

STRUCTURI DE DATE

TEMA 1

SIMULARE HARD DISK

Liste. Stive. Cozi

Cîrstei Răzvan

Oprea Bogdan

Facultatea de Automatică și Calculatoare

1 Contextul temei

Ca ingineri ai unei companii de calculatoare (voi ați vrut...) vă confrunțați cu următoarea situație:

Compania tocmai a lansat pe piață un nou tip de hard disk. Imediat după lansare șefii companiei au primit o sesizare că acest produs va avea o durată de funcționare mai mică decât precizează garanția. Pentru a evita o lovitură financiară serioasă, consiliul directorilor a decis retragerea de pe piață a produsului.

Observând că eroarea sesizată nu există în practică, voi ați sugerat păstrarea pe piață a produsului. Consiliul directorilor a amânat retragerea produsului cu 3 săptămâni, însă vi s-a cerut să veniți cu o dovadă clară că eroarea sesizată nu prezintă un pericol. Astfel va trebui să construiți o simulare aproximativă a dispozitivului în C sau C++.

2 Descriere dispozitiv

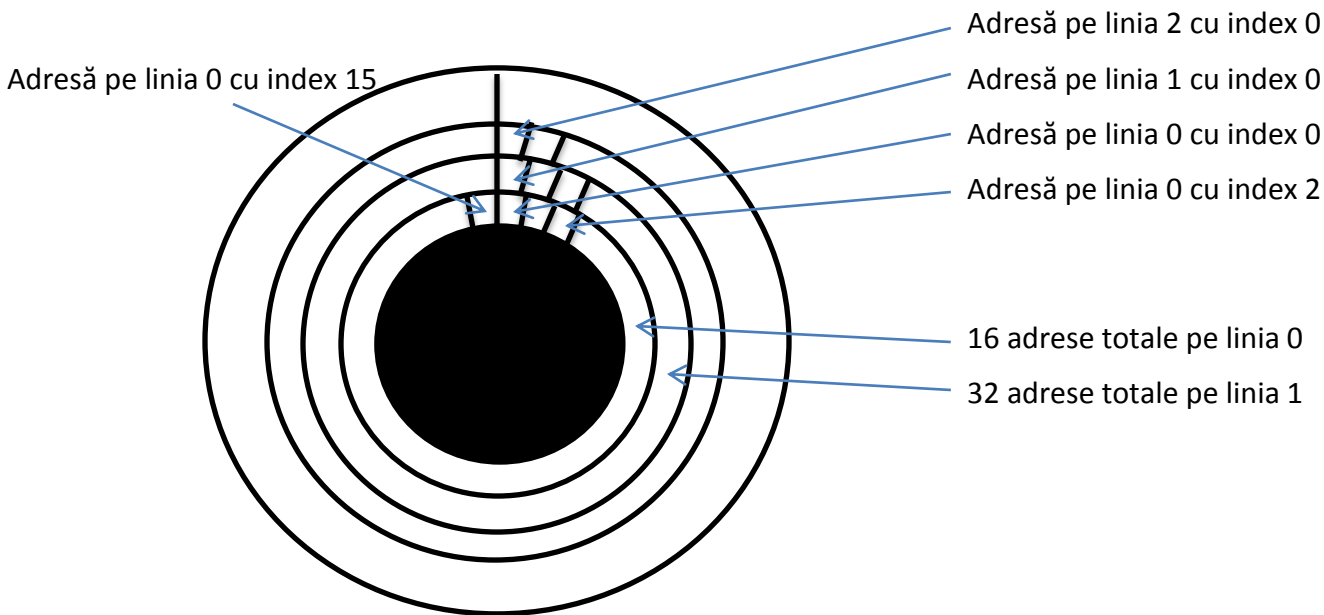
Hard disk-ul este compus dintr-un disk pe care se scrie informația și dintr-un cursor ce execută operații pe disk. Informația este stocată în grupări de 2 octeți în adrese de memorie. Astfel o adresă va conține:

$\{xxxx \mid x \in \{0, 1, 2, \dots, 9, A, B, C, D, E, F\}\}$, deci o adresă va putea stoca **{ 0000, 0001, ..., FFFF }**.

Adresele sunt grupate pe linii astfel:

- **Prima linie va conține 16 adrese** (una lângă alta).
- Fiecare linie dinspre exterior va conține **de 2 ori mai multe adrese decât linia precedentă** din interior.

