Tema 1 - printf

- Deadline: 08.11.2020 23:55
- Data publicării: 25.10.2020, 23:55
- Ultima actualizare a checker-ului: 28.10.2020, 21:33
- Responsabili:
  - Lucian-Ștefan TEODORESCU [mailto:stefanl.teodorescu@gmail.com]
  - Dorin Andrei GEMAN [mailto:doringeman@gmail.com]
  - Bogdan-Cristian FIRUȚI [mailto:firutibogdan@gmail.com]

Enunț

`printf` este o funcție din biblioteca standard C care afișeaza la `stdout` datele primite ca argumente formatate după un anumit șablon, primit de asemenea ca prim argument. Semnătura acestei funcții este:

```
int printf(const char *format, ...);
```

În continuare, vom discuta despre un subset din funcționalitățile lui `printf`, cele relevante pentru temă. Pentru mai multe detalii, consultați pagina de manual `man 3 printf`.

Primul parametru al funcției se numește format string și este cel care determină ce afișează `printf`. Acesta este un șir de caractere format din zero sau mai multe directive:

- caractere obișnuite (cu excepția lui %), care sunt afișate neschimbate la output;
- specificații de conversie, care rezultă în consumarea a zero sau mai multe argumente;

Fiecare specificație este introdusă prin caracterul % și se încheie printr-un specificator de conversie.

Deoarece format string-ul poate conține oricâte astfel de specificații, `printf` poate primi un număr variabil de argumente. Aceste argumente trebuie să corespundă cu specificatorii din format string, în aceeași ordine. De exemplu, `printf("%d", 2)` va produce output-ul `2`, iar `printf("%d %d", 2, 2)` va produce output-ul `2 2`.

Specificatorii ce vor fi implementați sunt următorii:

- %d întreg, convertit la reprezentare zecimală, cu semn;
- %u întreg, convertit la reprezentare zecimală, fara semn;
- %x întreg, convertit la reprezentare hexazecimală, fara semn;
- %c caracter, convertit la reprezentarea ASCII;
- %s pointer la un șir de caractere, ce va fi afișat neschimbat;

Valoarea de retur a lui `printf`, în caz de succes, este numărul de caractere afișate (excluzând byte-ul null folosit pentru terminarea string-urilor).

Să se implementeze în limbajul de programare C funcția `int iocla_printf(const char *format, ...)`, care acceptă specificatorii listați mai sus, oferind comportament similar cu `printf` din biblioteca standard C.

În implementarea funcției `iocla_printf` este **interzisă** folosirea tuturor funcțiilor din biblioteca standard C din familia `printf` care oferă deja comportamentul cerut. Acestea includ, dar nu sunt limitate la `printf, fprintf, dprintf, sprintf, snprintf, vprintf, vfprintf, vdprintf, vsprintf, vsnprintf`

Implementare

În limbajul C, pentru implementarea funcțiilor cu mai mulți parametri se folosește familia de macro-uri `va_arg` din header-ul `stdarg.h`. Mai multe detalii puteți găsi în pagina de manual `man 3 va_arg`.

Cunoscând primul parametru al unei funcții (în cazul nostru `const char *format`), putem determina următorii parametri prin folosirea macro-urilor `va_arg`. Pentru a ști tipul următoarelor argumente, acestea vor fi consumate în paralel cu format string-ul.

Un exemplu de implementare a unei funcții cu număr variabil de parametri, care primește ca prim argument numărul de parametri de tip întreg ce urmează, și întoarce suma acestor întregi, este prezentat în continuare:

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

int sum(size_t count, ...)
{
        va_list args;
        size_t i;
        int result = 0;

        va_start(args, count /* the first parameter */);

        for (i = 0; i < count; i++)
                result += va_arg(args, int);

        return result;
}

int main(void)
{
        /* TODO: replace with iocla_printf */
        printf("Sum = %d\n", sum(5, 10, 20, 30, 40, 50));

        return 0;
}
```

Pentru afișarea la `stdout` este pusă la dispoziție în scheletul temei o funcție wrapper peste apelul de sistem `write`. Folosirea apelului de sistem `write` nu face parte din scopul temei, dar cei mai aventuroși dintre voi pot găsi mai multe informații în pagina de manual `man 2 write`.

Funcția din schelet `int write_stdout(const char *token, int length)` are doi parametri:

- `token` un șir de caractere
- `length` lungimea acestuia

Nu este necesar ca șirul `token` să aibă caracterul null ('\0') la final, întrucât se vor scrie doar câte caractere sunt precizate de argumentul `length`.

De exemplu, apelul `write_stdout("123", 3)` va afișa `123`, iar apelul `write_stdout("Ana are mere", 7)` va afișa `Ana are`.

Exemple de rulare

```
printf("%d", 3); // 3
printf("%u", 3); // 3
printf("%c", 65); // A
printf("%x", 123); // 7b
printf("%s", "Ana are mere"); // Ana are mere
printf("Ana are %d mere si \t%%Hex@na%% are 0x%x mere\n", 5, 16); // Ana are 5 mere si       %Hex@na% are 0x10 mere
```

Trimitere și notare

Temele vor trebui încărcate pe platforma vmchecker [https://vmchecker.cs.pub.ro/ui/#IOCLA] (în secțiunea IOCLA) și vor fi testate automat. Arhiva încărcată trebuie să fie o arhivă `.zip` care să conțină:

- fișierul sursă ce conține implementarea temei, denumit `tema1.c`
- fișier `README` ce conține descrierea implementării

Punctajul final acordat pe o temă este compus din:

- punctajul obținut prin testarea automată de pe vmchecker - 90%
- fișier README - 10%

A fost facut un update al regulamentului de realizare a temelor - s-a introdus o secțiune pentru depunctări, vă rugăm să o parcurgeți. De asemenea daca nu ați parcurs regulamentul de realizare a temelor deja vă recomandăm sa o faceți.

Mașina virtuală folosită pentru testarea temelor de casă pe vmchecker este descrisă în secțiunea Mașini virtuale din pagina de resurse.

Precizări suplimentare

- Tema **NU** se poate rezolva in C++.
- Folosirea altor funcții din biblioteca standard C este permisă, atât timp cât acestea nu implementează complet sau parțial funcționalități cerute în temă. De exemplu, folosirea `strlen, strchr, strtok` etc. este permisă.

Resurse

Arhiva ce conține checkerul, testele și fișierul de la care puteți începe implementarea este aici [https://ocw.cs.pub.ro/courses/_media/iocla/teme/tema1-resurse.zip].