

### Úloha X1: Prepínače gcc a rôzne štandardy C

Textový súbor z úlohy 2 upravte tak, aby sa na obrazovku vypísala veta "Program funguje." a následne sa vypísali čísla od 1 po 9 každé do samostatného riadku. Pri kompilácii použite štandard c90 (prepínač -std=c90 ) a zapnite výpis varovania (prepínač -Wall ). Pokúste sa zdôvodniť výsledok kompilácie a upraviť zdrojový kód tak aby fungoval pri danom prepínači.

*Správne riešenie:*

```
C:\Users\JanoPC_1\Documents\C porgrams>gcc testgcc.c -std=c90 -Wall -o run
C:\Users\JanoPC_1\Documents\C porgrams>run
Program funguje.1
2
3
4
5
6
7
8
9
C:\Users\JanoPC_1\Documents\C porgrams>
```

*Nápoved':*

Ak ste zdrojový kód upravili nasledovne::

```
testgcc.c - Notepad
File Edit Format View Help
#include <stdio.h>

int main(void)
{
    printf("Program funguje.");
    for(int i=1;i<10;i++)
        printf("%d\n",i);
    return(0);
}
```

nastal problém s definovaním riadiacej premennej i, štandard c90 to nepodporuje. Dostali ste nasledovný výpis:

```
C:\Users\JanoPC_1\Documents\C porgrams>gcc testgcc.c -std=c90 -Wall -o run
testgcc.c: In function 'main':
testgcc.c:6:2: error: 'for' loop initial declarations are only allowed in C99 or C11 mode
   6 |   for(int i=1;i<10;i++)
     |   ^~~~~
testgcc.c:6:2: note: use option '-std=c99', '-std=gnu99', '-std=c11' or '-std=gnu11' to compile your code
```

Pre daný štandard (C90) nie je prípustné aby premenná bola priamo v cykle for deklarovaná a preto sa vypíše chyba resp. varovanie, ktoré by mal programátor odstrániť, taktiež je navrhnutá alternatíva aby sa pri kompilácii použil iný štandard napr. c99, pri tomto štandarde je už program spustiteľný, viď ukážka:

```
C:\Users\JanoPC_1\Documents\C porgrams>gcc testgcc.c -std=c99 -Wall -o run
C:\Users\JanoPC_1\Documents\C porgrams>run
Program funguje.1
2
3
4
5
6
7
8
9
C:\Users\JanoPC_1\Documents\C porgrams>
```

Úprava nastane tak že premennú *i* definujete pred cyklom `for`.

Viac o príkazoch gcc kompilátora na nasledovnej linke:

<https://gcc.gnu.org/onlinedocs/gcc-12.1.0/gcc/C-Dialect-Options.html#C-Dialect-Options>)

Iné užitočné linky:

<http://kmlinux.fjfi.cvut.cz/~fabiadav/cecko/poznamky-k-jazyku-c/preklad-pomoci-gcc>

<http://www.fit.vutbr.cz/~martinek/clang/gcc.html>

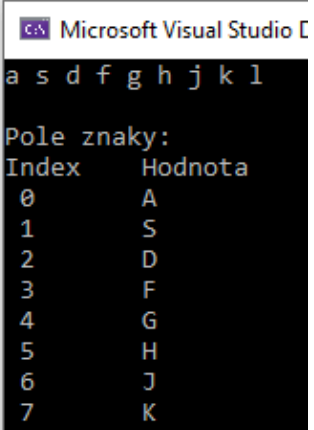
Časová náročnosť: 20min

## Úloha X2: Statické jednorozmerné pole znakov a ich konverzia

Definujte statické jednorozmerné pole s názvom znaky pre 8 znakov. Zo štandardného vstupu (klávesnice) načítajte znaky do poľa znakov. prostredníctvom funkcie ZMENAchar() vykonajte zmenu znakov na veľké znaky a vo funkcii main() ich vypíšte na obrazovku spolu aj s ich indexami.

*Správne riešenie:*

```
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  void ZMENAchar(char pole[], int velk) {
6      int j;
7      for (j = 0; j < velk; j++) {
8          if (pole[j] >= 'a' && pole[j] <= 'z')
9              pole[j] = pole[j] - ('a' - 'A');
10     }
11 }
12 int main(int argc, char* argv[]) {
13     int i, j;
14     char znaky[8];
15     for (j = 0; j < 8; j++) {
16         scanf("%c", &znaky[j]);
17     }
18     ZMENAchar(znaky, 8);
19     printf("\nPole znaky: \nIndex \t Hodnota");
20     for (j = 0; j < 8; j++) {
21         printf(" \n %d \t %c", j, znaky[j]);
22     }
23     return 0;
24 }
25
```



