Metóda pre sociálne programovanie a posudzovanie kódu

Michal Tomlein

Vedúci práce: Mgr. Jozef Tvarožek, PhD.

Dve časti metódy

- Sociálne programovanie
  - Tímový duch
- Posudzovanie kódu a výber vhodného posudzovatel'a
  - Synchrónne (live)

Ciele

- Preskúmať vzťah schopnosti spolupráce s osobnostnými charakteristikami
- Vyberať vhodného posudzovateľa na základe osobnostných charakteristík

Myšlienka

- Živé synchrónne posudzovanie
  - Vhodné pre kontext výučby
    - Netradičné a zaujímavé
- Živá diskusia o kóde
  - Možnosť pomôcť posudkom k riešeniu

Prvý prototyp

Firefox

127.0.0.1:7000/fiit/spp

Google

Peoplia | 5 Histogram | 2 Najčastejši znak | 0 Cézarova šifra | 0 Dešifrovanie | 0 Tajná správa | Bc. Michal Tomlein

Napíšte funkciu `char most_frequent_char(const char *str)`, ktorá vráti najčastejšie sa vyskytujúce písmeno malej anglickej abecedy v zadanom reťazci. Predpokladajte, že môžete využiť funkciu `histogram`

sifry-2.c | Štandardný vstup | Štandardný výstup

```c
// sifry-2.c -- Sifry - Najcastejsi znak

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

int *histogram(const char *str);

char most_frequent_char(const char *str)
{
  int count[256], i, max;

  // Tu napis svoj kod
  for (i = 0; i < 256; i++)
    count[i] = 0;
  for (i = 0; str[i]; i++)
    if (str[i] >= 'a' && str[i] <= 'z')
      count[str[i]]++;
  for (max = 0, i = 1; i < 256; i++)
    if (count[max] < count[i])
      max = i;
  return max > 0 ? (char)max : '?';
}

int main()
{
  printf("%c\n", most_frequent_char("")); // ?
  printf("%c\n", most_frequent_char("hell0000000")); // l
  printf("%c\n", most_frequent_char("aardvark")); // a
  return 0;
}
```

Kompilácia ... OK

sifry-2.c: In function 'most_frequent_char':

sifry-2.c:17:7: warning: array subscript has type 'char'

Kompilovať | Spustiť program | Odoslať riešenie

Otázky / komentáre:

Michal: Čo je to subscript?

Jožko: To je výraz medzi [], ktorým pristupuješ k prvku poľa.

Jožko: Stačí, keď ho pretypuješ.

Michal: Na char?

Jožko: Áno.

Tvoja otázka / komentár:

Pošli

Napíšte funkciu `int *histogram(const char *str)`, ktorá vytvorí histogram výskytov malých písmen anglickej abecedy v reťazci `str`.

sifry-1.c | Štandardný vstup | Štandardný výstup

```c
// sifry-1.c -- Sifry - Histogram

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

int *histogram(const char *str)
{
  // hist je pole 26 intov s hodnotou 0
  int *hist = (int *)calloc(26, sizeof(int));

  int i;
  for (i = 0; str[i]; ++i) {
    if (str[i] >= 'a' && str[i] <= 'z')
      hist[str[i] - 'a'] ++ 1;
  }

  return hist;
}

int main()
{
  int *hist = histogram("hello");
  free(hist);
  return 0;
}
```

Kompilácia ... CHYBA (kliknite na chyby nižšie)

sifry-1.c: In function 'histogram':

sifry-1.c:14:29: error: expected ';' before numeric constant

Správy pre mňa:

Michal: ++ je unárny operátor, nemôžeš zaň napísať číslo.

Nová správa:

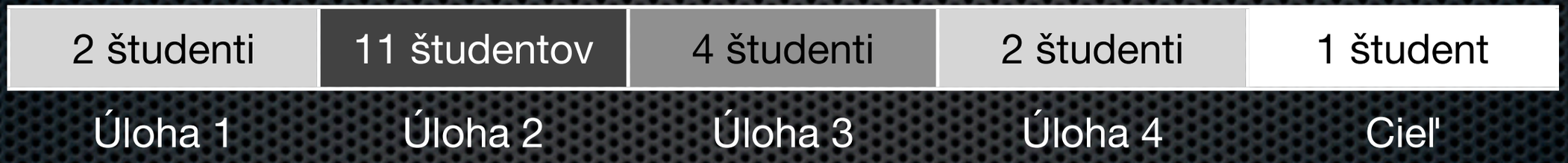
Pošli správu

Sociálne programovanie

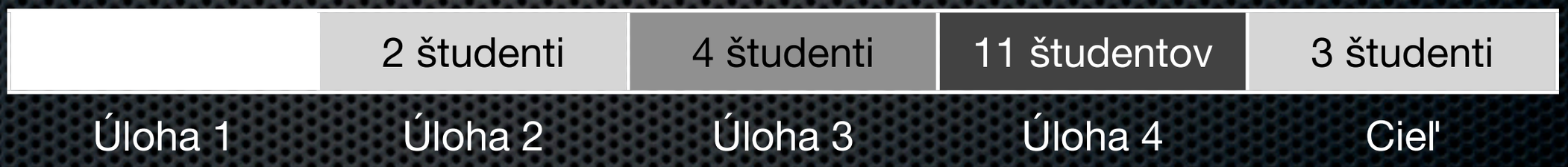
- Tímový duch
- Nadväzujúce úlohy
- Cieľ: úspešné vyriešenie úloh **celou skupinou**
- Prostriedky:
  - posudzovanie kódu
  - vizualizácia postupu

