Tehnici de programare - TP

Cursul 6 – Funcții cu număr variabil de argumente
Argumente din linia de comandă

Ș.l. dr. ing. Cătălin Iapă

catalin.iapa@cs.upt.ro

Argumente din linia de comandă

Funcții cu număr variabil de argumente

Argumente din linia de comandă

Un program poate primi date când este lansat în execuție de la sistemul de operare, folosind linia de comandă. Parametrii vor fi scriși după numele programului pe care îl rulăm:

./prog ana 21

Această linie de comandă va transmite lui *prog* la execuție 2 parametri. Fiecare parametru este separat prin spațiu. Toți parametrii se transmit ca șiruri de caractere, chiar dacă ele de fapt sunt numere (ex: 21 este transmis ca un șir de 2 caractere, nu ca un număr de tip *int*).

Argumente din linia de comandă

În interiorul unui program C, parametrii din linia de comandă sunt preluați în felul următor: funcția *main* se declară ca având 2 parametri, care trebuie să aibă exact tipurile specificate mai jos:

*int main(int argc, char *argv[])*

- **argc** (arguments count) este un întreg care specifică numărul parametrilor din linia de comandă, incluzând numele programului executat. Dacă în linia de comandă vor fi dați 3 parametrii, *argc* va fi 4.
- **argv** (arguments vector) este un vector de șiruri de caractere care conține câte un parametru pe fiecare poziție. Este inclus și numele programului executat pe poziția 0, deci primul parametru propriu-zis începe de la indexul 1.

Argumente din linia de comandă

```
int main(int argc, char *argv[])
{
   int i, n;
      for (i = 0; i < argc; i++) {
            printf("Parametrul i este: %s",argv[i]);
            printf("\n");
      }
       return 0;
}
```

Dacă rulăm programul astfel:

./prg Ana 23 "Un program"

Va afișa:

Parametrul 0 este: ./prg
Parametrul 1 este: Ana
Parametrul 2 este: 23
Parametrul 3 este: Un program

Argumente din linia de comandă

Dacă vrem să parcurgem efectiv doar parametrii din linia de comandă vom parcurge parametrii doar de la 1 la argc:

```
for (i = 1; i < argc; i++)
        printf("Parametrul i este: %s",argv[i]);
```

Dacă vrem să prelucrăm ca și număr un argument primit din linia de comandă îl vom converti cu funcția atoi() din biblioteca <*stdlib.h*>*:*

```
nr = atoi(argv[i]);
```

Dacă un argument din linia de comandă conține și spațiu, vom pune întreg conținutul său între ghilimele la pornirea în execuție a programului:

```
./prg Ana 23 "Un program"
```

Argumente din linia de comandă

Funcții cu număr variabil de argumente

Funcții cu nr variabil de argumente

Funcțiile cu număr variabil de argumente (varargs) sunt funcții care pot accepta oricât de multe argumente.

Acestea sunt utile:

- atunci când nu știm în avans de câte argumente vom avea nevoie sau
- când vrem să evităm repetarea codului pentru funcții cu diferite numere de argumente.

Img: https://img.fruugo.com/product/5/62/130602625_max.jpg

Funcții cu nr variabil de argumente

Img: https://guardian.ng/wp-content/uploads/2018/06/Various-cocktails.-Photo-Getty-Images.jpg

Cocktail-urile sunt ca funcțiile cu număr variabil de argumente: primesc număr variabil de ingrediente

Arizona Sunset

1. 500 ml Sprite
2. 500 ml suc de portocale
3. Sirop Grenadine
4. Cirese Maraschino sau felii de portocala pentru decor
5. Gheata macinata sau cuburi

Cinderella

1. 400 ml suc de portocale
2. 400 ml suc de ananas
3. 75 apa minerala
4. 75 ml suc de lamaie
5. 50 ml sirop de zahar
6. Gheata macinata
7. Felii de ananas sau de portocala pentru ornarea paharului

Tropical Breeze

1. 300 ml suc de portocale
2. 300 ml suc de mango
3. 300 ml suc de ananas
4. Gheata macinata

https://www.arctic.ro/blog/cele-mai-populare-cocktailuri-fara-alcool-retete-delicioase-si-simplu-de-preparat/

Funcții cu nr variabil de argumente

Dacă avem definită corect funcția *fnva()* cu număr variabil de argumente vom putea să o apelăm ca în exemplele de mai jos:

fnva(1, 2, 3, 4);

fnva(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);

fnva(7);

fnva(8, 9, 7, 6, 5, 4, 3, 2, 1);

Practic la fiecare apel poate avea număr diferit de parametrii.

