Tema de casă 1 - Data Analysis On The Fly

Responsabili: Carmen Popa [mailto:carmengpopa13@gmail.com] Valentin Ioniță [mailto:valentin.ionita1201@stud.acs.upb.ro]

Termen de predare: 05.11.2018, ora 23:55

Update

19.10.2018 - Corectare checker (again)

25.10.2018 - Updatare checker de coding style; informatii despre checker

26.10.2018 - Updatare checker

30.10.2018 - Modificare checker + adăugare pe vmchecker

Obiectivele temei

- să utilizeze funcțiile din limbajul C de citire și scriere a datelor
- să utilizeze funcțiile matematice din C
- să utilizeze structuri condiţionale si repetitive din C
- să utilizeze tool-ul de automatizări Make pentru a compila automat programele scrise

Limitările temei

- tema va fi realizată doar cu materia prezentată la curs până în săptămâna a patra inclusiv
- tema se va rezolva fără variabile globale

Ce învăț din această tema?

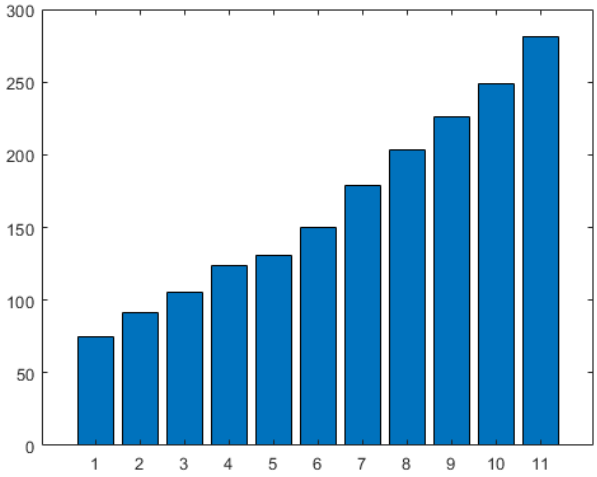
- devin familiar cu reprezentarea datelor într-un mod prietenos pentru cei ce le citesc (ex: grafic, pie chart, histogramă etc)
- învăț cum se lucrează cu date numeroase și cum se calculează statistici pe baza acestora
- descopăr un domeniu nou, Internet of Things, și ce aplicații poate avea acesta

Finalitățile temei

- abordarea unei problemei mai complicate și divizarea acesteia în subprobleme ce pot fi rezolvate individual
- scrierea de cod ușor de citit și de întreținut
- modularizarea codului în subprograme

Problema 1 - Histograma

O histogramă (bar graph) este un grafic care reprezintă datele sub forma de bare verticale/orizontale (dreptunghiuri). Într-o histogramă, pe una dintre axe sunt reprezentate denumirile datelor (de exemplu, Ianuarie, Februarie etc.), iar pe cealaltă axa sunt reprezentate valorile frecvențelor corespunzătoare datelor (numărul de intrări care corespund lunii Ianuarie, lunii Februarie etc.).



Un student vrea să țină evidența serialelor la care s-a uitat în sesiune și dorește să facă un program care să îi afișeze o histogramă cu numărul de episoade pentru fiecare serial. Acesta ia în calcul maxim 10 seriale, codificate astfel:

```
Black Mirror - 0
Doctor Who - 1
Doctor House - 2
Rick & Morty - 3
Breaking Bad - 4
Mr. Robot - 5
Game of Thrones - 6
Grey’s Anatomy - 7
Fullmetal Alchemist: Brotherhood - 8
Vikings - 9
```

Datele de intrare se citesc de la stdin. Mai întai va fi citit un numar natural n care arată numărul de seriale ce vor fi luate în calcul pentru fiecare test în parte. Apoi urmează n intrări de forma *[codificare serial] [număr episoade văzute] [număr total de episoade]*, care corespund serialelor menționate mai sus. Histograma va conține 2 caractere:

```
'*'  pentru a marca umplerea unui nivel
'.'  pentru fundal
```

Fiecare rând al histogramei conține 10 niveluri, ceea ce înseamna că datele vor fi scalate și apoi rotunjite: dacă un serial are 30 de episoade, din care au fost văzute 10, se face regula de trei simplă pentru a scala la 10 și astfel se obține (10 / 30) * 10 = 3.33 care e rotunjit la 3 (adică histograma va conține 3 steluțe și restul de 7 puncte). Similar, valorile 3.70 sau 3.50, de exemplu, se vor rotunji la 4. În cadrul datelor de intrare, codificările pot apărea în orice ordine, dar histograma finală va conține serialele în ordine crescătoare după codificarea acestora.