Vizualizácia postupu

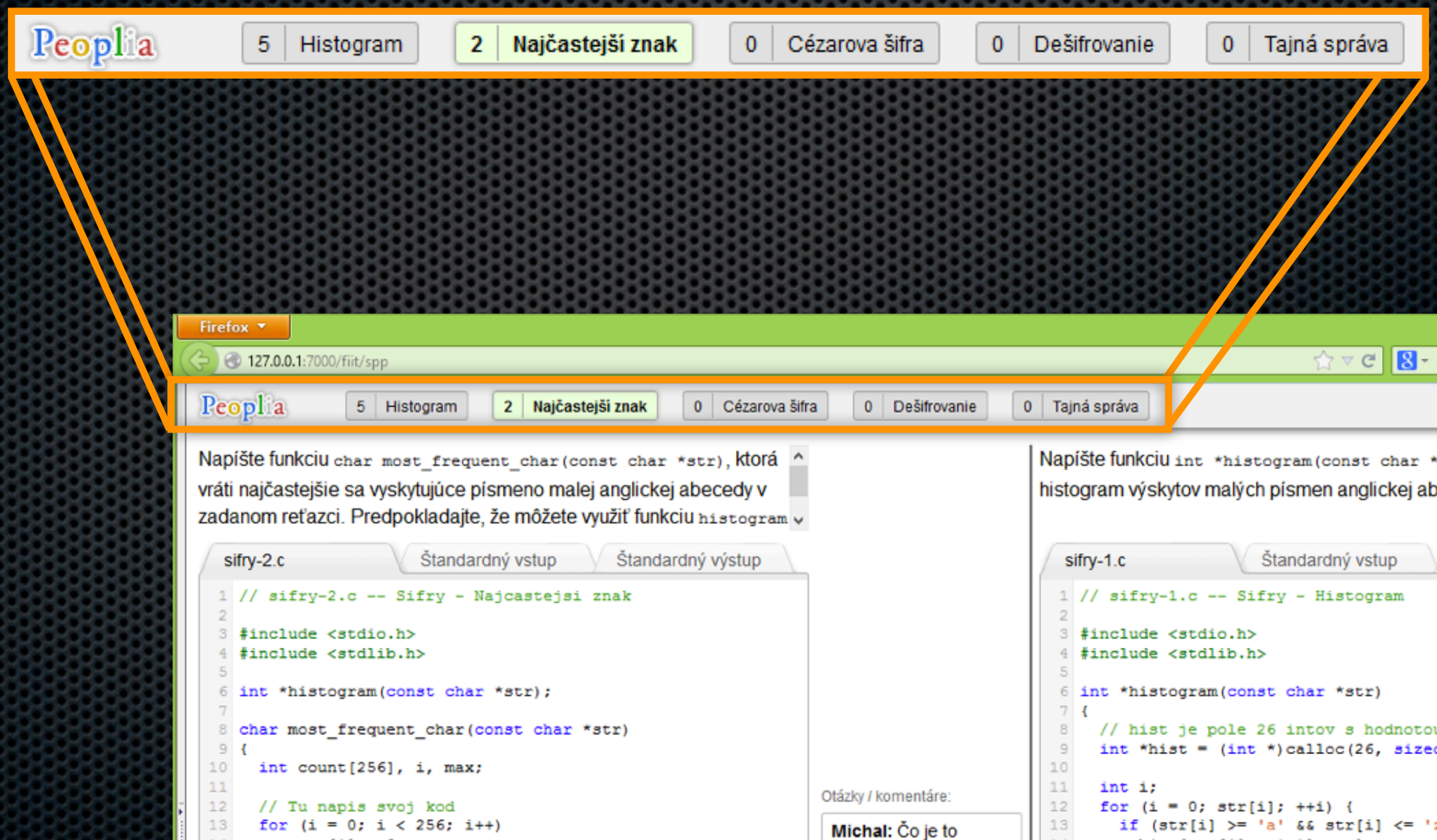
Väčšina študentov pracuje na úlohe 2



Neskôr, väčšina študentov už pracuje na úlohe 4

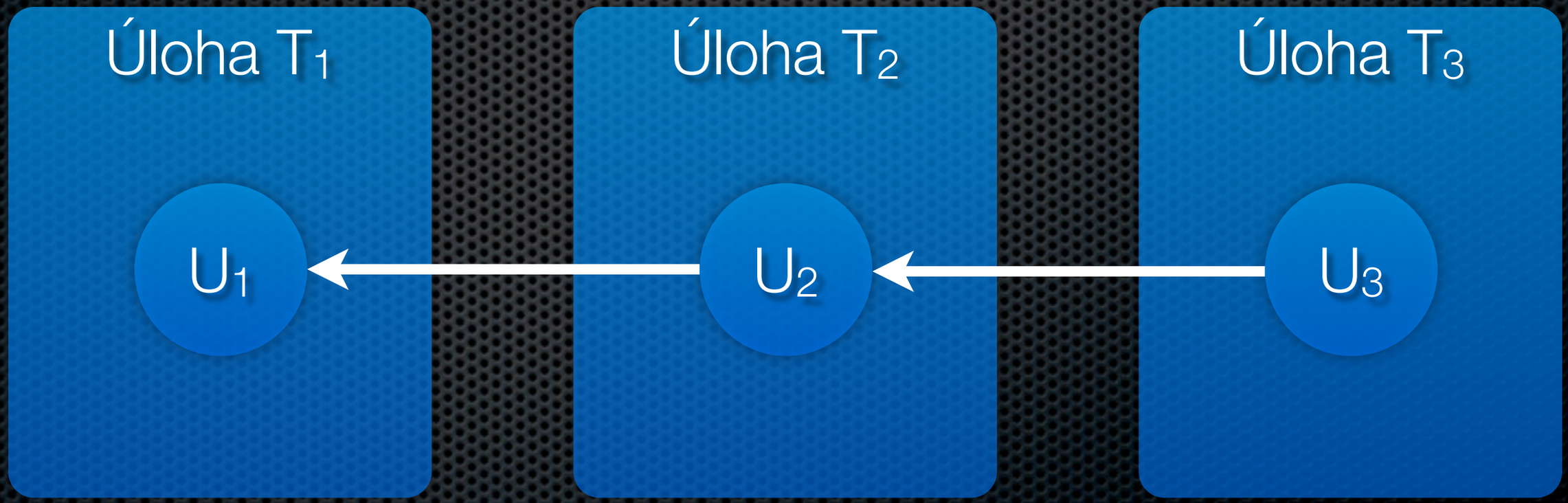


Vizualizácia postupu



Posudzovanie

- **Len vyriešené úlohy** — len ak posudzovateľ už danú úlohu vyriešil



Model používateľa

- Osobnostný model Big Five (extroverzia, ochota, svedomitosť, neuroticizmus, otvorenosť)
- Určujeme schopnosti:
  - Poskytnúť pomoc/radu — *deliver*
  - Prijať pomoc/radu — *receive*

Pravdepodobnosť úspechu pri pridelení

- Raschov model

$$P(i,j) = \frac{e^{deliver_i - receive_j}}{1 + e^{deliver_i - receive_j}}$$

Dotazník na získanie osobnostných charakteristík

Náhodný výber posudzovateľov

Ohodnotenie výberu – ako to dopadlo?

Ohodnotenie pridelení

Posudz.\Použ.	U_1	U_2	U_3	U_4
U_1		1		0
U_2	0		1	1
U_3	1			0
U_4		1		

Maximum Likelihood Estimation pre Raschov model

deliver, receive

Nájdenie vzťahu medzi schopnosťami spolupráce a osobnostnými charakteristikami

Výber posudzovateľa podľa osobnostných charakteristík

Experimenty....

....alebo ako naplniť maticu

1. experiment

- Prvý prototyp

- 172 študentov doma v určenom čase

- Hypotéza: Existuje korelácia medzi osobnostnými črtami a schopnosťou poradiť alebo prijať radu.

