for(i=1; i<=n; i++) {
        r = i;
        for(j=1; j<=2*n-1; j++) {
            if(j <= n-i || j >= n+i)
                putchar('*');
            else
                printf("%d", i%10);
        }
        putchar('\n');
    }
    return 0;
}
```

Pre daný počet riadkov vykreslite obrázok

```
*******1*******
******222******
*****33333*****
****4444444****
***555555555***
**6666666666**
*7777777777*
88888888888888
```

Vstup a výstup z textového súboru

Súbory

- súbor
  - postupnosť Bytov uložených na médiu (disku) v niekoľkých blokoch (nie nutne za sebou)
  - prístup k blokom - operačný systém
- vstup zo súboru
  - naraz sa prečíta celý blok z disku do pamäte (buffer) - položky sa potom čítajú z pamäte (rýchlejšie)
- výstup
  - dáta sa zapisujú do bufferu a keď je plný, zapíše sa na disk
  - napr. v UNIXE sa dá používať aj nebufrované vstupné a výstupné operácie (**`io.h`**)
- koniec súboru
  - často špeciálny znak (napr. "**`Ctrl Z`**")

Začiatky práce so súborom

- základný dátový typ: **`FILE *`**
  - ukazovateľ (pointer - **`*`**) na objekt typu FILE
    - ukazovateľ obsahuje adresu objektu typu FILE
    - ako adresár - zapísaná adresa, kde začína súbor na disku
  - dodržať veľké písmená (**`FILE *`**, nie **`file *`**)
- definícia premennej **`f`** pre prácu so súborom:

```
FILE *f;
```

  - aj pre čítanie, aj pre zápis rovnaké
  - pre viac premenných:

```
FILE *fr, *fw;
```

pre čitateľnosť je vhodné používať **`fr`** pre čítanie, **`fw`** pre zápis

Otvorenie súboru na čítanie a zápis

- čítanie

`r` ako read

```
fr = fopen("POKUS.TXT", "r");
```

- zápis

`w` ako write

```
fw = fopen("POKUS.TXT", "w");
```

- aj ďalšie režimy otvorenia súboru (nielen "**r**" a "**w**")
- režimy "**r**" a "**w**" - otvorenie textového súboru
- režimy "**rb**" a "**wb**" - otvorenie binárneho súboru

  → neskôr

Základné práce s otvoreným súborom

- porovnanie s čítaním z klávesnice a zápisom na obrazovku

```
int fgetc(FILE *f);
int getc(FILE *f);
int getchar();
```

čítanie znaku zo súboru

```
int fputc(int c, FILE *f);
int putc(int c, FILE *f);
int putchar(int c);
```

zápis znaku do súboru

```
fscanf(FILE *f, "format", args);
scanf("format", argumenty);
```

formátované čítanie zo súboru

```
fprintf(FILE *f, "format", args);
printf("format", argumenty);
```

formátovaný zápis do súboru

Ukončenie práce so súborom

- keď už nebudeme zo súboru čítať ani doňho zapisovať - uzatvoriť súbor, premenná **`f`** je typu **`FILE *`**:

```
fclose(f);
```

- nespoliehať sa, že po skončení programu sa v mnohých systémoch automaticky uzavrie súbor
  - počet súčasne otvorených súborov je obmedzený
  - zápis bufferu do súboru (preto uzatvárať ihneď) - pri spadnutí programu by zostali dáta v bufferi a stratili by sa

Príklady

program vytvorí súbor **pokus.txt**, zapíše doňho čísla od 1 po 10, každé v zvlášť riadku

```
#include <stdio.h>

int main() {
    FILE *fw;
    int i;

    fw = fopen("pokus.txt", "w");
    for (i = 1; i <= 10; i++)
       fprintf(fw, "%d\n", i);
    fclose(fw);
    return 0;
}
```

Príklady

program načíta tri double čísla zo súboru **data.txt** a ich súčet vypíše na obrazovku

```
#include <stdio.h>

int main() {
    FILE *fr;
    double x, y, z;

    fr = fopen("data.txt", "r");
    fscanf(fr, "%lf %lf %lf\n", &x, &y, &z);
    printf("%lf\n", x + y + z);
    fclose(fr);
    return 0;
}
```

funkcia **fscanf()** vracia počet úspešne prečítaných položiek (v prípade konca súboru vracia hodnotu **EOF**)

Príklady

program načíta tri double čísla zo súboru **data.txt** a vypíše ich na obrazovku - testuje, či sú v súbore 3 čísla

```
#include <stdio.h>

int main() {
   FILE *fr;
   double x, y, z;

   fw = fopen("data.txt", "r");
   if(fscanf(fr, "%lf %lf %lf\n", &x, &y, &z) == 3)
      printf("%lf \n", x + y + z);
   else
      printf("Subor neobsahuje 3 realne cisla\n.");
   fclose(fr);
   return 0;
}
```

Príklady

program prečíta 2 znaky zo súboru **znaky.txt** a zapíše ich do súboru **kopia.txt**

