Tipuri de date. Operatori. Masurarea timpului de executie. Functii matematice.

Responsabili:

- Emil Racec (2012) [mailto:emil.racec@gmail.com]
- Bogdan-Cristian Drutu (2010) [mailto:bogdandrutu@gmail.com]

Obiective

În urma parcurgerii acestui laborator studentul va fi capabil să:

- utilizeze în cadrul programelor variabile declarate cu tipurile standard de date
- înțeleagă și utilizeze operatorii limbajului
- măsoare timpul de execuție al programelor scrise
- utilizeze funcțiile matematice din C

Noţiuni teoretice

Tipuri fundamentale de date

Tipurile de date reprezintă tipul de informație care poate fi stocat într-o variabilă. Un tip de data definește atât gama de valori pe care o poate lua o variabilă de un anume tip cât și operațiile care se pot efectua asupra ei. În continuare sunt prezentate tipurile fundamentale ale limbajului C, împreună cu o scurtă descriere a acestora:

- `char` - reprezentat printr-un număr pe 8 biți (un byte), stochează un caracter, definit în C printr-un număr în intervalul [-128; +127]. De observat că valorile pozitive codifică caracterele standard ASCII
- `int` - stochează numere întregi. Lungimea sa (și implicit plaja de valori) este dependentă de compilator si sistemul de operare considerat. De obicei, pe Linux, int se reprezintă pe 32 de biți (deci 4 bytes). În acest caz, poate memora numere între –2.147.483.648 și 2.147.483.647
- `float` - reprezintă un număr real stocat în virgulă mobilă, simplă precizie (7 cifre), în gama de valori 3.4E +/- 38(reprezentat pe 4 bytes)
- `double` - reprezinta un număr real stocat în virgulă mobilă, dublă precizie (15 cifre), în gama de valori 1.7E +/- 308 (reprezentat pe 8 bytes)

Acestor tipuri fundamentale li se mai pot adăuga un număr de calificatori, după cum urmează:

- `short` - aplicabil doar pentru int, rezultând, de obicei, un întreg pe 2 octeți
- `long` - aplicabil doar pentru int. Rezultă un întreg pe 32 de biți (4 octeți), schimbare semnificativa doar pentru Windows)
- `unsigned` - precizează faptul că valoarea variabilei este pozitivă. Aplicabil doar tipurilor întregi

În cazul în care este absolut necesar ca tipul întreg să aibă o anumită lungime (ca în cazul lucrului cu structuri, care va fi discutat într-un laborator viitor), este indicată consultarea cu atenție a documentației compilatorului. Compilatorul GCC pune în acest sens la dispoziția programatorului, următoarele tipuri de întregi cu lungime clar specificată:
`uint_8`, `uint_16`, `uint_32`, `uint_64` - pentru întregi fară semn pe 8, 16, 32 respectiv 64 de biți
`int_8`, `int_16`, `int_32`, `int_64` - pentru întregi cu semn reprezentați pe 8, 16, 32 respectiv 64 de biți.

Determinarea corectă a tipurilor de date care vor fi folosite este esențială pentru securitatea și buna funcționare a aplicațiilor pe care le scrieți. În cazul în care valoarea conținută de o variabilă depașește limitele impuse de tipul de date folosit, se produce așa-numit-ul over-flow care poate cauza erori aparent inexplicabile. (*Ca o anecdotă, în fiecare an (până acum trei ani), Bill Gates primea de la FISC o scrisoare prin care era somat să își plătească taxele, deoarece apărea in evidențele lor ca având datorii însemnate. Asta deoarece valoarea averii lui (mult peste 4.000.000.000$) producea un overflow în softul folosit de către FISC. În final situația a fost soluționată, introducând un câmp special pentru el în softul folosit. (A modifica softul peste tot ar fi introdus un plus de stocare nejustificat pentru fiecare din cei aproximativ 300.000.000 de cetățeni ai SUA.)*)

Operatori

Operatorii limbajului C pot fi unari, binari sau ternari, fiecare având o precedenţă şi o asociativitate bine definite. Tabelul următor sintetizează operatorii limbajului C. Operatorii sunt prezentaţi în ordine descrescătoare a priorităţii.