Výsledok 1. experimentu

- Študenti nevyužili možnosť komunikovať
- Nezapojili sa do posudzovania, pridelený kód ignorovali
- Problém pri menšom rozlíšení

Zmeny po 1. experimente

- Jednoúlohové rozhranie
- Žiadosti o radu (review requests)
- Druhý prototyp

Jednoúlohové rozhranie

FIIT STU – Procedurálne programovanie 2012/2013 (leto) | Peoplia

https www.peoplia.eu/fiit/pp Reader

Peoplia | 1 Úloha 7-1 | 0 Úloha 7-2 | Bc. Matus Tomlein

Napíšte program, ktorý zistí počet riadkov N v súbore `neusporiadane.txt`, alokuje v pamäti dva bloky po N položiek reálnych čísel a do prvého bloku načíta čísla zo súboru. Predpokladajte, že súbor obsahuje v každom riadku práve jedno reálne číslo. Predpokladajte, že súbor neobsahuje rovnaké číslo viackrát. Potom prekopírujte obsah z prvého bloku do druhého tak, aby čísla v druhom bloku boli vzostupne usporiadané. Nakoniec program zapíše obsah usporiadaného bloku do suboru `usporiadane.txt`. Program negeneruje žiaden výstup na štandardný výstup. Využite ukazovateľovú aritmetiku.
Ukážka súboru `neusporiadane.txt`:

```
4.8
9.26
```

uloha-7-1.c | neusporiadane.txt | Štandardný výstup

```
1  // uloha-7-1.c -- Tyzden 7 - Uloha 1
2
3  #include <stdio.h>
4
5  int main()
6  {
7    // sem napis svoje riesenie
8
9    return 0;
10 }
```

Tvoja otázka:

Ako načítam čísla zo vstupu?

Už netreba

Kamoši sa pýtajú:

Kompilovať | Spustiť program | Odoslať riešenie

Jednoúlohové rozhranie

FIIT STU – Procedurálne programovanie 2012/2013 (leto) | Peoplia

https www.peoplia.eu/fiit/pp Reader

Peoplia | 1 Úloha 8-1 | 0 Úloha 8-2 | 0 Úloha 8-3 | Bc. Michal Tomlein

Napíšte funkciu `int strinsert(char *dst, int len, const char *src, int offset)`, ktorá do reťazca `dst` od pozície `offset` vloží kópiu reťazca `src`. Argument `len` určuje počet znakov vyhradených pre pole `dst` (vrátane ukončovacieho znaku `\0`). Ak nie je možné do vyhradeného miesta reťazec vložiť, funkcia nič nevykoná a vráti 1. Inak (ak je možné do vyhradeného miesta reťazec vložiť), vráti 0. Napr. pre dst:totojeretazec, offset:6, src:druhy volanie strinsert(dst, 50, src, 6) vrati 0 a v reťazci dst bude to

uloha-8-1.c | Štandardný vstup | Štandardný výstup

```c
// uloha-8-1.c -- Tyzden 8 - Uloha 1

#include <stdio.h>

int strinsert(char *dst, int len, const char *src, int offset)
{
  // sem napis svoje riesenie

  return 1;
}

int main()
{
  char buf[100];

  sprintf(buf, "totojeretazec");
  if (strinsert(buf, 100, "druhy", 6))
    printf("Nepodarilo sa vlozit retazec.\n");
   else
    printf("%s", buf);
  return 0;
}
```

Tvoja otázka:

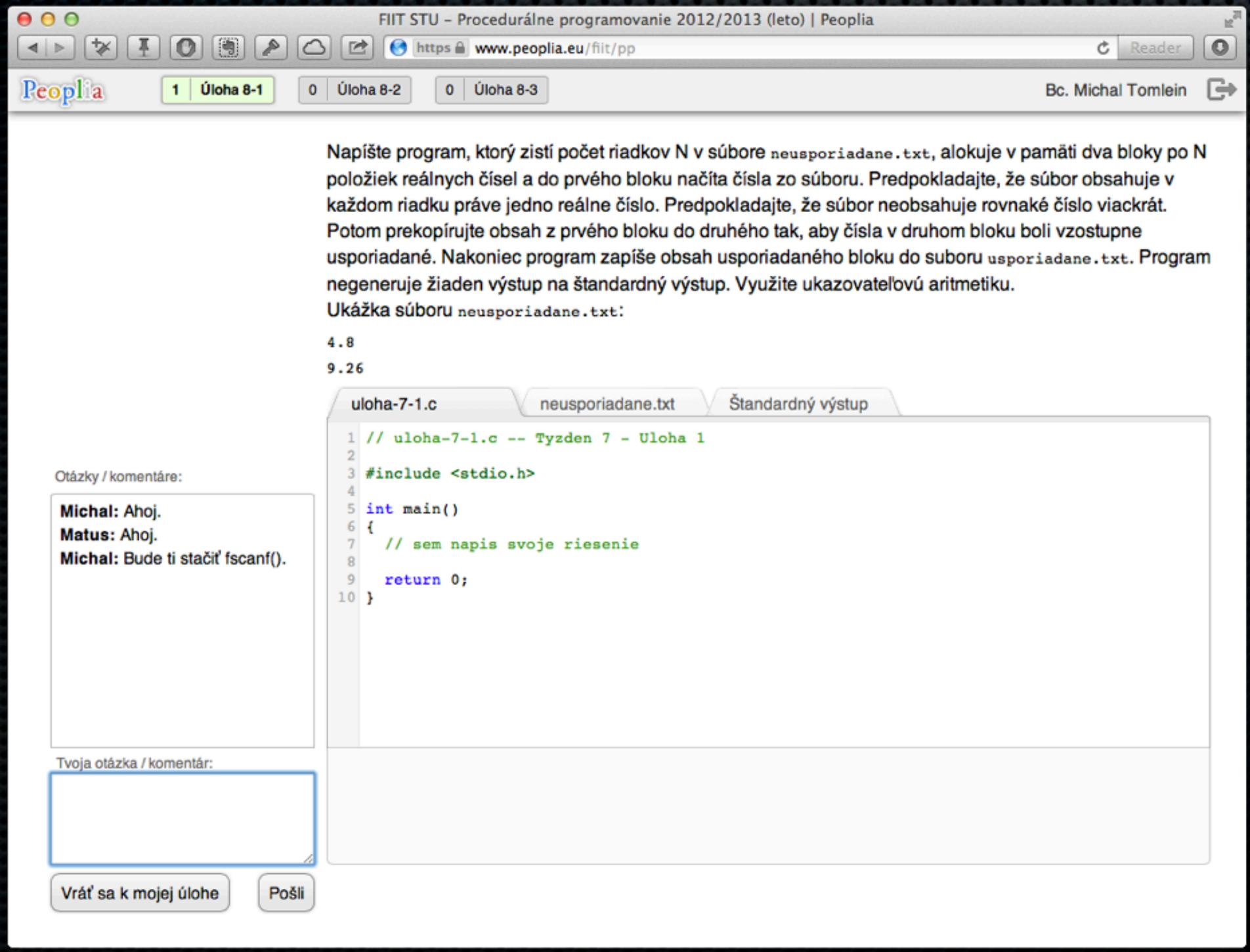
Spýtaj sa kamoša

Kamoši sa pýtajú:

Matus (Úloha 7-1):
Ako načítam čísla zo vstupu?
Klikni a poraď

Kompilovať | Spustiť program | Odoslať riešenie

Jednoúlohové rozhranie



2. experiment

- Jednoúlohové rozhranie + žiadosti o radu

- 79 študentov v učebni (dve cvičenia)

- Hypotéza: Ak študenti budú mať možnosť vedome požiadať o radu a ponúknuť radu, budú aktívnejší.

Výsledok 2. experimentu

- Študenti ignorovali možnosť požiadať o radu
- Počet odoslaných správ: 3
- Dotazník pre 73 z nich:
  - Všimlo si to 86 %

Výsledok 2. experimentu

Prečo sa nezapojili	Počet
nepotreboval	**12**
nevšim. / nevedel	**8**
nebola príležitosť	2
nikto sa neozval	1
zaneprázdnený	1
nebolo koho	1
nikto sa nepýtal	1
bojím sa	1

Výsledok 2. experimentu

Hodnotenie	Počet
fajn	43
dobré, ale	12
nefungovalo	5
netreba	2

„veľmi dobrá funkcia ak by na otázky neodpovedali študenti“

Zmeny po 2. experimente

- Animácia príchodu žiadosti o radu
- Automatické žiadosti
  - po 3 chybných kompiláciách

3. experiment

- Jednoúlohové rozhranie
- Automatické a manuálne žiadosti o radu

- ~ 80 študentov v učebni (dve cvičenia)

- Hypotéza: Študenti budú ochotnejší poradiť ako požiadať o radu.

Výsledok 3. experimentu

- Ešte prebieha, čakám na ďalšie dáta
- Budú menšie zmeny

Výsledná matica

Posudz.\Použ.	U_1	U_2	U_3	U_4
U_1				
U_2				
U_3				
U_4		1		

Zhrnutie