Funcții cu nr variabil de argumente

Limbajul C are un fel minimalist de a trata argumentele variabile. Ele sunt pur și simplu depuse într-o zonă de memorie, fără a se memora nimic despre ele: nici numărul și nici măcar tipul lor.

Această tratare minimalistă asigură o viteză mare de execuție, dar în schimb necesită ca funcția să mai primească informații auxiliare despre valorile cu care a fost apelată.

Funcții precum printf sau scanf sunt funcții cu care am lucrat și știm că acceptă număr variabil de parametrii. Acestor funcții le transmitem suplimentar numărul și tipul parametrilor lor prin placeholdere (ex: %d, %f, %c).

Funcții cu nr variabil de argumente

Pentru implementarea funcțiilor cu număr variabil de argumente, limbajul C pune la dispoziție mai multe funcții și macrouri definite în antetul

#include <stdarg.h>

Pentru prelucrarea parametrilor primiți prin astfel de funcții e necesar să se itereaze lista de argumente variabile și să se prelucreze pe rând fiecare argument.

Funcții cu nr variabil de argumente

Pentru a defini și implementa o funcție cu număr variabil de argumente trebuie să urmăm un număr de pași:

1. La definiția funcției, pe ultima poziție a parametrilor se pun trei puncte, **…** (ellipsis). Aceste trei puncte (care au un rol analogic lui ”etc”) se vor putea înlocui la apelul funcției prin oricâte argumente;

void functie(int n, ...)

2. În interiorul funcției (în implementarea funcției) definim o variabilă de tipul **va_list** (este un tip de date, analogic unui pointer, care va pointa la argumentele variabile);

va_list va;

Funcții cu nr variabil de argumente

Pentru a defini și implementa o funcție cu număr variabil de argumente trebuie să urmăm un număr de pași:

3. Apelăm funcția **va_start***(va, ultimul_arg_fix)* – acest apel atribuie lui *va*, care trebuie să fie de tipul *va_list*, adresa de început a argumentelor variabile. Pentru aceasta se folosește poziția ultimului argument fix al funcției, de dinainte de **...**

```
va_start(va, n);
```

4. Iterăm lista de argumente variabile folosind macroul **va_arg***(va,tip)* cu 2 parametrii: lista de argumente variabile care se iterează și tipul de date corespunzător argumentului curent. La fiecare apel, *va_arg* va returna valoarea argumentului curent din listă, ca fiind de tipul dat și va trece la următorul argument.

```
int a = va_arg(va, int);
```

Funcții cu nr variabil de argumente

Pentru a defini și implementa o funcție cu număr variabil de argumente trebuie să urmăm un număr de pași:

5. În final, după ce au fost preluate și prelucrate toate argumentele funcției, se va apela **va_end**(*va*) - după folosirea listei de argumente variabile *va*, trebuie apelat *va_end* pentru a elibera eventualele resurse alocate lui *va*.

```
va_end(va);
```

Exemplu (1)

Să se calculeze suma argumentelor variabile (întregi). Numărul argumentelor variabile va fi specificat de argumentul fix al funcției.

```
#include <stdarg.h>

int suma(int n, ...) {
   va_list va;
   int suma = 0;
   va_start(va, n);

   for(int i = 0; i < n; i++) {
      suma += va_arg(va, int);
   }
   va_end(va);
   return suma;
}
```

La apel:

suma(3, 1, 2, 3) -> rez: 6

suma(5, 5, 5, 5, 5, 5) -> rez: 25

suma(1, 100) -> rez: 100

Exemplu (2)