Index	Hodnota
0	A
1	S
2	D
3	F
4	G
5	H
6	J
7	K

*Nápoved:*

- keďže vo funkcii ideme pracovať s polom, musíme si do funkcie poslať aj veľkosť daného poľa. (príklad : ["Pass Arrays as Function Parameters"](#))
- používame ASCII tabuľku a teda aj jej hodnôt, pre zistenie rozdielu medzi malým a veľkým písmenom použije kód ('a' - 'A') namiesto čísla 32, ak by sme použili toto číslo, tak sa zhorší čitateľnosť Vášho kódu pre osob, ktorá nemá potuchy odkiaľ autor nabral číslo 32
- tento príklad nadobúda väčší význam, keď si preberieme funkcie, polia a ukazovatele

**Časová náročnosť:** 10 minút

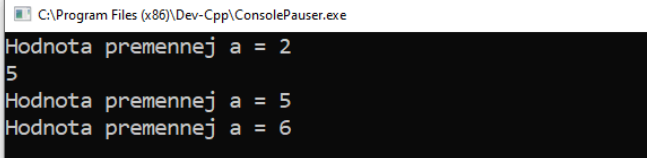
### Úloha X3: Ukazovatele 1

Vytvorte program v ktorom definujete premennú a typu int, s inicializáciou na hodnotu 2; Vytvorte smerník pa, ukazujúci na adresu premennej a; Realizujte načítanie vstupu z klávesnice - celého čísla do premennej a. Následne realizujte zvýšenie hodnoty premennej a o 1, bez toho, aby ste vo výraze použili premennú a. Opakovane po jednotlivých krokoch vypíšte hodnotu premennej a na obrazovku, a zdôvodnite ju.

Časová náročnosť: 5 min.

Riešenie:

```
1  #include<stdio.h>
2
3  int main(void)
4  {
5
6      int a = 2, *pa= &a;
7      printf("Hodnota premennej a = %d\n", a);
8      scanf(" %d", &a);
9      printf("Hodnota premennej a = %d\n", a);
10     *pa = *pa+1;
11     printf("Hodnota premennej a = %d\n", a);
12     return 0;
13 }
```



Nápoveda:

<https://igor.podlubny.website.tuke.sk/C/Kap9.htm>

<https://www.pcforum.sk/pochopenie-smernikov-v-jazyku-c-vt134003.html>

### Úloha X4: Ukazovatele 2

Vytvorte program, ktorý z klávesnice načítá 2 celé čísla do premenných a, b; Následne vytvorte a zavolajte funkciu swap(), ktorá zabezpečí vzájomnú výmenu hodnôt v premenných a, b. Na výmenu hodnôt vo funkcií použite smerníky. Pôvodné hodnoty premenných ako aj vymenené (po aplikovaní funkcie swap) vypíšte na obrazovku.

Vstup: 2 5

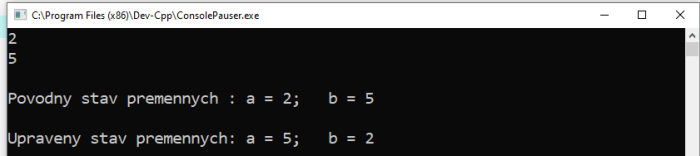
Výstup:

Povodny stav premennych : a = 2; b = 5

Upraveny stav premennych: a = 5; b = 2

Riešenie:

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include<stdio.h>
3
4 void swap(int *c1, int *c2)
5 {
6     int pom = *c1;
7     *c1 = *c2;
8     *c2 = pom;
9 }
10 int main(void)
11 {
12     int a, b;
13     scanf(" %d %d", &a, &b);
14     printf("\nPovodny stav premennych : a = %d; b = %d \n", a, b);
15     swap(&a, &b);
16     printf("\nUpraveny stav premennych: a = %d; b = %d \n", a, b);
17     return 0;
18 }
19
```



Nápoveda:

- pokiaľ sa vo funkcií majú meniť hodnoty premenných a zmenené hodnoty majú byť zachované aj po návrate do mainu, je potrebné aby do funkcie vstupovali adresy premenných
- Príklad vynesenia viacerých hodnôt z funkcie v jazyku C je dostupný : <https://www.techiedelight.com/return-multiple-values-function-c/>
- Pre výmenu hodnôt vo funkcií použite vo funkcií pomocnú premennú do ktorej si dočasne uložte hodnotu prepisovanej premennej
- <https://www.pcforum.sk/pochopenie-smernikov-v-jazyku-c-vt134003.html>

Časová náročnosť: 5 minút

### Úloha X5: Ukazovatele 3

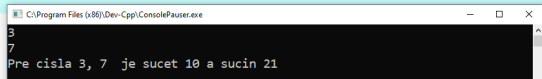
V jazyku C vytvorte program, ktorý zo štandardného vstupu načíta 2 celé čísla. Z nich nasledne vypočíta ich súčet a súčin, ktorý vypíše na obrazovku. Výpočet súčtu a súčinu realizujte v spoločnej funkcii. Vypočítané hodnoty vyneste z funkcie do mainu za použitia smerníkov a v maine ich vypíšte na obrazovku.

Vstup: 3 7

Výstup: Pre cisla 3, 7 je sucet 10 a sucin 21

Riešenie:

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include <stdio.h>
3
4 void sucetSucin(int a, int b, int *sucet, int *sucin)
5 {
6     *sucet = a+b;
7     *sucin = a*b;
8 }
9
10 int main(void)
11 {
12     int x, y;
13     int vysl, vys2;
14
15     scanf("%d %d", &x, &y);
16     sucetSucin(x, y, &vysl, &vys2);
17
18     printf("Pre cisla %d, %d je sucet %d a sucin %d \n", x, y, vysl, vys2);
19     return 0;
20 }
```



