

Vnorené cykly úvod k súborom

Základy procedurálneho programovania 1, 2020

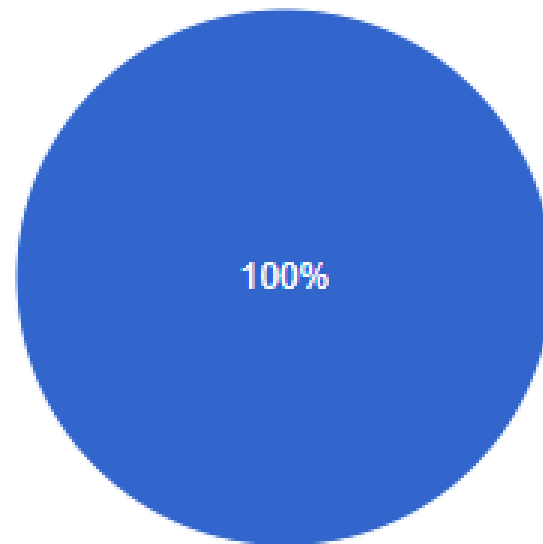
Ing. Marek Galinski, PhD.





Dotazník

Ďakujem

Ako hodnotíš prednášky? Kvalita, zrozumiteľnosť, pútavosť, ...

40 odpovedí



-  Výborne, všetko super
-  Ujde to, ale mohlo by to byť lepšie
-  Podpriemer, nuda, nezrozumiteľné
-  Úplne celé zle

Ďakujem

Páči sa mi že na všetko kladiete dôraz a čakáte na naše reakcie. Veľmi sa mi páči aj to že každú zložitejšiu vec ideme hneď aj prakticky vyskúšať. A aj to že nám ukazujete chyby. Myslím tým že nenapišete program hneď správne ale ukažete nám aj možné chyby, ako sa program správa keď je zle napísany.

Vnorené cykly úvod k súborom

Základy procedurálneho programovania 1, 2020

Ing. Marek Galinski, PhD.

Opakovanie

Vyhodnocovanie výrazov

- Čo platí a čo neplatí?

```
int i = 1, j = 1, k;
```

```
k = i && (j == 3);
```

NEPLATÍ, k bude 0

```
k = !i && j;
```

NEPLATÍ, k bude 0

SKRÁTENÉ VYHODNOCOVANIE

```
k = (i = 2) || j;
```

PLATÍ, k bude 1

SKRÁTENÉ VYHODNOCOVANIE

Príkaz else-if

- Jednotlivé výrazy sa vyhodnocujú postupne
- Prvý, ktorý sa vyhodnotí ako splnená podmienka, ten sa vykoná a končí
- Ak nie je splnená žiadna podmienka, vykoná sa príkaz za posledným else

```
if (podmienka_1)
    prikaz_1
else if (podmienka_2)
    prikaz_2
else if ...
    ...
else
    prikaz_n
```


Ternárny operátor

- Podmienený výraz – ternárny operator

```
podmienka ? vyraz_1 : vyraz_2
```

má význam:

Ak **podmienka**, tak **vyraz_1**, inak **vyraz_2**

```
int i, j = 2, k;  
  
i = (j == 2) ? 1 : 3;  
  
k = (i > j) ? i : j;
```

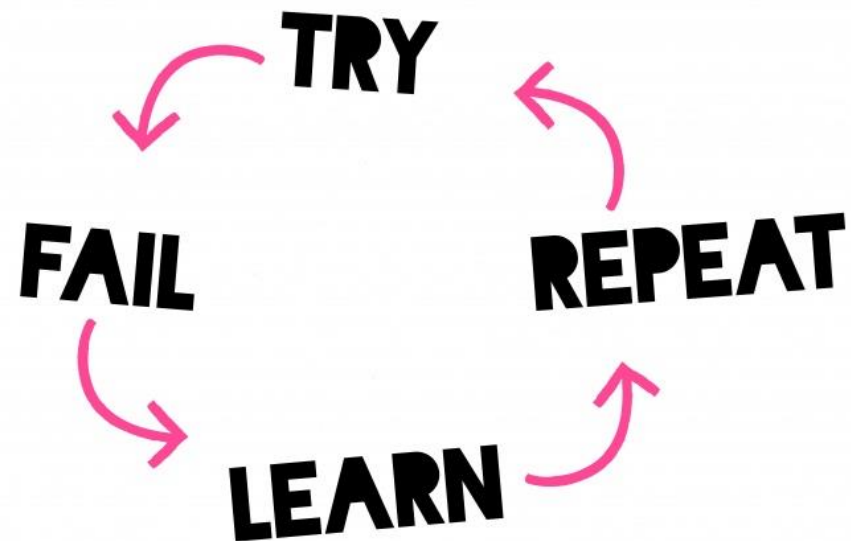
Operátor čiarky

- Operátor čiarky vyhodnocuje ľavý operand pred pravým operandom
- Syntax: `vyraz_1, vyraz_2`

```
int i = 2, j = 4;  
j = ( i++ , i - j );
```

