Úlohy na 8. cvičenie

Prvé 2 úlohy riešte na papier.

1. Doplňte hodnoty premenných po vykonaní týchto príkazov:

```
1  {
2     int x;
3     int * p;
4     p = &x;
5     *p = 2;
6     x = *p + x;
7  }
```

Riešenie:

	hodnota:	adresa:
int * p		104
int x		100

2. Doplňte hodnoty premenných po vykonaní týchto príkazov:

```
1  {
2     int a, b;
3     int * p;
4     p = &b;
5     b = 1;
6     a = *p + 1;
7  }
```

Riešenie:

	hodnota:	adresa:
int * p		108
int b		104
int a		100

Ďalej riešte úlohy v Turingu.

1. Do programu doplňte na vyznačené miesta:
   - referenčný operátor &
   - dereferenčný operátor *
   - nič

   tak, aby program správne pracoval s 3 premennými `i`, `j` a `k` typu `int` a 3 ukazovateľmi `p`, `q` a `r` na typ `int`. Program načíta do premenných 3 celé čísla a zistí, či súčet prvých 2 načítaných čísel sa rovná tretiemu načítanému číslu. Ak sa rovná, program vypíše správu `sucet ok`, inak správu `sucet nesedi`.

```
// program zisti, ci sucet prvych dvoch nacitanych cisel
// je rovny tretiemu
#include <stdio.h>

int main() {
        int i, j, k, *p, *q, *r;

        scanf("%d %d %d", □i, □j, □k); // nacitanie hodnot
                                       //do premennych i, j, k

        □p = □i;                // p ukazuje na i
        □q = □j;                // q ukazuje na j
        □r = □k;                // r ukazuje na k

        if (□p + □q == □r)      // ak sucet hodnot, na ktore ukazuju p
                                // a q je rovny hodnote, kam ukazuje r
                printf("sucet ok\n");
        else
                printf("sucet nesedi\n");

        return 0;
}
```

2. Do programu doplňte na vyznačené miesta:
   a. referenčný operátor &
   b. dereferenčný operátor *
   c. nič

   tak, aby program správne pracoval s 3 premennými `c1`, `c2` a `c3` typu `char` a jedným ukazovateľom `p` na typ `char`. Program načíta znak do premennej `c1` a ukazovateľ `p` nasmeruje na premennú `c2`. Pomocou ukazovateľa `p` potom program priradí do premennej `c2` znak o 1 väčší ako je načítaný znak a do `c3` znak o 2 väčší ako načítaný znak. Na konci program vypíše správu `znaky ok`, ak v `c1`, `c2` a `c3` sú 3 po sebe idúce znaky.

```
// program nacita znak do c1, pomocou ukazovatela priradi
// o 1 vacsi znak do c2 a o 2 vacsi znak do c3, na konci
// skontroluje, ci sa to podarilo
#include <stdio.h>

int main() {
        char c1, c2, c3;    // definicia 3 premennych typu char
        char *p;            // definicia ukazovatela na char

        scanf("%c", &c1);   // nacitanie znaku do premennej c1

        p = &c2;       // p ukazuje na c2
        *p = c1 + 1;   // na miesto, kam ukazuje p sa priradi
                       // o 1 vacsia hodnota ako je v c1
        c3 = *p + 1;   // do c3 sa priradi o 1 vacsia hodnota
                       // ako ta, na ktoru ukazuje p

        printf("%c %c %c\n", c1, c2, c3);  // vypis znakov
        if (c2 == c1+1 && c3 == c2 + 1)    // ak v c1, c2 a c3 su
                                           // 3 po sebe iduce znaky
                printf("znaky ok\n");

        return 0;
}
```

3. Napíšte program, v ktorom vo funkcii `main` budete mať deklarované dve premenné typu char – `c1`, `c2` a jeden ukazovateľ na char – `p_c`. Zo štandardného vstupu postupne načíta znaky do premennej `c1`, pokiaľ nenačíta znak `'*'`. Každý načítaný znak prekopíruje do premennej `c2` pričom nebude použitý príkaz `c2 = c1`. Nepoužívajte ani žiadne iné premenné ako `c1,c2, p_c`. Koľkými spôsobmi sa to dá urobiť? Výstupom programu je jeden riadok pre každý načítaný znak. Každý z týchto riadkov je ukončený znakom konca riadku a má mať takýto formát:
`c1: x (y), c2: z (p), p_c: q (r)`
kde `x` je obsah premennej `c1` a `y` adresa tejto premennej. Podobne, `z` je obsah premennej `c2` a `p` jej adresa. `q` predstavuje hodnotu premennej, na ktorú ukazuje ukazovateľ `p_c` a `r` je hodnota `p_c`.
Ukážka vstupu:
`jazyk C*`
Ukážka výstupu:

```
c1: j (0012FF7C), c2: j (0012FF78), p_c: j (0012FF7C)↵
c1: a (0012FF7C), c2: a (0012FF78), p_c: a (0012FF7C)↵
c1: z (0012FF7C), c2: z (0012FF78), p_c: z (0012FF7C)↵
c1: y (0012FF7C), c2: y (0012FF78), p_c: y (0012FF7C)↵
c1: k (0012FF7C), c2: k (0012FF78), p_c: k (0012FF7C)↵
c1:   (0012FF7C), c2:   (0012FF78), p_c:   (0012FF7C)↵
```

```
c1: C (0012FF7C), c2: C (0012FF78), p_c: C (0012FF7C)↵
```

4. Vytvorte funkciu, ktorá prevedie zlomok do základného tvaru. Funkcia má ako argumenty dve čísla (volané odkazom). Argumenty vyjadrujú čitateľa a menovateľa zlomku. Funkciu otestujte v krátkom programe, ktorého vstupom je jeden riadok obsahujúci 2 celé čísla oddelené medzerou a ukončený znakom konca riadku. Prvé z čísel predstavuje čitateľa a druhé menovateľa zlomku. Výstupom programu je riadok ukončený koncom riadku obsahujúci správu `Zakladny tvar zlomku: c/m`, kde `c` je čitateľ a `m` menovateľ zlomku v základnom tvare.
   Ukážka vstupu:
   ```
   12 60↵
   ```
   Ukážka výstupu:
   ```
   Zakladny tvar zlomku: 1/5↵
   ```

5. Napíšte program na výpočet obsahu a obvodu obdĺžnika. Vytvorte funkciu `nacitaj`, ktorá načíta a pomocou argumentov vráti dve reálne čísla. Ďalej vytvorte funkciu `vypocitaj`, ktorá ako argumenty dostane dĺžky strán obdĺžnika a prostredníctvom argumentov vráti obsah a obvod obdĺžnika. V hlavnom programe volajte funkciu na načítanie rozmerov obdĺžnika `nacitaj` a funkciu na výpočet obsahu a obvodu obdĺžnika `vypocitaj`. Program vypíše obsah a obvod obdĺžnika zaokrúhlený na tri desatinné miesta.
   Ukážka vstupu:
   ```
   3.5 4.75↵
   ```
   Výstup pre ukážkový vstup:
   ```
   Obsah: 16.625↵
   Obvod: 16.500↵
   ```