Să se afișeze maximul dintre argumentele variabile (numere reale).
Știm că ne oprim din preluat argumente când întâlnim 0:

```
#include <stdarg.h>
double maxim(double x, ...) {
   va_list va;
   double max = x, y;
   va_start(va, x);

    do{
      y = va_arg(va, double);
      if ( y > max)  max = y;
   }while((y-0) > 0.001);
   va_end(va);
   return max;
}
```

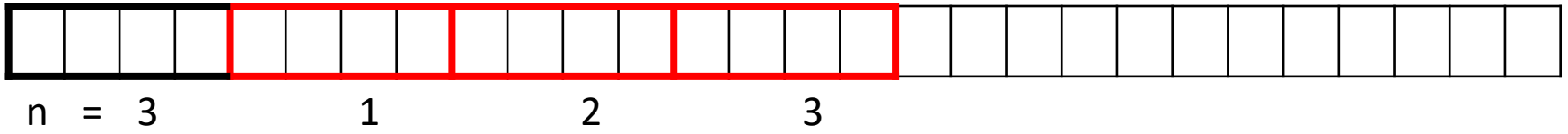
La apel:
maxim(7.1, 3.3, 8.0, 0.0) -> rez: 8.0
maxim(1.2, (double)7, 0.0) -> rez: 7.0
maxim(0.55, 1.1, (double)0) -> rez: 1.1

Particularități

Din Exemplul 1:

La definiție: int suma(int n, ...)

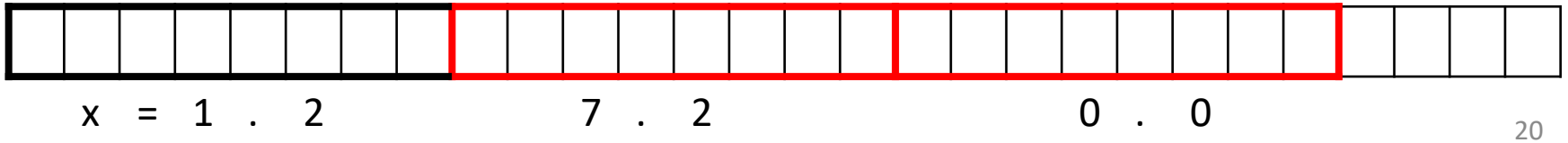
La apel: suma(3, 1, 2, 3); -> rez: 6



Din Exemplul 2:

La definiție: double maxim(double x, ...)

La apel: maxim(1.2, 7.2, 0.0); -> rez: 7.2

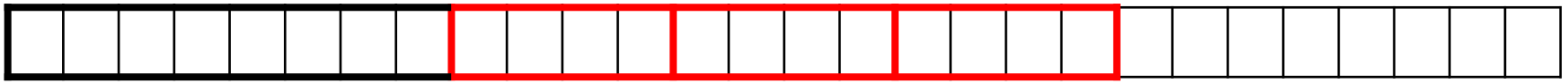


Particularități

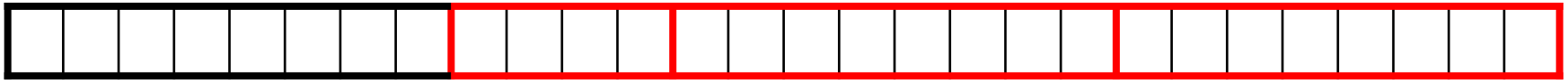
Din Exemplul 2:

La definiție: double maxim(double x, ...)

