Curs 5 Aplicatii cu vectori, Tablouri multi-dimensionale, Debugging

Aplicatii cu vectori

- Calcul minim/maxim

```
int minim(int *v, int n)
//v=vector, n=nr elemente
{
    int min=v[0],i;
    for(i=0;i<n;i++)
        if (min>v[i]) //pentru a calcula maxim conditia devine maxim<v[i]
            min=v[i];
    return min;
}
```

Aplicatii cu vectori

- Media elementelor care indeplinesc o conditie (divizibile cu un parametru)

```
double medie(int*v, int n, int p)
{
    int count=0,suma=0, i;
    for(i=0;i<n;i++)
        if(v[i]%p==0)
        {
            suma+=v[i];
            count++;
        }
     if (count)
         return suma/(double)count;
    return DBL_MAX; //va fi interpretat ca eroare
}
```

Sortari

- Diversi algoritmi
  - Insert sort
    - Presupune ca vectorul este sortat si insereaza valoarea curenta in pozitia potrivita
    - Similar cu modul in care sortam cartile de joc in mana :)
  - Complexitate mare (O(n*n))
  - Complexitate buna daca vectorul este aproape sortat
  - Nu necesita memorie suplimentara

Insert Sort

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 50
int insertSort(int *v, int n)
{
        int val, pos, count=0,i;
        for(i=1;i<n;i++)
        {
                val=v[i];
                pos=i;
                while(pos>0 && val<v[pos-1])
                {
                        v[pos]=v[pos-1];
                        pos--;
                        count++;
                }
                v[pos]=val;
        }
        return count;
}
```

```c
int main()
{
        int nrEl,nrOp,i, v[MAX];
        srand(time(NULL));
        nrEl=10+rand()%40;//vectori intre 10 si 50 el
        for(i=0;i<nrEl;i++)
        {
                v[i]=rand()%100;
                printf(" %d",v[i]);
        }
        printf("\n");

        nrOp=insertSort(v,nrEl);
        for(i=0;i<nrEl;i++)
                printf(" %d",v[i]);
```

```
 98 44 50 87 13 76 54 85 13 41 58 34 51 4 64 38 92 19 51 46 12 8 64 82 66 96 84
88 47 97
 4 8 12 13 13 19 34 38 41 44 46 47 50 51 51 54 58 64 64 66 76 82 84 85 87 88 92
96 97 98
```

QuickSort

- Algoritm
  - Alege un element (pivot)
  - Aranjeaza lista astfel incat valorile mai mici decat pivotul sa fie inaintea lui, valorile mai mari sa fie dupa
  - Aplica recursiv pasii precedenti pe sub-liste
- O(nlogn)
- Implementat in C (qsort in stdlib.h)

qsort

- Se defineste o functie de comparatie

int compara(const void ***a, const void *b)**

{

/*a si b sunt adrese ale elementelor din vector

intoarce

- **o valoare mai mica decat 0 daca a si b sunt ordonate corect (a inaintea lui b)**
- **O valoare mai mare decat 0 daca a si b trebuie inversate**

- 0 daca sunt echivalente*/

}

qsort

- void qsort (void* base, size_t num, size_t size, int (*compar)(const void*,const void*));

int (*compar)(const void*,const void*) - functie care primeste 2 parametri de tip void* si intoarce int

```c
int comparaCresc(const void*a, const void*b)
{
        int *x= (int*)a, *y=(int*)b;
        return *x-*y;
}

int comparaDescresc(const void*a, const void*b)
{
        int *x= (int*)a, *y=(int*)b;
        return *y-*x;
}
```

```
25 28 28 15 38 61 7 36 69 7:
7 15 25 28 28 36 38 61 69 7:
71 69 61 38 36 28 28 25 15 :
7 15 25 28 28 36 38 61 69 7:
```