Atât adresele cât și liniile sunt **numerotate de la 0**. Cursorul se poate deplasa **doar în sensul acelor de ceasornic** și doar de la o adresă la altă adresă vecină, deci **nu se poate face salt peste adrese**. Pentru a schimba linia, **cursorul se poate muta sus sau jos DOAR prin adresele cu index-ul 0** al fiecărei linii și nici aici nu se poate sări peste adrese ci se parcurge tot din aproape în aproape.



Se va lucra pe timpi. Pe fiecare timp cursorul poate să facă una din următoarele operații:

- **Execută** o comandă pe adresa la care se află, sau
- **Se mută** cu o adresă sau
- **Rămâne pe loc** (daca nu are operații de executat)

Toate operațiile aplică o uzură pe acea adresă de memorie (**DAMAGE**) pe care noi trebuie să o reținem deoarece reprezintă o parte crucială a demonstrației noastre.

3 Ce vrem să demonstrăm:

Sesizarea: “Deoarece pentru a ajunge la o adresă aflată la finalul unei linii trebuie să trecem prin toate cele de la început, uzura pe primul sfert din disk va fi mai mare decât pe restul disk-ului și va ceda în acea zonă.”

Este evident de ce afirmația de mai sus este eronată. Ce trebuie făcut acum este să simulăm comenzi pe dispozitiv după care, împărțind disk-ul în 4 zone, să calculăm media uzurii pe fiecare zonă. După multe comenzi mediile vor fi apropiate.



Roșu – prima zonă, începe de la adresa 0 a fiecărei linii (include adresa 0)

Galben – zona 2 ; Verde – zona 3 ; Albastru – zona 4.

Notă: Toate zonele conțin un număr egal de adrese de memorie

4 Comenzi

Se trimite o comandă urmată de un timp de pauză ce reprezintă timpul până se va citi următoarea comandă. Cursorul trebuie să se mute timp cu timp către adresa de memorie precizată în comandă, după care să execute comanda. Cursorul va fi inițial pe linia 0, index 0.

Comenzi de bază:

- **Read data:**

::r [linie] [index] , unde linie și index sunt de tip “int”

Se vor afișa datele stocate la adresa de memorie de pe linia și indexul precizate (newline)

- **Write data:**

::w [linie] [index] [data] , unde linie și index sunt de tip “int”, iar data este un string de 4 caractere

Se va scrie data în adresa de memorie de pe linia și indexul precizate

- **Read damage:**

::d [linie] [index] , unde linie și index sunt de tip “int”

Se va afișa DAMAGE-ul adresei de memorie de pe linia și indexul precizate (newline)

- **End:**

::e

Se vor neglija uzurile produse de și după această comandă. Se vor afișa, în ordine, separate prin spațiu, mediile celor 4 zone de memorie: CU 2 ZECIMALE, APROXIMATE PRIN LIPSA. Programul se va termina IMEDIAT după această comandă (**cursorul va rămâne pe linia următoare**).

Comenzi BONUS:

- **Multiple Read data:**

::mr [linie] [index] [no] , unde linie, index și no sunt de tip “int”

Se va executa “Read data” pe ‘no’ adrese de memorie începând cu cea de la linia și indexul precizate. Datele citite se vor afișa câte una pe un rând. Dacă se termină linia, se continuă cu linia superioară.

- **Multiple Write data:**

::mw [linie] [index] [data] [data] [data] [data] ... [data] . , unde linie și index sunt de tip “int”, iar data este un string de 4 caractere; comanda se termină cu “.” Dacă se termină linia, se continuă cu începutul liniei superioare. Se va scrie de la adresa de memorie de pe linia și indexul precizate mai departe datele precizate.

DAMAGE

Read data – 5 Damage pe acea adresă de memorie după ce a citit datele

Write data – 30 Damage pe acea adresă de memorie după ce a scris datele

Read damage – 2 Damage pe acea adresă de memorie după ce a citit damage-ul

La finalul fiecărui timp cursorul aplică **1 Damage** pe adresa de memorie pe care se află, indiferent dacă s-a mutat pe adresă, a rămas pe adresă sau a executat o comandă la acea adresă.

