Cvičenie 4

Miroslav Cupák

Obsah

- organizačné záležitosti
- reťazce
- pole
- preťažovanie funkcií (overloading)
- STL
  - kontajnery
  - iterátory
  - algoritmy
- úloha 4

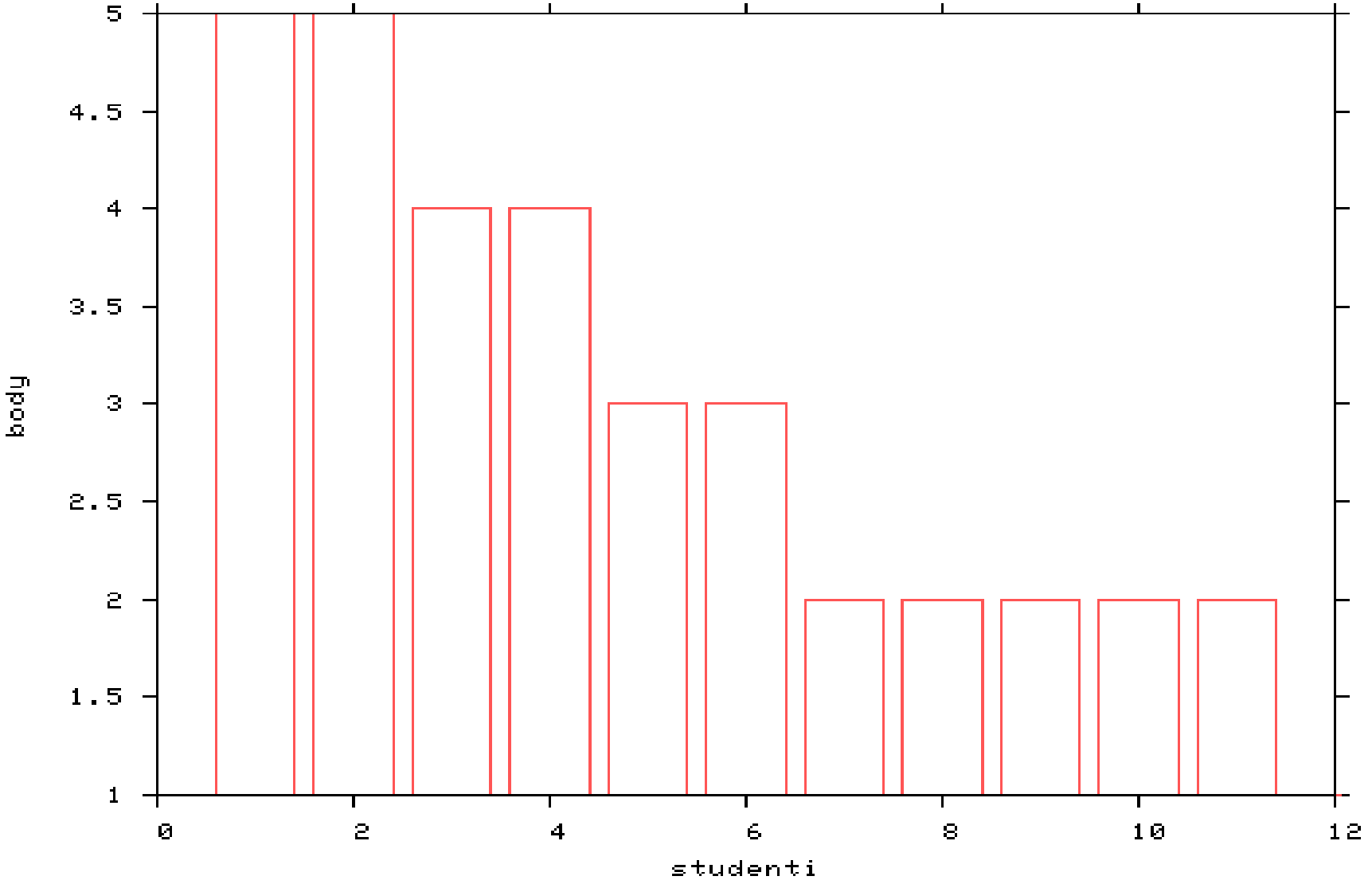
Organizačné záležitosti

- 2. úloha
  - nepovinná, deadline dnes o polnoci
  - nabudúce zhrnutie
- 3. úloha
  - povinná, deadline o 7 dní
- odovzdávanie úloh

  [PB161][cvicXX][login][odevzdání|oprava|dotaz]
  - upozornite na príp. bonusové rozšírenie

Organizačné záležitosti

- aktuálna štatistika (priemer: 2.92b):



Reťazce

- v štýle C
  - pole znakov zakončené “\0”
    (preto pri deklarácii nutné uviesť o 1 znak viac,
    ako je dĺžka reťazca)
  - manipulácia v C++:

```
#include <cstring>
        - strcpy(a,b), strlen(a)...
#include <cctype>
        - toupper(), tolower(), isalpha(), isdigit()...
```

Reťazce

- cvičenie
  - kde sú chyby?

```
char a[ ] = ”retazec” ;
const char * b = ”retazec”;
char c[15];
char d[8] = ”retazec”;
char e[7] = ”retazec”;
c = ”retazec”;
strcpy (c, ”ahoj”);
```

Reťazce

- cvičenie
  - kde sú chyby?

```
char a[ ] = ”retazec” ;
const char * b = ”retazec”;
char c[15];
char d[8] = ”retazec”;
char e[7] = ”retazec”;                    // chyba!
c = ”retazec”;                            // chyba!
strcpy (c, ”ahoj”);
```

Reťazce

- v štýle C++
  - trieda *string*
  - hlavičkový súbor <*string*>
  - http://www.cppreference.com/cppstring/index.html
  - objekt, nie pole znakov
  - je ich možné:
    - dynamicky zväčšovať/zmenšovať
    - lexikograficky porovnávať bežnými operátormi
    - reťaziť
    - konvertovať do štýlu C a späť
    - ...