La apel: maxim(1, 3, 7, 0);



La apel: maxim(1.0, 3, (float)7, 0.0);



Particularități

La iterarea listei de argumente cu va_arg(va, tip) trebuie ținut cont că pentru argumentele variabile (varargs), compilatorul face unele conversii implicite. Cele mai importante reguli pentru conversii implicite sunt:

- *float* → *double*
- orice tip întreg cu o dimensiune mai mică decât *int* (*char*, *short*) → *int*
- conversiile tipurilor întregi țin cont că tipul este sau nu cu semn (ex: *short* → *int*, *unsigned short* → *unsigned int*)

O constantă numerică, dacă nu are punct sau exponent, este considerată ca fiind de tip *int*. Din acest motiv, dacă dorim să pasăm constante numerice într-o listă variabilă de argumente, iar acolo este nevoie de valori reale, va trebui să scriem constantele numerice ca fiind reale (ex: să adăugăm **.0** după ele, chiar dacă nu au parte zecimală sau să folosim cast (double) în fața constantei).

Funcții cu nr variabil de argumente

Reluăm faptul că nu există stocate de către limbajul de programare C nici numărul de argumente variabile și nici tipurile lor, deci aceste informații trebuie să fie accesibile prin alte mijloace.

În exemplele anterioare am transmis în interiorul funcției numărul de argumente variabile, fie transmițând direct numărul în argumentul fix, fie adăugând o valoare specială la final de șir (0).

Dacă argumentele cu număr variabil sunt de tipuri diferite va fi nevoie să transmitem și tipul fiecăruia, într-un mod asemănător cu cel în care operează scanf sau printf.

Exemplu de implementare (3)

Să se afișeze pe ecran paramentrii variabili primiți în funcție. Parametrii vor fi de tipuri diferite. Tipurile lor se vor specifica într-un șir de caractere(primul parametru) astfel: i – întreg, f – real, s - șir de caractere.

```
void afisare(char *sir, ...) {
    int a; double x; char s[10];
    va_list va;
    va_start(va, sir);
    for(int i = 0; i<strlen(sir); i++)
        switch(sir[i]) {
            case 'i': a=va_arg(va,int); printf("%d ", a); break;
            case 'f': x=va_arg(va,double); printf("%f ", x); break;
            case 's': strcpy(s, va_arg(va,char *)); printf("%s ", s); break;
        }
    va_end(va);
}
```

La apel: *afisare("iifsi" ,2 , 50, 5.5, "TP", 10);*

Particularități

Pentru situații în care se cere parcurgerea argumentelor variabile de mai multe ori, aceasta se poate realiza în două feluri:

- după ce se încheie o parcurgere cu *va_end*, se poate folosi din nou *va_start* pentru reinițializarea listei de argumente variabile
- dacă dorim să ținem minte o anumită poziție din lista de argumente variabile, putem salva lista respectivă într-o copie, folosind macroul **va_copy***(va_destinație, va_sursă)*. *va_copy* copiază *va_sursă* în *va_destinație*, deci ambele liste vor pointa la același argument. *va_destinatie* va trebui eliberată și ea în final cu *va_end*.

Particularități

În antetul *<stdio.h>* există mai multe variante pentru funcțiile *printf* și *scanf*. Unele variante ne permit să transmitem direct o variabilă va_list:

int **vprintf***(const char *format,va_list arg)* - primește argumentele sub forma unei liste varargs și le scrie la *stdout*

int **vfprintf***(FILE *stream,const char *format,va_list va)* - primește argumentele sub forma unei liste varargs și le scrie în fișierul specificat

```
va_list va;
va_start(va, n);
vprintf("%d %d %f %s %d", va);
```

Cocktail sau Funcții cu nr variabil de arg?

Când facem un cocktail, putem folosi diferite ingrediente, cum ar fi suc de fructe, siropuri, lichioruri sau alte băuturi. Fiecare cocktail poate avea *un număr variabil de ingrediente*, iar cantitățile acestor ingrediente pot varia în funcție de preferințele personale sau de reteta specifica.

Pentru a crea o funcție care să creeze un cocktail, *putem utiliza un număr variabil de argumente pentru a indica tipurile și cantitățile de ingrediente necesare*.

În acest exemplu, funcția cu număr variabil de argumente permite crearea de cocktail-uri personalizate, *fără a fi necesar să definim o funcție separată pentru fiecare combinație de ingrediente*.

Comparație sugerată pe https://chat.openai.com/chat

Vă mulțumesc!