STU FIIT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Základy procedurálneho programovania 2

FIIT STU, Mlynská dolina

Aula Minor, pondelok 9:00

letný semester

2016/2017

Ideme podľa plánu

dátum	prednáška	8:00	9:00	cvičenie	obsah
13.2.	1		Opakovanie	1	Projekt 1 **Snehulienka**
20.2.	2		Algoritmické obchodovanie	2	
27.2.	3	Test 1	Riešenie testu 1, Snehulienka	3	
6.3.	4		Snehulienka (pokr.)	4	
13.3.	5	Test 2	Riešenie testu 2, Smerníky	5	Projekt 1: odovzdanie, konzultovanie
20.3.	6		Čítanie kódu, Hľadanie chýb v kóde	6	

Druhý priebežný test... Úloha A

- V nasledujúcich príkladoch napíš hlavičku funkcie a vysvetli ako bude s údajmi pracovať.

  a) Napíš hlavičku funkcie **pridaj**, ktorá do jednorozmerného poľa celých čísel (**int**) pridá na koniec nové číslo. Predpokladajte, že pole je dostatočne veľké aj pre nové číslo.

Druhý priebežný test... Úloha A

- a) Napíš hlavičku funkcie **pridaj**, ktorá do jednorozmerného poľa celých čísel (**int**) pridá na koniec nové číslo. Predpokladajte, že pole je dostatočne veľké aj pre nové číslo.
- Riešenie?

```
void pridaj(int *pole, int *n, int cislo)
  pole[(*n)++] = cislo;


pridaj(a, &n, 5);
```

Druhý priebežný test... Úloha A

- V nasledujúcich príkladoch napíš hlavičku funkcie a vysvetli ako bude s údajmi pracovať.

  b) Daný je názov súboru, v ktorom sú celé čísla (**int**). Napíš hlavičku funkcie **nacitaj**, ktorá zo súboru načíta všetky čísla do jednorozmerného poľa. V súbore je vopred neznámy počet čísel, ale môžete predpokladať, že máte v počítači dosť voľnej pamäte.

Druhý priebežný test... Úloha A

- b) Daný je názov súboru, v ktorom sú celé čísla (**int**). Napíš hlavičku funkcie **nacitaj**, ktorá
  zo súboru načíta všetky čísla do jednorozmerného poľa. V súbore je vopred neznámy počet čísel, ale môžete predpokladať, že máte v počítači dosť voľnej pamäte.
- Riešenie?
- Čo mám na vstupe: názov súboru (char*)
- Alternatívy tvaru funkcie
  1. **int* nacitaj(char *nazov, int *n)**
  2. **void nacitaj(char *nazov, int** pole, int *n)**
  3. **int nacitaj(char *nazov, int** pole)**
- Čo by som chcel na výstupe: jednorozmerné pole
  - adresu na začiatok poľa (int*), počet prvkov (int)
- Ukážka volania (2): nacitaj("input.txt", &pole, &n);

Druhý priebežný test... Úloha B

- Vo vstupnom súbore **znamky.txt** sa nachádzajú známky (A, B, C, D, E a Fx), ktoré študenti získali v predmete ZPrPr2. **Známky sú v súbore ako reťazce oddelené medzerami**. Napíšte (celý) program v jazyku C, ktorý na štandardný výstup vypíše histogram získaných známok a dosiahnuté počty do zátvorky podľa ukážky nižšie. Každá známka je v histograme reprezentovaná jednou hviezdičkou.

```
 A: **** (4)
 B: * (1)
 C: ******* (7)
 D: ********* (9)
 E: ***** (5)
Fx: *** (3)
```

Druhý priebežný test... Úloha B

- Čo má na vstupe: neusporiadané známky
  Fx Fx A A E A A B C C C C C C C
  D D C C D D D E D D D D D D
  E E E E Fx
- Riešenie?

```
int i, j, pocet[5] = 0;
while(scanf(“%s“, buf) > 0)
  pocet[buf[0]-’A’]++;
for (i = ‘A’; i <= ‘F‘; i++)
{
  printf(“%c: “, i);
  if (i == ’F’) printf(“x“);
  for (j = 0; j < pocet[i]; j++)
    printf(“*“);
  printf(“(%d)\n“, pocet[i]);
}
```

```
A: **** (4)
B: * (1)
C: ******* (7)
D: ********* (9)
E: ***** (5)
Fx: *** (3)
```

STU FIIT

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Základy procedurálneho programovania 2