Cykly

- Keď potrebujeme, aby program tú istú operáciu vykonal opakovane
- Cykly umožňujú opakovať príkaz, alebo celý blok príkazov
- Tri typy cyklov
 - for
 - while
 - do-while



Príkaz break a continue

- Príkazy, ktoré je možné použiť vo všetkých typoch cyklov, menia štandardné správanie cyklu

break

Ukončuje cyklus (ukončuje najvnútornejšiu slučku)

continue

Skáče na koniec slučky, v ktorej sa nachádza a vynucuje ďalšiu iteráciu

Vnorené cykly

Práca s textovým súborom

Súbor

- **Postupnosť bytov uložených na pamäťovom médiu**
- **Vstup zo súboru**
 - Prečíta sa naraz celý blok z disku do buffera (do pamäte) a následne sa už číta z pamäte
- **Výstup**
 - Dáta sa zapíšu do buffer (pamäť) a keď je plný, vykoná sa zápis na disk
- **Koniec súboru**
 - Spravidla špeciálny znak (EOF) - ASCII

Práca so súborom v C

- Základný dátový typ: `FILE *`
- Pointer (ukazovateľ) na object typu `FILE`
 - Zapísaná adresa, kde začína súbor na disku
 - Case sensitive
- Definícia premennej `f` pre prácu so súborom (môže ich byť viac)

```
FILE *f;
```

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


Práca so súborom v C

- Otvorenie súboru v C

- Na čítanie

```
fr = fopen("pokus.txt", "r");
```

- Na zápis

```
fw = fopen("pokus.txt", "w");
```

- Aj ďalšie režimy ...

- Binárny súbor – rb, wb

Práca so súborom v C

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

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

```
int getchar();
```

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

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

```
int putchar(int c);
```

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

```
scanf("format", argumenty);
```

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

```
printf("format", argumenty);
```

Práca so súborom v C

- Ukončenie práce so súborom
- Keď už súbor ďalej nepotrebujeme, môžeme ho uzatvoriť

```
fclose(f) ;
```

- Nespoliehať sa na to, že po skončení program sa súbor uzatvorí sám
 - Neuzatvorený súbor môže spôsobovať problémy
 - Môžu sa stratiť data, atď

Práca so súborom v C

- **Testovanie konca riadku**
- Koniec riadku – EOLN (označenie, nie konštanta)
 - `\n` – pre čítanie aj zápis
 - `\n` – je buď CR, LF alebo CRLF podľa systému

Práca so súborom v C

- Testovanie konca súboru - EOF
- Koniec súboru – konštanta EOF
 - Symbolická konštanta, spravidla definovaná v stdio.h
 - Hodnotu má spravidla -1
 - EOF je int, nie char!!

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

Práca so súborom v C

- Testovanie konca súboru – `feof()`
- Koniec súboru – makro `int feof(FILE *stream)` vracia
 - TRUE ak posledné čítanie bolo za koncom súboru
 - FALSE (nulu) ak sme pri čítaní ešte nedošli na koniec
- Využitie pri čítaní binárneho súboru
 - Znak `0xFF` môže byť nesprávne chápaný ako EOF (neskôr ...)

```
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);
```

Práca so súborom v C

- Testovanie správnosti otvorenia súboru
- Čo ak otváram neexistujúci súbor? Čo ak nemám práva čítať súbor?
- **fopen()**
 - V úspešnom prípade vráti ukazovateľ na súbor
 - V neúspešnom prípade vráti NULL (0)

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

Práca so súborom v C

- Testovanie správnosti zatvorenia súboru
- Analogicky k testovaniu správnosti otvárania
- **fclose()**
 - V úspešnom prípade vráti hodnotu 0
 - V neúspešnom prípade vráti hodnotu konštanty EOF

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


Footnotes

- Prednáška je dostupná na YouTube:
<https://www.youtube.com/watch?v=ne36Fog2auo>

V prednáške boli použité materiály zo slidov prednášok ZPrPr1 od Gabriely Grmanovej.

Q?