Nápoveda:

- pokiaľ sa vo funkcií majú meniť hodnoty premenných a zmenené hodnoty majú byť zachované aj po návrate do mainu, je potrebné aby do funkcie vstupovali adresy premenných
- Príklad vynesenia viacerých hodnôt z funkcie, v jazyku C je dostupný : <https://www.techiedelight.com/return-multiple-values-function-c/>
- <https://www.pcforum.sk/pochopenie-smernikov-v-jazyku-c-vt134003.html>

Časová náročnosť: 5 minút

### Úloha X6: Ukazovatele 4

V jazyku C vytvorte program, v ktorom zadefinujete premennú c typu char. Taktiež premennú n typu int, a premennú r typu double. Ku každej z týchto premenných vytvorte smerník (pointer / ukazovateľ), ktorý bude na danú premennú ukazovať. Tieto smerníky nazvite pn, pc, a pr. Do premenných c, n, r načítajte vstupy z klávesnice.

Následne vypíšte na obrazovku:

- adresy premenných aj smerníkov
- hodnoty premenných aj smerníkov
- hodnoty výrazov \*pn, \*pc, \*pr (v príslušnom type)

Preskúmajte výstupy na obrazovke a všimnite si vzťahy medzi nimi. Uvažujte o možnosti, ako načítať hodnoty zo štandardného vstupu do premenných c, n, r bez použitia týchto premenných.

Upozornenie: Vypisované adresy sa budú na jednotlivých počítačoch/zariadeniach výrazne líšiť.

Upozornenie 2: Niektoré kompilátory neakceptujú formatovaný zápis pre výpis adresy (vo funkcii printf) v tvare %d; v takom prípade použite zápis %p

Vstupy: a 2 5.5

Výstup:

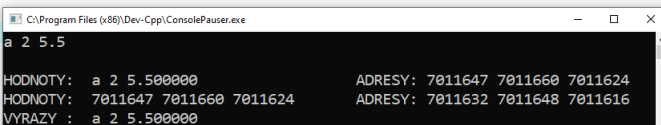
```
HODNOTY: a 2 5.500000          ADRESY: 7011647 7011660 7011624
HODNOTY: 7011647 7011660 7011624  ADRESY: 7011632 7011648 7011616
VYRAZY : a 2 5.500000
```

Nápoveda:

- <https://igor.podlubny.website.tuke.sk/C/Kap9.htm>
- <https://www.pcforum.sk/pochopenie-smernikov-v-jazyku-c-vt134003.html>

Riešenie:

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include<stdio.h>
3 int main(void)
4 {
5     int n, *pn;
6     char c, *pc;
7     double r, *pr;
8
9     pn = &n;
10    pc = &c;
11    pr = &r;
12
13    scanf(" %c %d %lf", &c, &n, &r);
14
15    // kedze smerniky ukazuju na dane premenne
16    // preto mozeme nacitanie realizovat aj formou:
17    // scanf(" %c %d %lf", pc, pn, pr); // bez pouzitia premennych n, c, r
18
19    // vypisy
20    printf("\nHODNOTY:  %c %d %lf    \t\tADRESY: %d %d %d", c, n, r, &c, &n, &r);
21    printf("\nHODNOTY:  %d %d %d    \t\tADRESY: %d %d %d", pc, pn, pr, &pc, &pn, &pr);
22    printf("\nVYRAZY :  %c %d %lf    ", *pc, *pn, *pr);
23    return 0;
24 }
25
```



Časová náročnosť: 15 minút

### Úloha X7: Ukazovatele 5

Vytvorte statické 3-prvkové pole celých čísel a pomenujte ho ints. Taktiež vytvorte 3-prvkové pole reálnych čísel a pomenujte ho doubles. Následne vytvorte 2 smerníky, jeden typu int\* s názvom pi a druhý double\* s názvom pd. Jednotlivým prvkom v poliach priradte nejaké platné rozlíšiteľné hodnoty. V cykle prejdite jednotlivé prvky poľa. Pre každý prvok poľa naň nasmerujte príslušný pointer. Vypíšte adresy a hodnoty jednotlivých prvkov poľa. Podobne tak aj použitím daného smerníka. Vypisy priamym indexovaním prvkov v poli a cez smerník by sa mali zhodovať. Použitím makra sizeof(<typ>) zistíte veľkosti typov double a int, ako aj jednotlivých typov smerníkov. Vypíšte ich na obrazovku.

Všimnite si, ako sa menia adresy jednotlivých prvkov v poli, a aký je súvis s veľkosťou zaberaného typu.

Inšpirujte sa priloženým kódom, ktorý rieši ulohu pre celé čísla, zatiaľ bez použitia smerníkov:

```
#define _CRT_SECURE_NO_WARNINGS
#define POCET 3
#include<stdio.h>
int main(void)
{
    int i, ints[POCET], *pi;           // zadefinujeme premenne
    // pridať pole typu double a smerník na tento typ
    for (i = 0; i < POCET; i++) // zadefinujeme hodnoty prvkov poľa
    {
        ints[i] = 5*i+1; }
    for (i = 0; i < POCET; i++)    // vypisy v cykle
    {
        pi = &ints[i];
        printf("\nindex: %d\n", i);
        printf("ints[%d] = %d  \t&ints[%d] = %d\n", i, ints[i], i, &ints[i]);

        // doplnenie prístupu a vypisu použitím smerníka pi

        // celé zopakovať pre pole reálnych čísel typu double
    }
    return 0;
}
```

upozornenie: Pre rozličné prostredia / kompilátory niektoré typy nezaberajú vždy rovnaký počet bytov.

Riešenie:



```

1 #define _CRT_SECURE_NO_WARNINGS
2 #define POCET 3
3 #include<stdio.h>
4 int main(void)
5 {
6     // zadefinujeme premenne
7     int i, ints[POCET], *pi;
8     double doubles[POCET], *pd;
9
10    for (i = 0; i < POCET; i++) // zadefinujeme hodnoty prvkov pola
11    {
12        ints[i] = 5*i+1;
13        doubles[i] = 0.1 * i;
14    }
15
16    // vypisy v cykle
17    for (i = 0; i < POCET; i++)
18    {
19        pi = &ints[i];
20        pd = &doubles[i];
21
22        printf("\nindex: %d\n", i);
23        printf("ints[%d] = %d \t&ints[%d] = %d\n", i, ints[i], i, &ints[i]);
24        printf("doubles[%d] = %lf \t&doubles[%d] = %d\n", i, doubles[i], i, &doubles[i]);
25
26        // vypis za pouzitia smernikov
27        printf("ints[%d] = %d \t&ints[%d] = %d\n", i, *pi, i, pi);
28        printf("doubles[%d] = %lf \t&doubles[%d] = %d\n", i, *pd, i, pd);
29    }
30
31    // vypisy velkosti sizeof jednotlivych typov
32    printf("\nPocet bytov pouzivanych na typ int %d, na typ double %d\n", sizeof(int), sizeof(double));
33    printf("Pocet bytov pouzivanych na typ int* %d, na typ double* %d\n", sizeof(int*), sizeof(double*));
34
35    return 0;
36 }

```

```

index: 0
ints[0] = 1      &ints[0] = 7011616
doubles[0] = 0.000000      &doubles[0] = 7011584
ints[0] = 1      &ints[0] = 7011616
doubles[0] = 0.000000      &doubles[0] = 7011584

index: 1
ints[1] = 6      &ints[1] = 7011620
doubles[1] = 0.100000      &doubles[1] = 7011592
ints[1] = 6      &ints[1] = 7011620
doubles[1] = 0.100000      &doubles[1] = 7011592

index: 2
ints[2] = 11     &ints[2] = 7011624
doubles[2] = 0.200000      &doubles[2] = 7011600
ints[2] = 11     &ints[2] = 7011624
doubles[2] = 0.200000      &doubles[2] = 7011600

Pocet bytov pouzivanych na typ int 4, na typ double 8
Pocet bytov pouzivanych na typ int* 8, na typ double* 8

```

Časová náročnosť: 35 min.