Precedenţă	Operator	Descriere	Asociativitate
1	**[]**	Indexare	stanga-dreapta
	. şi ->	Selecţie membru (prin structură, respectiv pointer)	stânga-dreapta
	++ şi --	Postincrementare/postdecrementare	stânga-dreapta
2	**!**	Negare logică	dreapta-stânga
	~	Complement faţă de 1 pe biţi	dreapta-stânga
	++ şi --	Preincrementare/predecrementare	dreapta-stânga
	+ şi -	+ şi - unari	dreapta-stânga
	*	Dereferenţiere	dreapta-stânga
	&	Operator *adresă*	dreapta-stânga
	(tip)	Conversie de **tip**	dreapta-stânga
	sizeof()	Mărimea în octeţi	dreapta-stânga
3	*	Înmulţire	stânga-dreapta
	/	Împărţire	stânga-dreapta
	%	Restul împărţirii	stânga-dreapta
4	**+ şi -**	Adunare/scădere	stânga-dreapta
5	**<< si >>**	Deplasare stânga/dreapta a biţilor	stânga-dreapta
6	**<**	Mai mic	stânga-dreapta
	<=	Mai mic sau egal	stânga-dreapta
	>	Mai mare	stânga-dreapta
	>=	Mai mare sau egal	stânga-dreapta
7	**==**	Egal	stânga-dreapta
	!=	Diferit	stânga-dreapta
8	**&**	ŞI pe biţi	stânga-dreapta
9	**^**	SAU-EXCLUSIV pe biţi	stânga-dreapta
10	**|**	SAU pe biţi	stânga-dreapta
11	**&&**	ŞI logic	stânga-dreapta
12	**||**	SAU logic	stânga-dreapta
13	**:?**	Operator condiţional	dreapta-stânga
14	**=**	Atribuire	dreapta-stânga
	+= şi -=	Atribuire cu adunare/scădere	dreapta-stânga
	***= şi /=**	Atribuire cu multiplicare/împărţire	dreapta-stânga
	%=	Atribuire cu modulo	dreapta-stânga
	&= si |=	Atribuire cu ŞI/SAU	dreapta-stânga
	^=	Atribuire cu SAU-EXCLUSIV	dreapta-stânga

	<<= şi >>=	Atribuire cu deplasare de biţi	dreapta-stânga
15	**,**	Operator secvenţa	stânga-dreapta

Trebuie avută în vedere precedenţa operatorilor pentru obţinerea rezultatelor scontate. Dacă unele tipuri de precedenţă (cum ar fi cea a operatorilor artimetici) sunt evidente şi nu prezintă (aparent) probleme (şi datorită folosirii lor dese), altele pot duce la erori greu de găsit. De exemplu, următorul fragment de cod nu produce rezultatul dorit, deoarece:

```
if (flags & MASK == 0) {
  ...
}
```

se evaluează mai întai egalitatea care produce ca rezultat (0 pentru False, și 1 pentru True) după care se aplică Și pe biți între flags și 1.

Pentru a obţine rezultatul dorit se vor folosi parantezele:

```
if ((flags & MASK) == 0) {
  ...
}
```

acum mai întâi se va face ȘI pe biți între flags și MASK, după care se verifică egalitatea.

O expresie este o secventă de operanzi și operatori (validă din punct de vedere al sintaxei limbajului C) care realizează una din funcțiile: calculul unei valori, desemnarea unui obiect (variabilă) sau funcţii sau generarea unui efect lateral.

O altă greşeală frecventă este utilizarea greşită a operatorilor **=** şi **==**. Primul reprezintă atribuire, al doilea comparaţie de egalitate. Apar deseori erori ca:

```
if (a = 2) {
  ...
}
```

Compilatorul consideră condiţia corectă, deoarece este o expresie validă în limbajul C care face atribuire, care se evaluează mereu la o valoare nenulă.

Măsurarea timpului de execuție a programelor

Uneori este utilă măsurarea timpului de execuție a unei anumite parți a unui program sau chiar a întregului program. În acest scop putem folosi funcția `clock()` din fișierul antet `time.h`. Această funcție întoarce o aproximare a numărului de cicluri de ceas trecute de la pornirea programului. Pentru a obţine o valoare în secunde, împărțim această valoare la constanta **CLOCKS_PER_SEC**. Funcţia are antetul:

```
clock_t clock(void);
```

Următorul fragment este un exemplu de utilizare a acestei funcții:

```
#include <stdio.h>
#include <time.h>


clock_t t_start, t_stop;
float seconds;

// Marcam momentul de inceput
t_start = clock();

// Executam operatia pentru care masuram timpul de executie
// [....]

// Marcam momentul de sfarsit
```

```
t_stop = clock();

seconds = ((float)(t_stop - t_start)) / CLOCKS_PER_SEC;

printf("Timp de executie: %.3f sec.\n", seconds);
```

Următorul fragment este un exemplu de funcție care are ca scop oprirea programului pentru un anumit timp:

```
void wait(int seconds){
  clock_t endwait;
  endwait = clock () + seconds * CLOCKS_PER_SEC ;
  while (clock() < endwait) {}
}
```

Funcții matematice

Fișierul antet `math.h` conține un set de funcții matematice des utilizate în programe. Câteva dintre acestea sunt:

Antet	Descriere
double asin(double arg); *double acos(double arg);*	Calculează arcsinusul/arccosinusul valorii **arg**; rezultatul este măsurat în radiani
double atan(double arg); *double atan2(double y, double x);*	Calculează arctangenta valorii **arg**, respectiv a fracției y/x
double floor(double num);	Întoarce cel mai mare întreg mai mic sau egal cu **num** (partea întreagă inferioară)
double ceil(double num);	Întoarce cel mai mic întreg mai mare sau egal cu num (partea întreagă superioară)
double sin(double arg); *double cos(double arg);* *double tan(double arg);*	Calculează sinusul/cosinusul/tangenta parametrului **arg**, considerată în radiani
double sinh(double arg); *double cosh(double arg);* *double tanh(double arg);*	Calculează sinusul/cosinusul/tangenta hiperbolică a parametrului **arg**
double exp(double arg);	Întoarce valoarea e^{arg}
double pow(double base, double exp);	Întoarce valoarea $base^{exp}$
double log(double num);	Calculează logaritmul natural (de bază **e**) al valorii **num**
double log10(double num);	Calculează logaritmul în baza 10 al parametrului
double sqrt(double num);	Calculează radăcina pătrată a parametrului
double fmod(double x, double y);	Întoarce restul împărțirii lui **x** la **y**
double fabs(double arg);	Întoarce valoarea absolută a lui **arg**

Generarea numerelor aleatoare

Valorile aleatoare (a căror valoare nu poate fi prezisă dinaintea rulării programului şi care diferă între 2 rulări) pot fi generate în C cu funcţia:

```
int rand(void);
```

care face parte din antetul `stdlib.h`. Această întoarce o valoare cuprinsă între 0 și **RAND_MAX** (valoare care este dependenta de librariile folosite, dar care se garantează a fi minim 32767).

Numerele generate nu sunt cu adevărat aleatoare, ci pseudo-aleatoare; aceste numere sunt uniform distribuite pe orice interval, dar șirul de numere aleatoare generate este dependent de prima valoare, care trebuie aleasă de

utilizator sau programator. Această valoare, numită **seed**, se selectează cu funcţia:

```
void srand(unsigned int seed);
```

Cea mai întalnită utilizare a funcției de inițializare presupune setarea unui **seed** egal cu valoarea ceasului sistemului de la pornirea programului, prin instrucțiunea:

```
srand((unsigned)time(NULL));
d = rand(); //generează valori random.
```

Funcția `time()` din fişierul antet `time.h` întoarce numărul de secunde trecute de la ora 00:00, din data de 1 ianuarie 1970. Funcția primește şi un parametru de tip pointer, care reprezintă adresa unei variabile în care se salvează valoarea returnată. Pentru laboratorul curent, parametrul va avea valoarea **NULL**.

Exerciţii de Laborator

1. [2p] Scrieti un program care numără descrescător de la 3→1, și după afișați `START` (cu interval de 1 secundă între fiecare număr). Exemplu:

   ```
   3 //așteaptă o secundă
   2 //așteaptă o secundă
   1 //așteaptă o secundă
   START
   ```

2. [2p] Fiind date 3 numere naturale, citite de la tastatură, să se verifice dacă ele reprezintă laturile unui triunghi, în cazul în care acestea pot forma un triunghi se va afișa mesajul “DA”, în caz contrar “NU”.
3. [3p] Să se citească de la tastatură 3 numere reale (a,b,c), care reprezintă parametrii unei ecuații de gradul 2 ($ax^2 + bx + c = 0$), să se verifice câte rădăcini reale are şi valoarea acestora. Se vor folosi mesajele: `Ecuația nu are rădăcini reale, Ecuația are o singură rădăcină reală egală cu …`, respectiv `Ecuația are două rădăcini reale … şi ….`
4. [3p] Scrieți un program care calculează 5 valori aleatoare și le afișează pe ecran, folosind ca seed ceasul sistemului, ca mai sus. Executați de mai multe ori programul. Inlocuiți apoi seed-ul cu o valoare constantă (`0x1234` de exemplu) si rulați din nou programul de câteva ori la rând. Ce observaţi? Care e explicaţia?

Bonus:

1. [2p] Se dau N puncte in plan (date prin coordonatele x şi y), citite de la tastatură, care reprezintă vârfurile unui poligon convex. Punctele sunt date în ordine trigonometrică (sau invers trigonometrică) să se calculeze:
   - perimetrul poligonului convex
   - aria poligonului convex

Referinţe

- Wikipedia - Operators in C and C++ [http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B]
- Wikipedia - C mathematical functions [http://en.wikipedia.org/wiki/C_mathematical_functions]