```c
srand(time(NULL));
nrEl=10+rand()%40;//vectori intre 10 si 50 el
for(i=0;i<nrEl;i++)
{
        v[i]=rand()%100;
        printf(" %d",v[i]);
}
printf("\n");

nrOp=insertSort(v,nrEl);
for(i=0;i<nrEl;i++)
        printf(" %d",v[i]);
printf("\n");
qsort(v, nrEl,sizeof(int),comparaDescresc);
for(i=0;i<nrEl;i++)
    printf(" %d",v[i]);
printf("\n");
qsort(v, nrEl,sizeof(int),comparaCresc);
for(i=0;i<nrEl;i++)
    printf(" %d",v[i]);
```

void*

- Pointer generic
- Ii poate fi atribuita orice adresa
- Nu poate fi dereferentiat (accesata valoarea) fara a-i fi facuta conversie explicita de tip

```
int main()
{
        int x; void*p;
        p=&x;
        *(int*)p=5;
        printf("%d",*(int*)p);
        return 0;
}
```

Pointeri la functii

- int fprel(int *v, int n, int param)

/*efectueaza o prelucrare asupra vectorului v in functie de parametrul param (de ex cauta param in vector sau intoarce nr de elemente divizibile cu param*/

- int (*fp)(int*,int,int); //fp este un pointer la o astfel de functie – intoarce int si are parametrii int*,int, int.

fp=&fprel;

int rez=fp(v,n,20);

Matrice

- tip_date numeMatrice[dim1][dim2];
- Este rezervat un spatiu de dimensiune dim1*dim2*sizeof(tip_date)
- numeMatrice[i][j] acceseaza elementul de pe linia i si coloana j
- Indexarea incepe de la 0 si pe linii si pe coloane
- numeMatrice[i] – este un vector ce contine elementele de pe linia i

Exemplu matrice

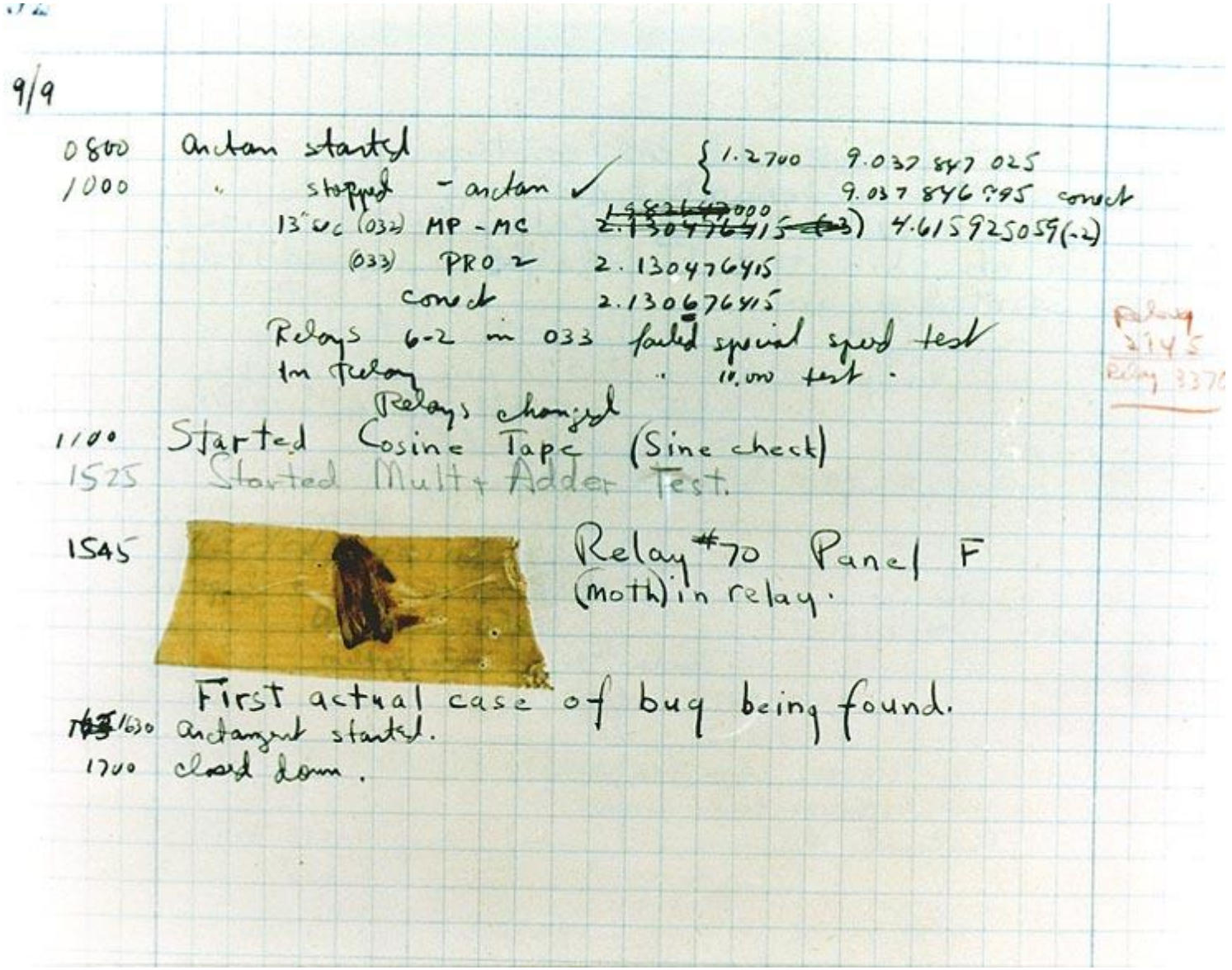
```c
#define MAX 3
#include<stdio.h>
#include<stdlib.h>
int main()
{
        int mat[MAX][MAX],n=3,*p,i,j;
        srand(time(NULL));
        p=mat[0];
        for(i=0;i<n;i++)
                for (j=0;j<n;j++)
                        mat[i][j]=rand()%100;

