Přehledový test znalostí

https://moodle.vut.cz/mod/quiz/view.php?id=320652

Přehledový test znalostí studentů o programování v C.
Pokrývá témata procvičovaná v počítačových laboratořích.

Test je nepovinný, ale doporučený. Výsledky testu mohou být zohledněny i v hodnocení cvičení.

Test je otevřen

od soboty 30. září 2023, 8:00 do pátku 13. října 2023, 23:59.

Proměnná v jazyce C

Proměnná je v matematice pojmenovaná obecná hodnota. V jazyce C a jiných imperativních jazycích však má slovo *proměnná* trochu odlišný význam.

Jde o *paměťové místo*, které může uchovávat hodnotu určitého typu. Tato uložená hodnota se může během výpočtu měnit.

Takové paměťové místo se někdy přesněji nazývá *přepisovatelná proměnná* (*mutable variable*). V jazyce C hovoříme krátce o *proměnné*.

Paměťové místo (proměnnou) definujeme zápisem:

```
double   x;
```

typ ukládaných hodnot — jméno proměnné

nebo

```
int      m   =  100;
```

typ ukládaných hodnot — jméno proměnné — počáteční vložená hodnota

Přiřazovací příkaz

Změnit hodnotu uloženou v proměnné lze *přiřazovacím příkazem*.

```
x     =     2.0 * pi ;
```

levá strana přiřazení	přiřazovací operátor	pravá strana přiřazení (výraz)

Pozor na rozdílné zacházení s proměnnou na levé a na pravé straně přiřazení:

- Proměnná na levé straně (tvoří tzv. *l-výraz*) a stojí sama za sebe.
- Proměnná na pravé straně (v tzv. *r-výrazu*) označuje hodnotu, kterou toto paměťové místo právě obsahuje. Kompilátor provádí automatické *dereferencování* proměnné.

```
m     =     m     +     1 ;
```

proměnná, do níž se přiřazuje

hodnota obsažená v proměnné

Poznámka: Kromě základního přiřazovacího operátoru `=` má C také operátory `+=`, `-=`, `*=`, `^=` apod.

Výměna hodnot ve dvou proměnných

Ze stavu, kdy v proměnné `a` je hodnota x a v proměnné `b` je hodnota y, chceme přejít do stavu, kdy jsou hodnoty v proměnných obráceně.
Použijeme pomocnou proměnou pro přechodné uchování hodnoty.
Pomocná proměnná musí mít stejný typ jako proměnné `a`, `b`.

```
{ int p = a;
  a = b;
  b = p;
}
```

V C se pro výměnu dvou hodnot používá trik využívající toho, že operace bitové nonekvivalence tvoří grupu, v níž je každý prvek svou vlastní inverzí.
Bitovou nonekvivalenci (operaci `^`) lze použít na hodnoty všech typů.

```
{ a ^= b;
  b ^= a;
  a ^= b;
}
```

Cykly

Cyklus s podmínkou na začátku

`while` (*podmínka*) *tělo cyklu*

Cyklus s podmínkou na konci

`do` *tělo cyklu* `while` (*podmínka*)

Platí

`while(`e`)` B $\equiv$ `if(`e`) { do` B `while(`e`) }`

`do` B `while(`e`)` $\equiv$ `{` B`; while(`e`)` B `}`

Cyklus for

`for (`p_1`;` p_2`;` p_3`)` *tělo cyklu*

- Na začátku se jednou provede *počáteční příkaz* p_1`;`
  (p_1 může být výraz nebo definice nové proměnné)
- Dokud je splněna *podmínka* p_2, provádí se opakovaně *tělo cyklu* vždy následované *příkazem kroku* p_3`;`

Platí

`for (`p_1`;` p_2`;` p_3`)` B	$\equiv$	`{` p_1`;` `while (`p_2`) {` B`;` p_3`; } }`
`while (`e`)` B	$\equiv$	`for (;`e`;)` B
`do` B `while (`e`)`	$\equiv$	`{` B`;` `for (;`e`;)` B `}`

Poznámka: Prázdná podmínka v hlavičce cyklu for se vyhodnotí jako pravdivá. Proto například `for (;;);` znamená zacyklení.

Příkazy break a continue v cyklu

`continue`

ukončí právě prováděnou iteraci těla cyklu.

`break`

slouží k okamžitému ukončení cyklu.

Cyklus lze také násilně skončit ukončením výpočtu celé funkce, tedy příkazem

`return`

Faktoriál

Cvičení: Napište program, který se zeptá na číslo n $(0 \leqslant n \leqslant 20)$ a spočítá $n!$.

```
#include <stdio.h>

long int fact (int n) {
    long int z = 1;
    for (int i = 1; i <= n; i++)
        { z *= i; }
    return z;
}

int main (void) {
    int n;
    do { printf("Number 0 to 20: n = ");  scanf("%d", &n);
    } while (n < 0 || n > 20);

    printf ("%d! = %ld\n", n, fact(n));
    return 0;
}
```

Největší společný dělitel

Cvičení: Program spočítá největšího společného dělitele dvou čísel.

```
#include <stdio.h>
#include <stdlib.h>

int gcd (int a, int b) {
    int p;
    a = abs(a);  b = abs(b);
    if (a > b) { p = a; a = b; b = p; }
    while (a != 0) { p = a; a = b % a; b = p; }
    return b;
}

int main (void) {
    int a, b;
    printf("Two numbers: ");  scanf("%d %d", &a, &b);
    printf("GCD(%d, %d) = %d\n", a, b, gcd(a,b));
    return 0;
}
```

Největší číslo v poli

Cvičení: Načtěte do pole 5 čísel a pak určete největší z nich.

```
#include <stdio.h>
int main (void) {
    const int n = 5;
    int a[n];
    int max, imax;
    for (int i=0; i<n; i++) {
        printf("a[%d] = ", i);  scanf("%d", &(a[i]));
    }
    max = a[0];  imax = 0;
    for (int i=1; i<n; i++) {
        if (a[i] > max) { max = a[i]; imax = i; }
    }
    printf("Maximum: a[%d] == %d\n", imax, max);
    return 0;
}
```

Změní se sémantika programu, když nerovnost v podmínce `a[i]>max` vyměníme za neostrou? Jak?

Slučování seřazených posloupností

Cvičení: Načtěte dvě vzestupně seřazená pole celých čísel.
Vypište prvky obou polí vzestupně seřazené.

```
#include <stdio.h>

int main (void) {
  const int m = 5, n = 5;
  int a[m], b[n];
  int i, j;

  for (int i=0; i<m; i++) { printf("a[%d] = ", i); scanf("%d", &a[i]); }
  for (int j=0; j<n; j++) { printf("b[%d] = ", j); scanf("%d", &b[j]); }

  for (i=0, j=0; i<m && j<n;)
      { printf ("%d ", a[i]<=b[j] ? a[i++] : b[j++]); }
  for (int ii = i; ii<m; ii++) { printf ("%d ", a[ii]); }
  for (int jj = j; jj<n; jj++) { printf ("%d ", b[jj]); }
  printf ("\n");
  return 0;
}
```