5 Cerințe și Punctaj

- Se construiește disk-ul. Modul de primire al comenzilor și numărul de linii se citesc din fișierul de intrare de pe prima linie. Toate adresele de memorie au inițial stocat **“0000”**.
- (QUEUE)** Pe prima linie a fișierului de intrare se va afla cifra 1 urmată de un spațiu și numărul de linii conținute de disk. Următoarele linii vor conține o comandă, iar pe rândul următor timpul de așteptare până când se citește următoarea comandă. Fișierul se termină cu comanda **::e** .
Să se implementeze un algoritm (C / C++) ce simulează acest dispozitiv pe care se pot executa comenzile de bază. Comenzile vor fi **executate în stil coadă** (first in – first out), astfel se vor executa în ordinea citirii. Se va construi fișierul de output.
3p comenzile de bază + 1p comanda end = 4p
- (STACK)** Pe prima linie a fișierului de intrare se va afla cifra 2 urmată de un spațiu și numărul de linii conținute de disk. Următoarele linii vor conține o comandă, iar pe rândul următor timpul de așteptare până când se citește următoarea comandă. Fișierul se termină cu comanda **::e** .
Să se implementeze un algoritm (C / C++) ce simulează acest dispozitiv pe care se pot executa comenzile de bază. Comenzile vor fi **executate în stil stivă** (last in – first out), astfel comenzile citite ultimele vor avea prioritate. Se va construi fișierul de output.
4p comenzile de baza + 1p comanda end = 5p
- Coding Style** **1p**
- BONUS 1** – pt cazul QUEUE să se implementeze și comenzile de bonus. **1p**
- BONUS 2** – pt cazul STACK să se implementeze și comenzile de bonus. Fiecare comanda de bonus este văzută ca o serie de comenzi individuale, deci, dacă se citește o comandă nouă în timpul executării unei comenzi de bonus, se suspendă executarea celei de bonus și se trece la comanda nouă. Ulterior se poate relua comanda suspendată când redevine prioritară. (ex. la (6) hints) **1p**

FISIERELE SE CITESC CA ARGUMENTE: nume_fisier_de_intrare urmat de nume_fisier_de_iesire

Deoarece o implementare profesionistă ar fi cu alocare dinamică și prezentarea soluției trebuie să fie cât mai profesionistă, **nu se vor folosi vectori / alocare statică** (depunțare consistentă).

IMPORTANT: **NU se vor folosi biblioteci cu liste, cozi, stive deja implementate!**
Se vor depuncta pierderile de memorie (1p).

Temele trebuie încărcate pe vmchecker. Arhiva temei se va numi:

Nume_Toate_Prenumele_GrupaSeria_Tema1_SD.zip (ex: Cirstei_Razvan_313CC_Tema1_SD.zip) și va conține:

- **Cod sursă**
- **Makefile** – conține build (compilare și crearea binarului myHDD) și clean (va șterge orice fișier creat în urma comenzii build). Lipsa fișierului Makefile va aduce o depunțare de 1p.
- **README** – conține detalii de implementare (scrierea sumară sau absența acestui fișier poate aduce o depunțare de până la 2p).

Temele se vor puncta doar pe vmchecker.

6 Hints

- Nu uitați: cursorul va fi inițial pe linia 0, index 0.
- **BONUS 2 - Intreruperea unei comenzi “multi-”, exemplu:**
Suntem pe linia 0 adresa 0 cu timp de așteptare 0. Citim comanda “::mr 0 0 8” urmată de 3 pe rândul următor. Ce va face cursorul: este în zona corectă deci citește și afișează valoarea de la 0 0. Timp de așteptare rămas = 2. Se mută pe linia 0 adresa 1. Timp de așteptare rămas = 1. Citește și afișează valoarea de la 0 1. Timp de așteptare rămas = 0, deci citește următoarea comanda (să zicem ::r 1 5). Fiind cazul STACK, comanda citită (::r 1 5) are prioritate mai mare, deci o vom executa pe aceasta acum. Rămâne să citim restul de 6 elemente de la comanda “mr” imediat după executarea comenzii “r” dacă avem timp de așteptare suficient și nu apare altă comandă prioritară.
- **NU se va anticipa!** Dacă o comandă nu este prioritară **NU** se execută chiar dacă ne aflăm în zona de memorie convenabilă.
- **ORDINEA ESTE:** (execuție comandă prioritară dacă suntem în adresa corespunzătoare) sau (se mută cursorul), apoi se aplica Damage 1 de la cursor (indiferent dacă se execută sau nu comenzi).
- Rețineți că liniile sunt ciclice.
- Nu folosiți vectori. Scopul temei este de a vă familiariza cu liste, cozi, stive. Pentru a vă puncta tema trebuie să le folosiți.
- La **BONUS**, în cazul în care sunteți la finalul unei linii și trebuie să continuați scrierea sau citirea pe linia superioară, de la începutul acesteia: se garantează că există o linie superioară. (superioară = cu un rând mai în exterior).
- **Eliberați memoria!** Există riscul ca vmchecker să nu aibă destulă memorie să vă suporte programul dacă nu o folosiți bine. Pentru a vă testa programul ca să fiți siguri că nu are pierderi de memorie în Linux puteți folosi un utilitar numit Valgrind (instrucțiuni de folosire se găsesc mai jos).
- Limita superioară a liniilor disk-ului este 10.
- Nu uitați să afișați end-ul cu 2 zecimale rotunjit inferior. **NU** rotunjiți prin adaos.
- Se garantează că nu se dau valori în afara liniilor sau adreselor pentru comenzi. Se garantează că toate comenzile sunt din cele precizate și după fiecare comandă (exceptând comanda end) se află un timp de așteptare.
- Pentru un coding style optim puteți intra [aici](#).
- Nu uitați că tema este individuală. Nu sunt permise copieri de la colegi / de pe internet.

