AS-0.1101 C-Ohjelmoinnin peruskurssi
Tentti 09.05.2008, Raimo Nikkilä

Ohjeet

Kaikki ohjelmointitehtävät tulee toteuttaa C-kielellä hyvää ohjelmointityyliä noudattaen. Vastaa **korkeintaan viiteen** tehtävään. Tentti arvostellaan asteikolla 0-25p ja kaikkien tehtävien painoarvo on sama (5 pistettä / tehtävä). Tentissä saa käyttää funktiolaskinta sekä tentissä jaettua C-kielen standardikirjaston minireferenssiä. Kaikki tentin tehtävät ovat tehtävissä minireferenssissä listatuilla funktioilla mutta kaikkia C-kielen standardikirjaston funktioita saa halutessaan käyttää.

Vastaa tehtäviin siten että **tehtävien 1. ja 2.** vastaukset ovat omalla konseptillaan, **tehtävien 3. ja 4.** vastaukset ovat omalla konseptillaan ja **tehtävien 5. ja 6.** vastaukset ovat omalla konseptillaan.

Kirjoita edes opiskelijanumerosi selkeällä käsialalla tenttipapereihin.

1. Tehtävä

Mitä alla oleva ohjelma tulostaa? Vastaukseksi riittää pelkkä tuloste ilman perusteluja.

```
1  #include <stdio.h>
2
3  void printByte(unsigned char byte, FILE* out)
4  {
5      for (size_t i = 0;i < 8;i++)
6          fputc((byte & (0x80 >> i))?'1':'0', out);
7      fputc('\n', out);
8  }
9
10 int main(void)
11 {
12     const unsigned char byte = 0xD5;
13
14     printf("A: "); printByte(~byte, stdout);
15     printf("B: "); printByte((byte | byte) ^ 0xCD, stdout);
16     printf("C: "); printByte((byte << 4) >> 3, stdout);
17     printf("D: "); printByte((byte && 0x64) << 1, stdout);
18     printf("E: "); printByte((byte + 8) / 2, stdout);
19
20     return 0;
21 }
```

2. Tehtävä

Toteuta ohjelma joka lukee käyttäjältä lukuja ja tulostaa tämän jälkeen kaikki tiettyä arvoa pienemmät luvut. Ohjelma lukee ensin käyttäjältä luettavien lukujen määrän jonka jälkeen ohjelma lukee luvut `double`-tyyppisinä arvoina. Lopuksi ohjelma lukee käyttäjältä raja-arvon ja tulostaa luetuista luvuista kaikki raja-arvoa pienemmät luvut omille riveilleen siten, että rivillä on luvun indeksi (Monesko luku on luetuista luvuista), välilyönti ja luku kahden desimaalin tarkkuudella.

Käyttäjälle ei tarvitse tulostaa minkäänlaisia kehotteita eikä syötteelle tarvitse tehdä minkäänlaista järkevyys-, tai virheentarkastusta. Kaikki syöte luetaan `stdin`-virrasta ja kaikki ulostulo kirjoitetaan `stdout`-virtaan. Jaa ohjelmasi toiminta järkevästi vähintään kahteen funktioon.

Ohjelman suorittamat I/O-toiminnot järjestyksessä:

1. Luetaan käyttäjältä lukujen määrä
2. Luetaan käyttäjältä kaikki luvut
3. Luetaan käyttäjältä raja-arvo
4. Tulostetaan kaikki raja-arvoa pienemmät luvut omille riveilleen, siten että rivillä on luvun järjestysnumero, välilyönti ja itse luku kahden desimaalin tarkkuudella

3. Tehtävä

Selitä lyhyesti minkä toiminnon kukin alla olevista kolmesta funktiosta toteuttaa. Älä kuvaa funktion sisäistä toimintaa vaan ainoastaan se millaisen muutoksen funktio saa aikaan syötteilleen tai minkä arvon funktio laskee syötteistä.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4
5  void function1(char *str)
6  {
7      for (size_t i = 0;str[i] != '\0';i++)
8          toupper(str[i]);
9  }
10
11 size_t function2(const char *str)
12 {
13     const char *ptr = str;
14     while (*ptr++);
15     return ptr - str - 1;
16 }
17
18 #define MACRO(L, R) ((L) > ((R))?(L):(R))
19
20 double function3(const double *ptr, size_t s)
21 {
22     double tmp = *ptr;
23     while (--s)
24         tmp = MACRO(tmp, *(s + ptr));
25     return tmp;
26 }
```

4. Tehtävä

Toteuta `stringArrayCopy()`-funktio joka kopioi merkkijonotaulukon jonka viimeinen alkio on `NULL`-arvo. Funktio ottaa ensimmäisenä parametrinaan osoitinmuuttujan johon kopio sijoitetaan ja toisena parametrinaan kopioitavan merkkijonotaulukon. Funktio palauttaa merkkijonotaulukossa olevien merkkijonojen määrän johon ei ole laskettu mukaan merkkijonotaulukon lopettavaa `NULL`-arvoa.

Merkkijonotaulukko on kopioitava täysin, eli pelkkä osoitinmuuttujien arvojen kopiointi ei riitä. Funktion on toimittava oikein riippumatta siitä miten kopiotavan merkkijonotaulukon muisti on varattu. `malloc()`-funktiokutsujen voi olettaa onnistuvan aina.