        for(i=0;i<n;i++)
        {
                for (j=0;j<n;j++)
                        printf("mat[i][j]=%d *(p+i*n+j)=%d\n",mat[i][j],*(p+i*n+j));
                printf("\n");
        }
        return 0;
}
```

ce facem cand programul nu merge?

- testare
- identificarea zonei de cod ce produce probleme
- analiza functionarii cu printf (printf debugging)
- analiza functionarii utilizand un debugger
- identificarea problemei in functie de eroare
- erori “clasice”

First actual case of bug being found



9/9

0800 Arctan started
1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 795 correct
13"UC (032) MP - MC ~~1.9826447000~~ ~~2.13047676415 - 63~~) 4.615925059(-2)
(033) PRO 2 2.13047676415
correct 2.13067676415
Relays 6-2 in 033 failed special speed test
In relay " 10.00 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
~~1700~~ 1630 Arctangent started.
1700 closed down.

testarea programului

- cazuri de test generale
  - cele pe care le introduceti de obicei
- cazuri de test particulare
  - date de intrare lipsa sau prea multe
  - numere f mari sau f mici…
- greseli facute de utilizator la introducerea datelor
  - alte tipuri de date decat cele cerute
  - datele primite in alta ordine …

identificarea zonei de cod cu probleme

- testarea fiecarei functii separat
- daca problemele apar intr-o functie mai lunga
  - izolam secvente de cod (comentam restul de ex) si vedem care merg
- pe secventa de cod cu probleme analizam functionarea codului

printf debugging

- adaugam instructiuni printf in cod pentru a urmari valorile variabilelor si pe ce ramuri ale if-urilor intra programul
- tiparim valorile care se schimba (sau care ar trebui sa se schimbe, rezultatele apelurilor de functie…)
- Este bine sa fortam scrierea pe ecran prin \n in print sau fflush(stdout) dupa
- Nerecomandat la programare paralela
- metoda utilizata in industrie prin scrierea in fisiere jurnal (log-uri)
- in mod tipic un program poate fi configurat sa scrie in fisiere jurnal mai multe tipuri de informatii pentru a putea identifica rapid erorile

debugging folosind un debugger

- notiuni de baza
  - debuggerul poate rula programul pas cu pas sau pana cand intalneste un breakpoint
  - Breakpoint-ul marcheaza o linie la care se va opri rularea programului in timpul rularii unui debugger
  - breakpoint-urile se adauga inainte de zona cu probleme

comenzi standard in debugger

- next (n) – se trece la linia urmatoare
- step (s) – daca la instructiunea curenta este un apel de functie se intra in functie si se continua executia pas cu pas
- continue (c) – continua executia fara intrerupere pana la urmatorul breakpoint sau pana la terminarea programului

exemplu printf debugging