```
#include <stdio.h>

int main() {
    FILE *fr, *fw;
    int c;

    fr = fopen("znaky.txt", "r");
    fw = fopen("kopia.txt", "w");

    c = getc(fr);
    putc(c, fw);
    putc(getc(fr), fw);

    fclose(fr);
    fclose(fw);
    return 0;
}
```

Testovanie konca riadku

- postarať sa o testovanie konca riadku (EOLN - označenie, nie symbolická konštanta)
  - testovanie štandardného znaku pre koniec riadku v C: **`\n`**
  - **`\n`** - aj pre čítanie, aj pre zápis
  - **`\n`** - význam určuje prekladač podľa systému (**`<CR>`**, **`<LF>`**, alebo **`<CR><LF>`**)

Príklad

program prečíta zo súboru **`list.txt`** riadok a aj s koncom riadku ho vypíše na obrazovku

```
#include <stdio.h>

int main() {
    FILE *fr;
    int c;

    fr = fopen("list.txt", "r");
    while ((c = getc(fr)) != '\n')
       putchar(c);
    putchar(c);     /* vypis \n */
    fclose(fr);
    return 0;
}
```

Testovanie konca súboru

- dva spôsoby:
  - pomocou konštanty **`EOF`** alebo
  - pomocou makra **`feof()`** - pomalejšie (volanie makra)

Testovanie konca súboru: `EOF`

- symbolická konštanta **`EOF`**
  - väčšinou definovaná v **`stdio.h`**
  - väčšinou má hodnotu -1

```
if ((c = getc(fr)) != EOF)
    ...
```

  - premenná **`c`** nesmie byť definovaná ako **`char`**, pretože **`EOF`** je reprezentovaná ako **`int`** s hodnotou -1 (-1 by bola na char konvertovaná ako iný znak)

Príklad

program skopíruje obsah súboru **list.txt** do súboru **kopia.txt**

```
#include <stdio.h>

int main() {
    FILE *fr, *fw;
    int c;

    fr = fopen("list.txt", "r");
    fw = fopen("kopia.txt", "w");

    while ((c = getc(fr)) != EOF)
        putc(c, fw);

    fclose(fr);
    fclose(fw);
    return 0;
}
```

Testovanie konca súboru: `feof()`

- makro **`feof()`** vracia
  - TRUE (nenulovú hodnotu), keď posledné čítanie bolo za koncom súboru
  - FALSE (nulu), keď sa pri čítaní nedošlo na koniec

- vhodné, keď čítame z binárneho súboru
  - môže obsahovať znak s hodnotou 0xFF - pomocou implicitnej konverzie je prevedený na hodnotu EOF - skončilo by sa s čítaním skôr

    $\rightarrow$ neskôr

Príklad

program skopíruje obsah súboru **list.txt** do súboru **kopia.txt**

```
#include <stdio.h>

int main() {
    FILE *fr, *fw;
    int c;

    fr = fopen("list.txt", "r");
    fw = fopen("kopia.txt", "w");

    while (c = getc(fr), feof(fr) == 0)
       putc(c, fw);

    fclose(fr);
    fclose(fw);
    return 0;
}
```

Testovanie správnosti otvorenia a zatvorenia súboru

- akcie: otvorenie a zatvorenie súboru
  - nemusia byť úspešné
  - preto testovať úspešnosť a podľa toho pokračovať

- **`fopen()`**

  - ak sa podarí otvoriť súbor - vracia ukazovateľ na súbor,
  - inak - vracia konštantu **`NULL`** (definovaná v **`stdio.h`**, má hodnotu 0)

testovanie:

```
if((fr = fopen("test.txt", "r")) == NULL)
   printf("Subor sa nepodarilo otvorit.\n");
```

Testovanie správnosti otvorenia a zatvorenia súboru

- `fclose()`
  - Ak sa podarí zatvoriť súbor – vracia hodnotu 0
  - Ak sa nepodarí zatvoriť súbor - vracia konštantu **`EOF`**

testovanie:

```
if(fclose(fr) == EOF)
   printf("Subor sa nepodarilo zatvorit.\n");
```

program skopíruje obsah súboru **list.txt** do súboru **kopia.txt** - s ošetreniami