Reťazce

- v štýle C++
  - operácie (operátory):
    - priradenie (=)
    - test na rovnosť (==) a nerovnosť (!=)
    - lexikografické porovnávanie (<, >, <=, >=)
    - vstup/výstup (>>, <<)
    - zreťazenie (+)
    - priamy prístup k znaku ([], at())
      - at() vhodnejšie, využíva výnimky
- demo1.cc

Reťazce

- metódy pre prácu s reťazcami
  - append(), clear(), empty(), erase(), find(), find_first_of(), replace()...

string s = "Ahoj jak se mas"; s.append(5, '#');
  - “Ahoj jak se mas#####“

s = "Ahoj jak se mas"; s.erase(3,3);
  - “Ahoak se mas“

s = "Ahoj jak se mas"; s.replace(0, 4, "Zdarec,");
  - “Zdarec, jak se mas“

s = "Ahoj jak se mas"; s.find("jak")
  - 5

s.clear();
  - “”

Pole

- jednorozmerné
  - jasné - deklarácia v tvare

    typ názov[rozsah] (napr. int a[5])
- viacrozmerné
  - ak je definované staticky, rozmery musia byť konštantné, ale nemusia to byť nutne makrá

Pole

- príklad
  - pole v štýle C

```
# define X 10
# define Y 5
char pole [X][Y];
```

  - pole v štýle C++

```
const int x = 10;
const int y = 5;
char pole3 [x][y];
```

Pole

- príklad
  - pole v štýle C

```
# define X 10
# define Y 5
char pole [X][Y];
```

  - pole v štýle C++

```
const int x = 10;
const int y = 5;
char pole3 [x][y];
```

Preťažovanie funkcií (overloading)

- možnosť deklarovať niekoľko rôznych funkcií s rovnakým menom
- musia sa líšiť počtom alebo typom parametrov
- na rozlíšenie nestačí:
  - rozdielny typ návratovej hodnoty
  - rozdiel v konštantnosti parametru
  - rozdiel medzi predaním odkazom a hodnotou
- zavolá sa funkcia, ktorá najlepšie “pasuje”

- demo2.cc

STL

- veľmi silný nástroj
- obsahuje
  - šablony kontajnerov, iterátorov
  - algoritmy
  - alokátory, komparátory a iné pomocné triedy
- výhody
  - urýchľuje prácu
  - štandardizovanosť
  - prenositeľnosť
  - efektívne triedenie a vyhľadávanie
- http://www.sgi.com/tech/stl/

Kontajner

- dátová štruktúra, do ktorej môžme ukladať a z ktorej môžme čítať dáta
- parametrizovaný
- môžem definovať pravidlá pre čítanie

Iterátor

- inteligentný ukazateľ na prvky kontajneru, objekt
- ako šikovný index do poľa, na ktorom definujem metódy pre pohyb (dopredu, dozadu, obojstranne, náhodne...)

STL kontajnery

- vector
  - <vector>
  - zovšeobecnené pole
  - efektívne vkladanie nakoniec a náhodný prístup
  - neefektívne vkladanie dostredu
  - metódy
    - náhodný prístup (*at()*), referencia na prvý/posledný prvok (*first(), back()*), vloženie (*insert()*), zmazanie (clear*()*), pripojenie prvku na koniec (*push_back()*), odstránenie posledného prvku (*pop_back()*), zmazanie niektorých prvkov (*erase()*), test prázdnosti (*empty()*)

STL kontajnery

- vector – práca s iterátormi
  - získanie iterátoru na prvý (*begin()*) a za posledný (*end()*) prvok
  - získanie reverzných iterátorov (*rbegin(), rend()*)

```
for (vector<int>::iterator i = v.begin(); i != v.end(); i++)
        cout << *i << " ";
```

STL kontajnery

- list
  - obojsmerný zoznam
  - metódy podobné ako u vektoru, obojsmerný iterátor
- queue
  - fronta
  - *front(), back(), pop(), push(), empty(), size(), ==, <, ...*
- deque
  - dvojstranná fronta
- priority_queue
  - fronta so zoradenými prvkami
- stack
  - zásobník (LIFO)
  - *top(), push(), pop(), empty(), size(), ==, <*

STL kontajnery

- set
  - množina
  - *size(), empty(), insert(), clear(), find(), count(), erase()*
- multiset
  - množina s opakujúcimi sa prvkami
- map
  - mapa, asociatívne pole, ktoré udržiava hodnoty zotriedené podľa kľúča
  - *map<typ kľúča, typ dát, komparátor, alokátor>*

```
map<string, int> sez;
sez.insert(pair<string, int>( Petr , 100));
```

Generické algoritmy

- až na občasné výnimky aplikovateľné na ľubovoľné kontajnery
- for_each(), find(), find_if(), count(), equal(), copy(), swap(), replace(), replace_if(), fill(), generate(), unique(), reverse(), random_shuffle(), sort(), stable_sort(), partial_sort(), merge(), set_operace(), …

- demo3.cc
- demo4.cc

Úloha 4

- http://www.fi.muni.cz/usr/jkucera/pb161/pamatovak.htm
- program testujúci pamäť na slová
- databáza slov daná (hlavičkový súbor)

- povinná
- deadline o 14 dní

Dotazy?