```
1 /* First parameter: pointer where the copy is stored
2  * Second parameter: the string array to copy
3  *                   the function may not change this parameter
4  * Return value: size of the string array (number of elements) */
5 size_t stringArrayCopy(char ***copy, char **array);
```

Alla oleva sudoku **ei ole** tenttitehtävä tai millään tavalla osa kurssisuoritusta, eikä sitä oteta tentin tai kurssin arvostelussa huomioon mitenkään. Mutta jos olet palauttamassa tyhjän paperin niin voit kuluttaa aikaasi ratkaisemalla sitä.

	4							7
		9	2				6	
				4	7			
						1		
		2	9		3		4	
9								2
8			6			2		
1			8			3	7	6
			7	9	5			

5. Tehtävä

Ohessa on abstraktin tietotyypin, merkkilistan `List`-määritelmä. Tietotyypin lisäksi on toteutettu kolme funktiota; `strToList()`-funktio joka muuttaa merkkijonon listaksi, `listToStr()`-funktio joka muuttaa listan merkkijonoksi ja `listDestruct()`-funktio joka vapauttaa kaiken listalle varatun muistin.

Toteutuksessa on kuitenkin muutamia selkeitä virheitä, etsi nämä virheet ja kerro hyvin lyhyesti miten korjaisit ne. `malloc()`-funktiokutsujen oletetaan onnistuvan aina jolloin `malloc()`-funktion paluuarvoa ei tarvitse tarkastaa.

Vihje: `strToList()`-funktiossa on ainoastaan yksi virhe.

```
1  #include <stdlib.h>
2  /* Definition of the List ADT */
3  typedef struct ListNodeStruct
4  {
5      unsigned char chr;
6      struct ListNodeStruct *next; /* NULL for the last node */
7  } *List;
8
9  /* Creates a list representation for the given string */
10 List strToList(const char *str)
11 {
12     List list, *ptr = &list;
13
14     while (*str)
15     {
16         *ptr = malloc(sizeof (List));
17         (*ptr)->chr = *str++;
18         ptr = &(*ptr)->next;
19     }
20     *ptr = NULL;
21
22     return list;
23 }
24
25 /* Creates a string representation for the given List */
26 char* listToStr(const List lst)
27 {
28     size_t size;
29     for (List ptr = lst;ptr;ptr=ptr->next)
30         size++;
31
32     char *str = malloc(size + 1);
33
34     for (List ptr = lst;ptr;ptr++)
35         str[size++] = ptr->chr;
36
37     return str;
38 }
39
40 /* Frees all memory allocated for a list */
41 void listDestruct(List lst)
42 {
43     if (lst)
44     {
45         free(lst);
46         listDestruct(lst->next);
47     }
48 }
```

6. Tehtävä

Toteuta `filterFPArray()`-funktio joka ottaa parametrinaan `TFunction*`-tyyppiä olevan funktio-osoitintaulukon ja suodattaa taulukon annetulla suodatusfunktiolla. Funktio palauttaa funktio-osoitintaulukon jossa on kaikki alkuperäisen funktio-osoitintaulukon funktiot joille suodatusfunktio on palauttanut epätoden arvon (nollan). Alkuperäisen funktio-osoitintaulukon ja palautettavan funktio-osoitintaulukon lopettaa `NULL`-arvo.

```
1 typedef void* (*TFunction)(void *);
2
3 TFunction* filterFPArray(TFunction *array,
4                          int (*filter)(TFunction))
```

ctype.h

```
int isalnum(int ch);
int isalpha(int ch);
int isdigit(int ch);
int islower(int ch);
int ispunct(int ch);
int isspace(int ch);
int isupper(int ch);
int tolower(int ch);
int toupper(int ch);
```

math.h

```
double pow(double base, double exponent);
double sqrt(double value);
```

stdio.h

```
int printf(const char* format, ...);
int fprintf(FILE *stream, const char* format, ...);
int sprintf(char *str, const char* format, ...);
int scanf(const char* format, ...);
int fscanf(FILE *stream, const char* format, ...);
int sscanf(const char *str, const char* format, ...);
int fgetc(FILE *stream);
int fputc(int ch, FILE *stream);
int getchar(void);
int putchar(int ch);
```

stdlib.h

```
void free(void *ptr);
void* malloc(size_t size);
void* realloc(void* ptr, size_t size);
void abort(void);
void exit(int);
void qsort(void* base, size_t nmemb, size_t size, int (*cmp)(const void*, const void*));
int abs(int val);
```

string.h

```
char* strcat(char *dest, const char *src);
char* strchr(const char *str, int ch);
int strcmp(const char *str1, const char *str2);
char* strcpy(char *dest, const char *src);
size_t strlen(const char *str);
size_t strxfrm(char *dest, const char *src, size_t n);
```