```
int insertSort(int *v, int n)
{
        int val, pos, count=0,i;
        for(i=1;i<n;i++);
        {
                val=v[1];
                pos=i;
                printf("\nval=%d v[%i]=%d",val,i,v[i]);
                while(pos>0 && val<v[pos-1])
                {
                        v[pos]=v[pos-1];
                        pos--;
                        count++;
                }
                v[pos]=val;
        }
        return count;
}
```

am adaugat cateva greseli in codul functiei de inserare
- sa adaugam printf-uri

```
34 int main()
35 {
36         int nrEl,nrOp,i, v[MAX];
37         srand(time(NULL));
38         nrEl=10+rand()%40;//vectori intre 10 si 50 el
39         for(i=0;i<nrEl;i++)
40         {
41                 v[i]=rand()%100;
42                 printf(" %d",v[i]);
43         }
44         printf("\n");
45
46         nrOp=insertSort(v,nrEl);
47         for(i=0;i<nrEl;i++)
48                 printf(" %d",v[i]);
49         printf("\n");
```

```
57 84 4 42 26 92 7 44 22 76 9
84 57 84 4 42 26 92 7 44 22 76
```

```
int insertSort(int *v, int n)
{
        int val, pos, count=0,i;
        for(i=1;i<n;i++);
        {
                val=v[1];
                pos=i;
                printf("\nval=%d v[%i]=%d",val,i,v[i]);
                while(pos>0 && val<v[pos-1])
                {
                        v[pos]=v[pos-1];
                        pos--;
                        count++;
                }
                v[pos]=val;
        }
        return count;
}
```

```
                                                                5 16 15 42
 35 86

val=73 v[30]=1 73 24 73 35 28 9 44 78 1 44 25 42 81 52 6 49 2 57 29   6 83 73 24
16 61 56 16 15 42 35
 83 81 78 73 73 73 61 57 56 56 52 49 46 44 44 42 42 35 35 29 28 25    24 16 15 9
 6 2 1
 1 2 6 9 15 16 24 24 25 28 29 35 35 42 42 44 44 46 49 52 56 56 57 6   73 73 73 78
 81 83
```

se afiseaza o singura data

i=30

- ⇒ for-ul se executa o singura data cand i=30
- ⇒ descoperim ; dupa for

```
25 0 12 29 63 86 97 30 82 46 50 14 97 67 54 27 26 69 49 11 74 28 28 28 18 41 28 37 38 21 96 16 22 60 97 85 46
```

```
val=0 v[1]=0
val=25 v[2]=12
val=0 v[3]=29
val=25 v[4]=63
val=0 v[5]=86
val=25 v[6]=97
val=0 v[7]=30
val=25 v[8]=82
val=0 v[9]=46
val=25 v[10]=50
val=0 v[11]=14
val=25 v[12]=97
val=0 v[13]=67
val=25 v[14]=54
val=0 v[15]=27
val=25 v[16]=26
val=0 v[17]=69
val=25 v[18]=49
val=0 v[19]=11
val=25 v[20]=74
val=0 v[21]=28
val=25 v[22]=28
val=0 v[23]=28
val=25 v[24]=18
val=0 v[25]=41
val=25 v[26]=28
val=0 v[27]=37
val=25 v[28]=38
val=0 v[29]=21
val=25 v[30]=96
val=0 v[31]=16
val=25 v[32]=22
val=0 v[33]=60
val=25 v[34]=97
val=0 v[35]=85
val=25 v[36]=46 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25 0 25
```

```
int insertSort(int *v, int n)
{
        int val, pos, count=0,i;
        for(i=1;i<n;i++)
        {
                val=v[1];
                pos=i;
                printf("\nval=%d v[%i]=%d",val,i,v[i]);
                while(pos>0 && val<v[pos-1])
                {
                        v[pos]=v[pos-1];
                        pos--;
                        count++;
                }
                v[pos]=val;
        }
        return count;
}
```

vedem ca val isi schimba valoarea intr-un mod ff ciudat

vectorul final contine doar 0 si 25 (primele 2 valori)

- ⇒ ne uitam unde isi schimba val valoarea
- ⇒ observam ca este initializata cu val[1] in loc de val[i]

exemplu gdb debugging