7 Valgrind

Valgrind este un instrument de debugging pentru Linux folosit pentru depistarea greșelilor de utilizare a memoriei: memory leakuri, pointeri invalizi, folosirea unor variabile neinițializate, freeuri invalide. Pentru a folosi valgrind va trebui să instalați pachetul valgrind.

De exemplu, pe distribuția Ubuntu:

```
sudo aptget install valgrind
```

Pentru a rula programul în Valgrind cu toolul pentru memcheck folosiți:

```
valgrind tool=memcheck ./executabil
```

Pentru afișarea leakurilor detaliat:

```
valgrind tool=memcheck leakcheck=full ./executabil
```

Un articol foarte bun și destul de complet despre valgrind găsiți [aici](#).

8 Precizări

TEMA ESTE INDIVIDUALĂ.

DEADLINE HARD:

05 / 04 / 2015 ora 23:55

9 Teste

data.in	data.out
<pre> 1 5 ::r 2 56 20 ::w 1 31 AA2A 1 ::r 1 31 100 ::d 1 31 1 ::d 1 31 1 ::e </pre>	<pre> 0000 AA2A 58 61 0.20 0.19 0.19 0.75 </pre>
<pre> 2 5 ::r 2 56 20 ::w 1 31 AA2A 1 ::r 1 31 100 ::d 1 31 1 ::d 1 31 1 ::e </pre>	<pre> 0000 0.34 0.24 0.19 0.49 </pre>
<pre> 2 5 ::r 2 56 20 ::w 1 31 AA2A 1 ::r 1 31 200 ::d 1 31 1 ::d 1 31 100 ::e </pre>	<pre> 0000 0000 39 42 0.41 0.38 0.38 1.75 </pre>
<pre> 1 3 ::r 0 0 2 ::w 0 2 AAAA 1 ::r 0 2 10 ::d 0 1 100 ::d 0 2 1 ::r 1 1 200 ::e </pre>	<pre> 0000 AAAA 2 46 0000 12.10 0.28 0.28 0.28 </pre>
<pre> 2 3 ::r 0 0 2 ::w 0 2 AAAA 1 ::r 0 2 10 ::d 0 1 100 ::d 0 2 1 ::r 1 1 200 ::e </pre>	<pre> 0000 0000 2 0000 47 11.25 0.57 0.57 0.57 </pre>

1 8 ::r 7 3 20 ::w 7 2 AAAA 1 ::r 7 2 10000 ::r 7 2 2 ::d 7 2 1 ::w 7 1 1 ::d 7 1 2000 ::d 7 1 2 ::d 7 2 3 ::d 7 2 5 ::e	0000 AAAA AAAA 7998 34 37 8002 8005 8.91 1.0 1.0 1.0
2 8 ::r 7 3 20 ::w 7 2 AAAA 1 ::r 7 2 10000 ::r 7 2 2 ::d 7 2 1 ::w 7 1 1 ::d 7 1 2000 ::d 7 1 2 ::d 7 2 3 ::d 7 2 5 ::e	0000 0000 AAAA 7998 8.86 1.0 1.0 0.96
1 7 ::mw 0 0 AAAA BBBB CCCC DDDD FFFF 2AAA 2BBB . 10 ::d 0 5 50 ::r 0 6 1 ::d 0 6 3 ::mr 0 1 4 90 ::e	33 2BBB 39 BBBB CCCC DDDD FFFF 0.30 0.44 0.1 0.1
2 7 ::mw 0 0 AAAA BBBB CCCC DDDD FFFF 2AAA 2BBB . 10 ::d 0 5 50 ::r 0 6 1 ::d 0 6 3 ::mr 0 1 4 90 ::e	1 2BBB 84 BBBB CCCC DDDD FFFF 0.29 0.46 0.0 0.0