```
#include <stdio.h>

int main() {
    FILE *fr, *fw;
    int c;

    if ((fr = fopen("list.txt", "r")) == NULL) {
       printf("Subor sa nepodarilo otvorit.\n");
       return 0;
    }
    if ((fw = fopen("kopia.txt", "w") ) == NULL) {
       printf("Subor sa nepodarilo otvorit.\n");
       return 0;
    }

    while ((c = getc(fr)) != EOF)
       putc(c, fw);

    if (fclose(fr) == EOF)
       printf("Subor sa nepodarilo zatvorit.\n");
    if (fclose(fw) == EOF)
       printf("Subor sa nepodarilo zatvorit.\n");
    return 0;
}
```

tu by sme mali správne zatvoriť súbor **list.txt**

Príklad

program vypíše súbor **`list.txt`** na obrazovku tak, že skonvertuje všetky malé písmená na veľké a nakoniec vypíše dĺžku najdlhšieho riadku

```
#include <stdio.h>

int main() {
   FILE *fr;
   int c, dlzka, max_dlzka;

   if ((fr = fopen("list.txt", "r")) == NULL) {
      printf("Subor sa nepodarilo otvorit.\n");
      return 0;
   }

   dlzka = max_dlzka = 0;
   while((c= getc(fr)) != EOF) {
      dlzka++;
      if (c>='a' && c<='z')
         c += 'A' - 'a';
      putchar(c);
      if (c == '\n') {
         if (max_dlzka < dlzka)
            max_dlzka = dlzka;
         dlzka = 0;
      }
   }
```

kontrola, či sa podarilo otvoriť súbor

Číta znaky pokým nie je koniec súboru

Napočítava dĺžku riadku

Konverzia malých písmen na veľké

Výpis znaku

Ak je riadok doteraz najdlhší, zapamätanie si jeho dĺžky

Vynulovať dĺžku pre počítanie dĺžky ďalšieho riadku

pokračovanie:

```
    if (max_dlzka < dlzka)
       max_dlzka = dlzka;

    printf("Max. dlzka: %d\n", max_dlzka);
    if (fclose(fr) == EOF)
       printf("Subor sa nepodarilo zatvorit\n");
    return 0;
}
```

Kontrola aj dĺžky posledného riadku, ktorý nemusí končiť \n

Výpis a zatvorenie súboru

Štandardný vstup a výstup

- **`stdin`** a **`stdout`** môžu byť použité v programe ako argumenty operácií so súbormi:

`getc(stdin)`	je ekvivalentné	`getchar()`
`putc(c, stdout)`	je ekvivalentné	`putchar(c)`

- v **`stdio.h`** je definovaný ešte tretí prúd **`stderr`**, ktorý sa používa pri vypisovaní chybových správ

Príklad

program ukáže ako využívať stdout a vyhnúť sa problémom s bufferom

program vypíše otázku, či má byť výstup vypísaný na obrazovku, alebo do súboru **`vystup.txt`**. Ak súbor existuje, vypíše otázku, či má byť súbor prepísaný

```
#include <stdio.h>

int main() {
  FILE *fw;
  int c;

  printf("Stlacte O pre vypis na Obrazovku\n");
  printf("alebo iny znak pre zapis do suboru VYSTUP.TXT: ");

  c = getchar();
  while (getchar() != '\n')
     ;
```

Vyprázdnenie buffera – preskočí zvyšok riadku

```
if (c == 'o'  ||  c == 'O')
  fw = stdout;
else {
  if ((fw = fopen("vystup.txt", "r")) != NULL) {
    printf("Subor vystup.txt existuje, prepisat? [A/N]: ");
    c = getchar();
    while (getchar() != '\n')
      ;
    if (fclose(fw) == EOF) {
      printf("Chyba pri zatvarani suboru\n");
      return 0;
    }
    if (!(c == 'a'  ||  c == 'A'))
      return 0;
  }
  if ((fw = fopen("vystup.txt", "w")) == NULL) {
    printf("Subor vystup.txt sa nepodarilo otvorit\n");
    return 0;
  }
}
```

Výpis na štandardný výstup (obrazovku)

Ak súbor existuje (podarilo sa ho otvoriť na čítanie), zistenie, či sa má prepísať

Ak nechce prepísať súbor, program skončí

Otvorenie súboru na zápis s kontrolou

pokračovanie:

```
  printf("Piste text a ukoncite ho znakom *\n");
  while ((c = getchar()) != '*')
    putc(c, fw);

  if (fw != stdout) {
    if (fclose(fw) == EOF) {
      printf("Subor vystup.txt se nepodarilo zatvorit\n");
      return 0;
    }
  }
  return 0;
}
```

Čítanie znakov a ich zápis do fw (súbor alebo štandardný výstup)

Ak sa zapisovalo do súboru, jeho zatvorenie s kontrolou

Vrátenie prečítaného znaku späť do bufferu

- často zistíme, že máme prestať čítať znak až potom, čo prečítame o znak naviac → vrátiť do bufferu

`ungetc(c, fr)`

vráti znak do vstupného bufferu
- ak je vrátenie úspešné, **`ungetc()`** vracia vrátený znak
- ak je vrátenie neúspešné, vráti **`EOF`**

- späť do bufferu môžeme zapísať aj iný ako práve prečítaný znak

Príklad

časť programu konvertuje
znakový reťazec na
zodpovedajúcu číselnú hodnotu