Exemplu:

Studentul a numărat că a văzut 22/55 episoade din Doctor Who, 40/50 din Game of Thrones, 16/19 Breaking Bad și 10/32 din Doctor House. Un exemplu de date de intrare ar arăta astfel:

```
4
1 22 55
6 40 50
4 16 19
2 10 32
```

Histograma realizată de programul scris de student arată astfel (numerele reprezintă codificările serialelor):

```
1 * * * * . . . . . .
2 * * * . . . . . . .
4 * * * * * * * * . .
6 * * * * * * * * . .
```

Dar realizarea histogramei pe orizontală a fost prea simpla. Acum, studentul este în căutarea unei noi provocări până apare următorul sezon, asa că decide să afișeze histograma vertical. Aceasta va avea 11 rânduri (10 pentru nivelul episoadelor și ultimul, cel de jos, pentru codificarea serialului). Caracterele de pe un rând sunt separate printr-un singur spațiu. Histograma va fi afișată pe ecran (stdout) și ar arata astfel:

```
. . . .
. . . .
. . * *
. . * *
. . * *
. . * *
* . * *
```

```
* * * *
* * * *
* * * *
1 2 4 6
```

Problema 2 - Vampirul componetelor electronice

Un fotorezistor este o componentă electronică care își modifică valoarea în funcție de cantitatea de lumină primită. Acesta are rezistentă mare la întuneric și mai scăzută la lumină. Cu cât valoarea este mai mică, cu atât este mai întunecată camera.

Odată mutat la cămin, studentul observă ca noaptea îi dispare mâncarea din frigider. Acesta cumpără o plăcuță Arduino, niște fotorezistențe și cabluri și scrie un programel care să măsoare cantitatea de lumină primita de fotorezistențe în timpul nopții. Astfel, își va putea da seama dacă cineva îi fură mâncarea cu adevărat.

Plăcuța Arduino pe care a cumpărat-o nu are suficientă memorie încât să rețină toate valorile citite de fotorezistențe, asa ca programul efectuează calculele pe măsură ce primește valorile. Pe baza inputului dat, voi calculați media aritmetică, geometrică, armonică și media pătratică, dar și abaterea standard a datelor, având în vedere formulele:

$$m_{aritmetica} = \frac{p_1 + p_2 + \ldots + p_n}{n}$$

$$m_{geometrica} = \sqrt[n]{p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_n}$$

$$m_{armonica} = \frac{n}{\frac{1}{p_1} + \frac{1}{p_2} + \ldots + \frac{1}{p_n}}$$

$$m_{patratica} = \sqrt{\frac{p_1^2 + p_2^2 + \ldots + p_n^2}{n}}$$

$$stdev = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(p_k - m_{aritmetica})^2}$$

Aceste 5 valori se vor afișa pe ecran cu exact 4 zecimale, fiecare pe câte un rând separat. Se consideră că valorile citite nu vor fi niciodată fix 0, deci nu vă faceți griji de împărțirea la 0. Mai mult, media geometrică nu se poate calcula dacă un termen al seriei este negativ, caz în care la ieșire se va afișa caracterul '-'.

De asemenea, studentul dorește să știe acuratețea cu care fotorezistențele citesc datele. Astfel, se va afla minimul și maximul din valorile citite și de câte ori apare fiecare, dar și cea mai lungă secvență crescătoare (nu neapărat strict crescătoare), pentru a vedea dacă lumina a fost cu adevărat aprinsă pentru câteva secunde, sau dacă a fost doar o eroare de citire.

Aceste rezultate se vor afișa pe ecran, pe trei rânduri diferite, separate printr-un spațiu, ca în exemplul de mai jos. De remarcat faptul că și minim si maxim sunt numerele reale, deci se vor afișa cu exact 4 zecimale:

```
minim nr_minim
maxim nr_maxim
lungime
```

Exemplu:

Date de intrare:

```
8
-15.2
27.5
-121.2
112.3
-121.2
1000
1000
82.13
```

Date de ieșire:

```
245.5412
-
-350.1560
506.1772
442.6340
-121.2000 2
1000.0000 2
3
```

Explicatie:

Pentru datele de intrare, prima valoare reprezintă N, adică numărul de valori înregistrate de fotorezistență, iar următoarele N numere reprezintă valorile efective înregistrate. Pentru datele de ieșire, primele 5 rânduri sunt pentru: media aritmetică, media geometrică, media armonică, media pătratică și abaterea standard. Următoarele două rânduri sunt pentru minim și maxim și numărul de apariții, iar ultimul rând arată lungimea celei mai lungi secvențe crescătoare.

Checkerul verifică datele cu o precizie de exact 4 zecimale. De exemplu, afișarea numărului 1 ca 1.000 sau 1.00000 (cu 3 sau cu 5 zecimale) nu va fi punctată, deoarece checkerul este construit să acorde punctajul doar pentru numere de forma 1.0000 (4 zecimale).

Ajutați studentul să scrie acest program pentru a vedea dacă cineva deschide frigiderul său în timpul nopții. În viitor, va adăuga și un buzzer care sa facă zgomot atunci când frigiderul e deschis și o cameră de filmat pentru a îl prinde pe hoț.

Precizările legate de implementare

Regulamentul general se găsește aici.