1 5 ::d 0 14 100 ::mw 0 14 AAAA BBBB CCCC DDDD FFFF 2AAA 2BBB . 5 ::mr 0 14 7 30 ::mr 0 14 7 30 ::d 0 14 100 ::mr 0 14 3 2 ::e	1 AAAA BBBB CCCC DDDD FFFF 2AAA 2BBB AAAA BBBB CCCC DDDD FFFF 2AAA 2BBB 135 AAAA BBBB CCCC 2.16 0.32 0.32 1.84
2 5 ::d 0 14 100 ::mw 0 14 AAAA BBBB CCCC DDDD FFFF 2AAA 2BBB . 5 ::mr 0 14 7 30 ::mr 0 14 7 30 ::d 0 14 100 ::mr 0 14 3 2 ::e	1 AAAA BBBB 0000 0000 0000 0000 0000 0000 128 AAAA BBBB 0000 0000 0000 0000 0000 0000 2.33 0.22 0.22 1.65
2 5 ::d 0 14 100 ::mw 0 14 AAAA BBBB CCCC DDDD FFFF 2AAA 2BBB . 5 ::mr 0 14 7 300 ::mr 0 14 7 300 ::d 0 14 100 ::mr 0 14 3 2 ::e	1 AAAA BBBB 0000 0000 0000 0000 0000 AAAA BBBB CCCC DDDD FFFF 2AAA 2BBB 135 AAAA 5.94 0.32 0.32 2.25
1 10 ::d 0 2 1 ::d 0 2 1 ::d 0 2 1 ::e	1 4 7 0.0 0.0 0.0 0.0