```
[itresearch@web316 c5]$ gcc -g insert.c
[itresearch@web316 c5]$ gdb a.out
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-60.el6_4.1)
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/itresearch/programare/c5/a.out...done.
(gdb) break 11
Breakpoint 1 at 0x400663: file insert.c, line 11.
(gdb) run
Starting program: /home/itresearch/programare/c5/a.out
warning: no loadable sections found in added symbol-file system-supplied DSO at
0x7ffff7ffa000
 12 65 5 4 78 28 40 53 15 37 86 32 73 34 54 98 58 73 26 61 91 13 30 63 5 78 38 2
6 29 60 77 93 25 82 98 55 10 90 60 77 79 46 61 52 33 16 2 91

Breakpoint 1, insertSort (v=0x7fffffffe3d0, n=48) at insert.c:11
11                      printf("\nval=%d v[%i]=%d",val,i,v[i]);
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.107.el6_4.5.x86
_64
(gdb) print val
$1 = 65
(gdb) print i
$2 = 48
(gdb) print v[i]
$3 = -140141688
(gdb)
```

i incepe de la 48, val=65 -> v[1] si v[i] = valoare in afara vectorului
putem afisa si nr elemente al vectorului si vedem ca are 48 de elemente
=> putem observa ca a sarit peste for

```
(gdb) break 9
Breakpoint 1 at 0x40063f: file insert.c, line 9.
(gdb) run
Starting program: /home/itresearch/programare/c5/a.out
warning: no loadable sections found in added symbol-file system-supplied DSO at 0x7ffff7ffa000
 83 28 23 17 73 15 35 11 15 15 39 15

Breakpoint 1, insertSort (v=0x7fffffffe3d0, n=12) at insert.c:9
9                       val=v[1];
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.107.el6_4.5.x86_64
(gdb) print val
$1 = 4195648
(gdb) n
10                      pos=i;
(gdb) print val
$2 = 28
(gdb) print v[i]
$3 = 28
(gdb) c
Continuing.

Breakpoint 1, insertSort (v=0x7fffffffe3d0, n=12) a
9                       val=v[1];
(gdb) n
10                      pos=i;
(gdb) print val
$4 = 83
(gdb) print v[i]
$5 = 23
(gdb) print i
$6 = 2
(gdb) c
Continuing.

Breakpoint 1, insertSort (v=0x7fffffffe3d0, n=12) at insert.c:9
9                       val=v[1];
(gdb) n
10                      pos=i;
(gdb) print val
$7 = 83
```

```
 4 int insertSort(int *v, int n)
 5 {
 6         int val, pos, count=0,i;
 7         for(i=1;i<n;i++)
 8         {
 9                 val=v[1];
10                 pos=i;
11                 while(pos>0 && val<v
12                 {
13                         v[pos]=v[pos-
14                         pos--;
15                         count++;
16                 }
17                 v[pos]=val;
18         }
19         return count;
20 }
```

Erori clasice

- Segmentation fault
  - Nu ati pus & la scanf
  - Lucrati cu pointeri neinitializati
  - Aveti o functie recursiva ce nu are punct de iesire (ciclu infinit in apel de functii)
  - Aveti un ciclu infinit in care prelucrati un vector si iese din zona de memorie rezervata
- Apar valori ciudate – numere mari
  - Nu ati initializat suma/contor
  - Iesiti din spatiul vectorului (verificati indecsii)

Erori clasice

- Prelucrarea nu se termina – ciclu infinit
  - Conditia de iesire pusa gresit
  - Nu este actualizat contorul sau este actualizat gresit
- O conditie nu este niciodata true/false
  - Verificati valorile variabilelor inainte de conditie si dupa
  - Pot fi greseli de tipul if (a=0) sau greseli de logica

De retinut

- Testati fiecare functie separat
- Nu scrieti mult cod fara sa testati mai intai
- Verificati ca parametrii au fost transmisi corect
- Cand testati, verificati si cazurile particulare, daca merge pe un singur test nu inseamna ca merge pe orice test
- Nimeni nu scrie cod fara buguri :)