- Implementarea se va face in limbajul C, iar tema va fi compilată şi testată DOAR într-un mediu LINUX. Nerespectarea acestor reguli aduce un punctaj NUL.
- Fișierul de implementare a temei se va numi histograma.c pentru Problema 1 și statistici.c pentru Problema 2, iar în cadrul testării vor fi două executabile diferite verificate.
- Deși programul vostru va trebui să citească direct de la tastatura și să afișeze pe ecran (folosind, de exemplu, scanf și printf), puteți sa citiți datele și să le scrieți în fișiere, folosind redirectările din consolă, fără sa modificați programul. Pentru Linux, comanda este:

```
./tema1 < in1 > out1
```

Testarea si trimiterea temei

- Fișierele temei trebuie OBLIGATORIU împachetate într-o arhiva de tip '.zip', cu numele 'Grupa_NumePrenume_Tema1.zip'. Această arhivă va conține:
    1. Codul sursă al programului vostru (două fișiere .c si, eventual, .h)
    2. Un fişier `Makefile` care să conţină regulile `build` şi `clean`. Regula `build` va compila programele în două executabile cu numele **`hist`** și **`stats`**. Regula `clean` va şterge executabilele şi eventual toate binarele intermediare (fişiere obiect) generate de voi.
    3. Un fişier `README` care să conţină explicaţii privitoare la modul de rezolvare.
    4. Arhiva temei NU va conține fișiere binare.
- Lipsa codului sursă și/sau a Makefile-ului aduce un punctaj NUL.
- Arhiva va fi trimisă pe vmchecker [https://elf.cs.pub.ro/vmchecker/].
- O temă care nu compilează nu va fi punctată.
- O temă care compilează, dar care nu trece niciun test, nu va fi punctată.

Checker

Checkerul poate fi descărcat de aici: tema01_ca2018_checker-v7.zip

Tema va fi verificată automat cu un script. În caz că aveți erori, checkerul va penaliza cu 15p din punctajul total obținut (conform regulamentului). Pentru a sti cum sa preveniți aceste probleme, vă rugam să parcurgeți pagina de coding style de pe ocw.

În cazul în care fișierul cs.py (cel care verifică coding style-ul) nu e executabil, atunci va apărea urmatoarea eroare (poza de mai jos). Acest lucru se rezolvă prin rularea comenzii

```
chmod u+x cs.py
```

în folderul cu fișierul cs.py.

```
Task 2 score: 80/80

==========> Check Coding Style <==========
./checker.sh: line 113: [: -gt: unary operator expected
Coding style OK. Punctajul final se poate modifica la corectare.
```

- În lipsa Makefile-ului, sau dacă sursele au erori de compilare, checker-ul nu va putea rula și nu va verifica tema. Punctajul aferent este 0 în acest caz. Veți vedea mesajul de mai jos:

```
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
|                                                     |
| Dalek 1: Align and Advance!                         |
| Dalek 2: Advance and Attack!                        |
| Dalek 3: Attack and Destroy!                        |
| All Daleks: Destroy and Rejoice!                    |
| Dalek ship: Makefile not found. Process interrupted.|
|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _|

==========> Total: 0/100 <==========
```

- Fișierul README valorează 5 puncte. Acestea nu se primesc daca fișierul lipsește sau e gol. Mesajele afișate sunt cele de mai jos:

```
==========> Check README <==========
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
|                                                        |
| The Doctor: Identify me! Access your files. Who am I?  |
| Dalek: Error of accessing files. README not found.     |
|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _|


Final score: 95/100
```

```
==========> Check README <==========
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
|                                                                    |
| The Doctor: I'm part of you. My mind is in your mind.              |
| Rusty, the Dalek: I see your mind, Doctor. I see your universe.    |
|     No, wait... the README is empty. I cannot see anything.        |
|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _|

Final score: 95/100
```

- Odată cu rularea testelor este verificat și coding style-ul. Dacă apar erori de coding style, checkerul va scădea 15 puncte din punctajul obținut. Mesajul arătat este găsit mai jos. De asemenea, checkerul menționează unde sunt erorile pentru a fi corectate.

```
Done processing ./histograma.c
Done processing ./statistici.c
Total errors found: 3
3 C errors found.

 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
|                                                 |
| The coding style standards are not fully met.   |
| You will lose points based on this.             |
| Dalek Prime Minister: Does it surprise you to   |
| know the Daleks have a concept of beauty?       |
|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _|
```

- Sfat:

  ```
  cat -e nume_fișier
  ```

  afișează conținutul fișierului și caracterele albe (spațiu, new line, tab etc). Este util pentru a vedea de ce nu trec anumite teste.

- Makefile:

```
build:
	gcc -std==c99 -Wall -Wextra nume_sursa_1 -o nume_executabil_1 -lm
	gcc -std==c99 -Wall -Wextra nume_sursa_2 -o nume_executabil_2 -lm
clean:
	rm -rf nume_executabil_1 nume_executabil_2
```

Regula de build din makefile conține niște argumente noi, care permit declararea variabilelor în cadrul for-urilor. Ca să vă asigurați că tema va fi rulată corect pe vmchecker, e recomandat să modificați makefile-ul ca mai sus.

Copierea parţială sau totală a unei rezolvări din altă sursă va atrage după sine anularea punctajelor pentru toate temele de casă, atât pentru cel care a copiat, cât şi pentru sursa acestuia.

programare/teme_2018/tema1_2018_ca.txt · Last modified: 2018/10/30 20:49 by teodora.serbanescu