Smerníky

13.3.2017

letný semester 2016/2017

Opakovanie – premenná

- **Premenná** je pomenovaný priestor v pamäti
  - previazanie identifikátora s pamaťou
- Začiatok (prvý byte) v pamäti, kde sú dáta pre premennú vyhradené, nazývame **adresa premennej**
  - **adresa je (obyčajné) číslo** – poradie prvého byte od začiatku (vyhradenej) pamäte
- Adresa sa označuje symbolom ampersand (&)
- Premenná x – **adresa x je &x**

- **Priradenie cez adresu** (px = &x)
  ***px = 42** je to isté ako **x = 42**

Opakovanie – premenná

- Adresa sa označuje symbolom ampersand (&)
- Premenná x – **adresa x je &x**
- Keď poznám adresu premennej, tak viem zmeniť dáta na príslušnej adrese (nepotrebujem nato meno premennej)
- Adresu môžem mat uloženú v premennej
  takúto premennú nazývame **smerník**
- **Priradenie cez adresu** (px = &x)
  ***px = 42** je to isté ako **x = 42**
- V tejto ukážke platí, že premenná **px** je smerník na premennú **x**

Smerník

- Smerník – premenná, ktorá obsahuje adresu
- Smerník môže mať dátový typ, napr. smerník na int (adresa pamäte v ktorej je vyhradené miesto pre int)
- Dátový typ „smerník na typ“ píšeme s hviezdičkou typ*, napr. **smerník na int** píšeme **int***
- Príklad:
  int i = 30; i je meno pre pamäť, kde je int
  int* p = &i; p je adresa premennej i
- Pre priradenie využitím smerníka použijeme operátor hviezdička (tzv. dereferencovanie smerníku):
  ***p = 20; je to isté ako i = 20;**

Opakovanie – Vymeň hodnotu (swap)

- Úloha: **Napíš program, ktorý vymení hodnotu dvoch premenných.**
- Program?
  - Jednorázový príkaz „niekde“ v programe
  - Znovupoužiteľná funkcia
- Vstup: **premenné int x a y**
- Výstup: **v premennej x bude pôvodná hodnota y, a v premennej y bude pôvodná hodnota x, ostatné premenné v programe zostanú neporušené**
- Postup: vytvorím si **pomocnú premennú tmp**, do ktorej si zapamätám pôvodnú hodnotu x: **tmp ← x, potom x ← y, y ← tmp**. Hotovo.

Opakovanie – Vymeň hodnotu (swap)

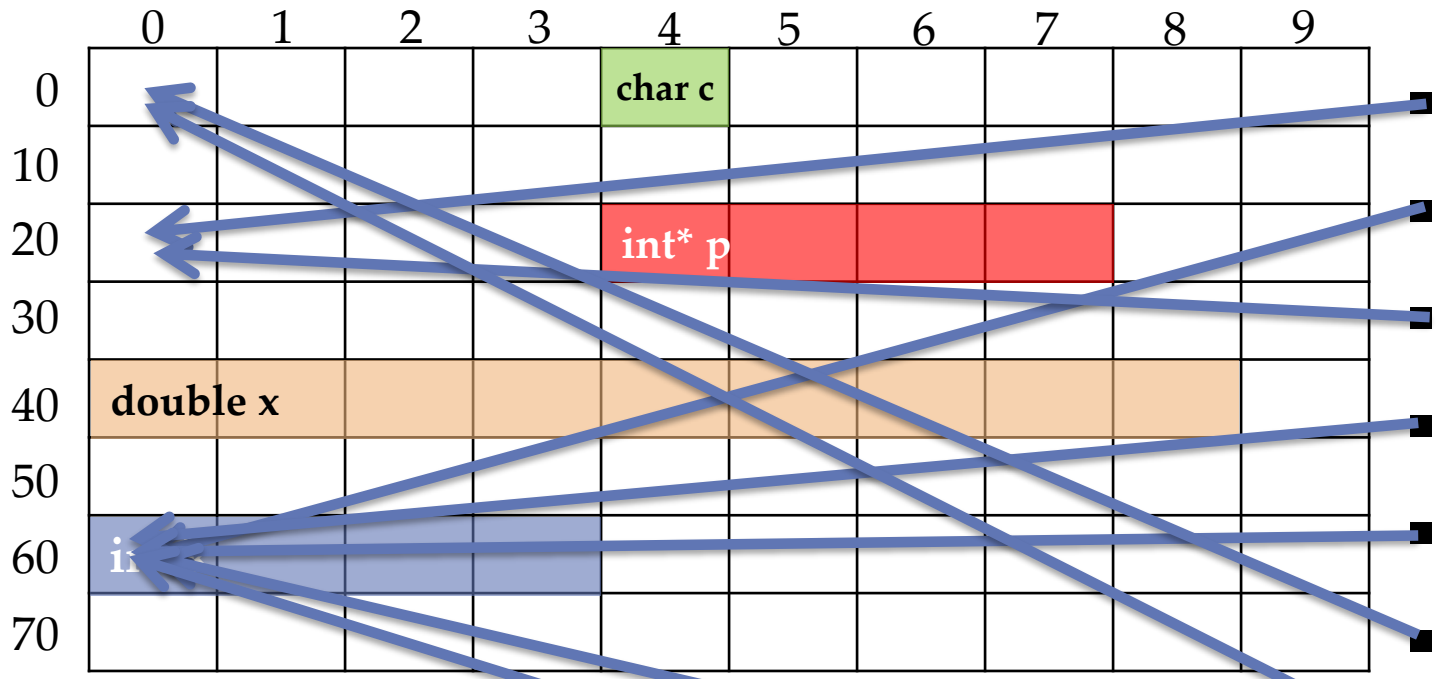
- Program
  - **Jednorázový príkaz „niekde“ v programe**

```
int tmp, u = 10, v = 20;
tmp = u; u = v; v = tmp;
```