InFile	OutFile	ProgData
1 1	1	# Command Readed #
::d 0 1	1	Time: 0 : NextComIn: 8 : LineOn: 0 : AreaOn: 0 : DamageNowHere: 0
8	11	Time: 1 : NextComIn: 7 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 1
::d 0 2	14	# Command will be Executed # at time: 2
3	6.0 1.0 1.0 1.0	Time: 2 : NextComIn: 6 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 4
::d 0 1		Time: 3 : NextComIn: 5 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 5
6		Time: 4 : NextComIn: 4 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 6
::d 0 1		Time: 5 : NextComIn: 3 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 7
3		Time: 6 : NextComIn: 2 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 8
::e		Time: 7 : NextComIn: 1 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 9
		# Command Readed #
		Time: 8 : NextComIn: 3 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 10
		Time: 9 : NextComIn: 2 : LineOn: 0 : AreaOn: 2 : DamageNowHere: 1
		# Command will be Executed # at time: 10
		Time: 10 : NextComIn: 1 : LineOn: 0 : AreaOn: 2 : DamageNowHere: 4
		# Command Readed #
		Time: 11 : NextComIn: 6 : LineOn: 0 : AreaOn: 2 : DamageNowHere: 5
		Time: 12 : NextComIn: 5 : LineOn: 0 : AreaOn: 3 : DamageNowHere: 1
		Time: 13 : NextComIn: 4 : LineOn: 0 : AreaOn: 4 : DamageNowHere: 1
		Time: 14 : NextComIn: 3 : LineOn: 0 : AreaOn: 5 : DamageNowHere: 1
		Time: 15 : NextComIn: 2 : LineOn: 0 : AreaOn: 6 : DamageNowHere: 1
		Time: 16 : NextComIn: 1 : LineOn: 0 : AreaOn: 7 : DamageNowHere: 1
		# Command Readed #
		Time: 17 : NextComIn: 3 : LineOn: 0 : AreaOn: 8 : DamageNowHere: 1
		Time: 18 : NextComIn: 2 : LineOn: 0 : AreaOn: 9 : DamageNowHere: 1
		Time: 19 : NextComIn: 1 : LineOn: 0 : AreaOn: 10 : DamageNowHere: 1
		# Command Readed #
		Time: 21 : NextComIn: END : LineOn: 0 : AreaOn: 12 : DamageNowHere: 1
		Time: 22 : NextComIn: END : LineOn: 0 : AreaOn: 13 : DamageNowHere: 1
		Time: 23 : NextComIn: END : LineOn: 0 : AreaOn: 14 : DamageNowHere: 1
		Time: 24 : NextComIn: END : LineOn: 0 : AreaOn: 15 : DamageNowHere: 1
		Time: 25 : NextComIn: END : LineOn: 0 : AreaOn: 0 : DamageNowHere: 1
		Time: 26 : NextComIn: END : LineOn: 0 : AreaOn: 1 : DamageNowHere: 11
		# Command will be Executed # at time: 27
		Time: 27 : NextComIn: END : LineOn: 0 : AreaOn: 1 : DamageNowHere: 14
		# Command will be Executed # at time: 28
		Time: 28 : NextComIn: END : LineOn: 0 : AreaOn: 1 : DamageNowHere: 17
		Press any key to continue . . .
2 1	1	# Command Readed #
::d 0 1	1	Time: 0 : NextComIn: 8 : LineOn: 0 : AreaOn: 0 : DamageNowHere: 0
8	4.0 1.0 1.0 0.0	Time: 1 : NextComIn: 7 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 1
::d 0 2		# Command will be Executed # at time: 2
3		Time: 2 : NextComIn: 6 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 4
::d 0 1		Time: 3 : NextComIn: 5 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 5
6		Time: 4 : NextComIn: 4 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 6
::d 0 1		Time: 5 : NextComIn: 3 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 7
3		Time: 6 : NextComIn: 2 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 8
::e		Time: 7 : NextComIn: 1 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 9
		# Command Readed #
		Time: 8 : NextComIn: 3 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 10
		Time: 9 : NextComIn: 2 : LineOn: 0 : AreaOn: 2 : DamageNowHere: 1
		# Command will be Executed # at time: 10
		Time: 10 : NextComIn: 1 : LineOn: 0 : AreaOn: 2 : DamageNowHere: 4
		# Command Readed #
		Time: 11 : NextComIn: 6 : LineOn: 0 : AreaOn: 2 : DamageNowHere: 5
		Time: 12 : NextComIn: 5 : LineOn: 0 : AreaOn: 3 : DamageNowHere: 1
		Time: 13 : NextComIn: 4 : LineOn: 0 : AreaOn: 4 : DamageNowHere: 1
		Time: 14 : NextComIn: 3 : LineOn: 0 : AreaOn: 5 : DamageNowHere: 1
		Time: 15 : NextComIn: 2 : LineOn: 0 : AreaOn: 6 : DamageNowHere: 1
		Time: 16 : NextComIn: 1 : LineOn: 0 : AreaOn: 7 : DamageNowHere: 1
		# Command Readed #
		Time: 17 : NextComIn: 3 : LineOn: 0 : AreaOn: 8 : DamageNowHere: 1
		Time: 18 : NextComIn: 2 : LineOn: 0 : AreaOn: 9 : DamageNowHere: 1
		Time: 19 : NextComIn: 1 : LineOn: 0 : AreaOn: 10 : DamageNowHere: 1
		# Command Readed #
		END
		Press any key to continue . . .