```
int c, hodnota = 0;

while ((c = getchar()) >= '0' && c <= '9') {
  hodnota = hodnota * 10 + (c - '0');
}
ungetc(c, stdin);
```

Príklad

časť programu prečíta číslo pomocou **`fscanf()`** - predtým však musí prečítať neznámy počet znakov **`'$'`** (predpokladáme otvorený súbor)

```
int c, hodnota = 0;

while ((c = getc(fr)) == '$')
    ;
ungetc(c, stdin);
fscanf(fr, "%d", &hodnota);
```

Rôzne možnosti otvárania súborov v textovom režime

- súbory sa otvárajú stále rovnakou funkciou **`fopen()`**
  - či ide o textový alebo binárny súbor
  - či ide o zápis alebo čítanie
- prototyp funkcie:

- **`const`**: len vstupný argument, nebude sa meniť vo funkcii
- **`char *`** - reťazec znakov

```
FILE *fopen(const char *meno, const char *rezim)
```

vráti ukazovateľ (adresu) na otvorený súbor alebo **`NULL`**

meno súboru

aký typ súboru a na akú činnosť sa bude otvárať

Významy parametru `rezim` (textové súbory)

r	textový súbor pre čítanie
w	textový súbor pre zápis alebo pre prepisovanie
a	textový súbor pre pripojenie na koniec
r+	textový súbor pre čítanie a zápis
w+	textový súbor pre čítanie, zápis alebo prepisovanie
a+	textový súbor pre čítanie a zápis na koniec

Významy parametru `rezim` (textové súbory)

požiadavky / režim otvorenia	"r"	"w"	"a"	"r+"	"w+"	"a+"
súbor musí existovať	+			+		
existujúci súbor bude wymazaný		+			+	
existujúci súbor bude rozšírený			+			+
neexistujúci súbor bude vytvorený		+	+		+	+
čítať - z ľubovoľného miesta v súb.	+			+	+	+
zapisovať - na ľubovoľné miesto v súb.		+		+	+	
zapisovať - iba na koniec súb.			+			+

Významy parametru `rezim` (textové súbory)

- niektoré implementácie umožňujú explicitne určiť, že ide o textový režim: **`"rt"`** **`"wt"`** **`"at"`**
- ak otvoríme existujúci súbor v režime **`"w"`**, tak sa tento súbor najprv vymaže a potom sa vytvorí nový
- ak otvoríme existujúci súbor v režime **`"a"`** tak sa tento súbor otvorí a ukazovateľ sa presunie na koniec súboru (rozširovanie existujúceho súboru)
- ak použijeme režim rozšírený o znak **+**, je možné do súboru aj zapisovať

Nastavenie sa na zvolenú pozíciu v súbore

```
int fseek(FILE *stream, long offset, int whence)
```

nastavenie ukazovateľa na pozíciu čítania alebo zápisu v otvorenom súbore.
Argumenty:

- **`stream`** – ukazovateľ na súbor
- **`offset`** – relatívna pozícia oproti whence, na ktorú sa má ukazovateľ posunúť (v Bytoch)
- **`whence`** – k čomu je offset relatívny
  - **SEEK_SET:** **`offset`** – relatívne k začiatku súboru
  - **SEEK_CUR:** **`offset`** – relatívne k aktuálnej pozícii
  - **SEEK_END:** **`offset`**– relatívne ku koncu súboru (treba používať negatívne hodnoty)

Návratová hodnota: 0 pri úspechu, -1 pri neúspechu

```
void rewind(FILE *stream)
```

nastavenie ukazovateľa na začiatok súboru

Nastavenie sa na zvolenú pozíciu v súbore

```
fseek(fp, 100, SEEK_SET);
fseek(fp, -30, SEEK_CUR);
fseek(fp, -10, SEEK_END);

fseek(fp, 0, SEEK_SET);
rewind(fp);
```

Nastavenie na 100 byte súboru

Nastavenie na 30 bytov dozadu od aktuálnej pozície

Nastavenie na 10 bytov pred koniec súboru

Nastavenie na začiatok súboru

Nastavenie na začiatok súboru

Nastavenie sa na zvolenú pozíciu v súbore

```
long ftell(FILE *stream);
```

zistenie pozície ukazovateľa čítania, zápisu v otvorenom súbore relatívne k začitaku súboru, t.j. kde nastane nasledujúca operácia (opak **`fseek()`**)
Použitie: zapamätať si pozíciu, na ktorú sa neskôr plánujete vrátiť (zapamätať si návratovú hodnotu a potom ju použiť vo fseek() relatívne k začiatku súboru)
Návratová hodnota: aktuálna pozícia alebo -1 v prípade neúspechu