  - **Znovupoužiteľná funkcia**

```
void swap(int *x, int *y)
{
    int tmp = *x;
    *x = *y;
    *y = tmp;
}
// pouzitie
int u = 10, v = 20;
swap(&u, &v);
```

Komplikovanejší príklad



- **p = hodnota**, znamená že menín adresu, na ktorú ukazuje smerník
- ***p = hodnota**, znamená, že mením obsah pamäte, na ktorú ukazuje smerník

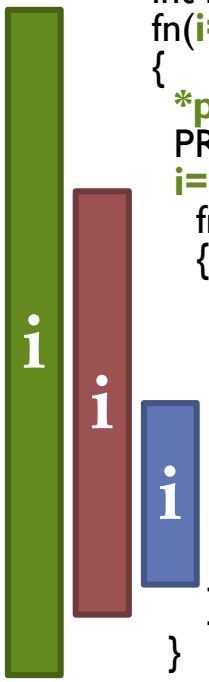
Smerníková aritmetika

- Adresu smerníka je možné určiť aj výpočtom
- Tzv. smerníková aritmetika
- Čo sa stane, keď k smerníku pripočítam +1?
  Smerník sa posunie na nasledujúcu pamäť posunutú o jednu veľkosť dátového typu na ktorý ukazuje
- Napr. int* p = 40;
  p = p+1; alebo p++;
  Hodnota p bude?
  44

Smerník v argumente funkcie

- Čo ak chceme vo volanej (vnorenej) funkcii upraviť premennú, ktorá je vo volajúcej funkcií:

```
fn(int i, int *p)
{
  *p++;
  if (i <= 0)
    PRINT koniec
   else
  {
    PRINT i
    i = i-2;
    fn(I);
  }
}
```



- int k=3, volanie fn(3, **&k**)

```
fn(i=3, p=&k)
{
  *p++ … nastaví k=4
  PRINT 3
  i=1
    fn(i=1, p=&i)
    {
      *p++ … nastaví i=2
      PRINT 1
      i=-1
        fn(i=-1, p=&i)
        {
          *p++ … nastaví i=0
          PRINT koniec
        }
    }
}
```

Ďalšie materiály (LMGTFY)

- http://denniskubes.com/2017/01/24/the-5-minute-guide-to-c-pointers/
- https://users.cs.cf.ac.uk/Dave.Marshall/C/node10.html
- http://cslibrary.stanford.edu/106/
- https://www.programiz.com/c-programming/c-pointers
- http://boredzo.org/pointers/
- https://pdos.csail.mit.edu/6.828/2010/readings/pointers.pdf

Nabudúce...

dátum	prednáška	8:00	9:00	cvičenie	obsah
13.2.	1		Opakovanie	1	Projekt 1 **Snehulienka**
20.2.	2		Algoritmické obchodovanie	2	
27.2.	3	Test 1	Riešenie testu 1, Snehulienka	3	
6.3.	4		Snehulienka (pokr.)	4	
13.3.	5	Test 2	Riešenie testu 2, Smerníky	5	Projekt 1: odovzdanie, konzultovanie
20.3.	6		Čítanie kódu, Hľadanie chýb v kóde	6	