InFile	OutFile	ProgData
1 1	1	# Command Readed # Time: 0 : NextComIn: 2 : LineOn: 0 : AreaOn: 0 : DamageNowHere: 0 Time: 1 : NextComIn: 1 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 1 # Command will be Executed # at time: 2
::d 0 1	4	# Command Readed # Time: 2 : NextComIn: 4 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 4 # Command will be Executed # at time: 3
2	2.50 0.0 0.0 0.0	Time: 3 : NextComIn: 3 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 7 Time: 4 : NextComIn: 2 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 8 Time: 5 : NextComIn: 1 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 9 # Command Readed # Press any key to continue . . .
::d 0 1		
4		
::e		
2 1	1	# Command Readed # Time: 0 : NextComIn: 2 : LineOn: 0 : AreaOn: 0 : DamageNowHere: 0 Time: 1 : NextComIn: 1 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 1 # Command will be Executed # at time: 2
::d 0 1	4	# Command Readed # Time: 2 : NextComIn: 4 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 4 # Command will be Executed # at time: 3
2	2.50 0.0 0.0 0.0	Time: 3 : NextComIn: 3 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 7 Time: 4 : NextComIn: 2 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 8 Time: 5 : NextComIn: 1 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 9 # Command Readed # END Press any key to continue . . .
::d 0 1		
4		
::e		
2 2	0000	# Command Readed # Time: 0 : NextComIn: 22 : LineOn: 0 : AreaOn: 0 : DamageNowHere: 0 Time: 1 : NextComIn: 21 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 1 Time: 2 : NextComIn: 20 : LineOn: 0 : AreaOn: 2 : DamageNowHere: 1 Time: 3 : NextComIn: 19 : LineOn: 0 : AreaOn: 3 : DamageNowHere: 1 Time: 4 : NextComIn: 18 : LineOn: 0 : AreaOn: 4 : DamageNowHere: 1 Time: 5 : NextComIn: 17 : LineOn: 0 : AreaOn: 5 : DamageNowHere: 1 Time: 6 : NextComIn: 16 : LineOn: 0 : AreaOn: 6 : DamageNowHere: 1 Time: 7 : NextComIn: 15 : LineOn: 0 : AreaOn: 7 : DamageNowHere: 1 Time: 8 : NextComIn: 14 : LineOn: 0 : AreaOn: 8 : DamageNowHere: 1 Time: 9 : NextComIn: 13 : LineOn: 0 : AreaOn: 9 : DamageNowHere: 1 Time: 10 : NextComIn: 12 : LineOn: 0 : AreaOn: 10 : DamageNowHere: 1 Time: 11 : NextComIn: 11 : LineOn: 0 : AreaOn: 11 : DamageNowHere: 1 Time: 12 : NextComIn: 10 : LineOn: 0 : AreaOn: 12 : DamageNowHere: 1 Time: 13 : NextComIn: 9 : LineOn: 0 : AreaOn: 13 : DamageNowHere: 1 Time: 14 : NextComIn: 8 : LineOn: 0 : AreaOn: 14 : DamageNowHere: 1 Time: 15 : NextComIn: 7 : LineOn: 0 : AreaOn: 15 : DamageNowHere: 1 # Command will be Executed # at time: 16 Time: 16 : NextComIn: 6 : LineOn: 0 : AreaOn: 15 : DamageNowHere: 7 Time: 17 : NextComIn: 5 : LineOn: 0 : AreaOn: 0 : DamageNowHere: 1 Time: 18 : NextComIn: 4 : LineOn: 1 : AreaOn: 0 : DamageNowHere: 1 # Command will be Executed # at time: 19 Time: 19 : NextComIn: 3 : LineOn: 1 : AreaOn: 0 : DamageNowHere: 7 Time: 20 : NextComIn: 2 : LineOn: 1 : AreaOn: 1 : DamageNowHere: 1 # Command will be Executed # at time: 21 Time: 21 : NextComIn: 1 : LineOn: 1 : AreaOn: 1 : DamageNowHere: 7 # Command Readed # Time: 22 : NextComIn: 3 : LineOn: 1 : AreaOn: 2 : DamageNowHere: 1 Time: 23 : NextComIn: 2 : LineOn: 1 : AreaOn: 3 : DamageNowHere: 1 Time: 24 : NextComIn: 1 : LineOn: 1 : AreaOn: 4 : DamageNowHere: 1 # Command will be Executed # at time: 25 # Command Readed # END Press any key to continue . . .
::mr 0 15	0000	
10	0000	
22	1	
::d 1 4	2.0 0.33 0.33 0.83	
3		
::e		
2 2	0000	# Command Readed # Time: 0 : NextComIn: 1 : LineOn: 0 : AreaOn: 0 : DamageNowHere: 0 # Command Readed # Time: 1 : NextComIn: 2 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 1 # Command will be Executed # at time: 2 Time: 2 : NextComIn: 1 : LineOn: 0 : AreaOn: 1 : DamageNowHere: 7 # Command will be Executed # at time: 3 # Command Readed # Time: 2 : NextComIn: 3 : LineOn: 1 : AreaOn: 2 : DamageNowHere: 1 Time: 23 : NextComIn: 2 : LineOn: 1 : AreaOn: 3 : DamageNowHere: 1 Time: 24 : NextComIn: 1 : LineOn: 1 : AreaOn: 4 : DamageNowHere: 1 # Command will be Executed # at time: 25 # Command Readed # END Press any key to continue . . .
::d 0 1	7	
1	0.83 0.0 0.0 0.0	
::r 0 1		
2		
::e		

