AS-0.1103 C-ohjelmoinnin peruskurssi
Aalto-yliopiston sähkötekniikan korkeakoulu
Tentti 15.05.2013, Raimo Nikkilä

Ohjeet

Kaikki ohjelmointitehtävät tulee toteuttaa C-kielellä hyvää ohjelmointityyliä noudattaen.

Tentti arvostellaan asteikolla 0-25p ja kaikkien tehtävien painoarvo on sama (5 pistettä / tehtävä).

Vastaa **viiteen** vapaavalintaiseen tehtävään.

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.

Jaa vastauksesi konsepteille siten, että:

Tehtävät **1. ja 2.** ovat omalla konseptillaan

Tehtävät **3. ja 4.** ovat omalla konseptillaan

Tehtävät **5. ja 6.** ovat omalla konseptillaan

Tavussa (`char`) on aina 8 bittiä ja eniten merkitsevä bitti on aina vasemmalla.

Kurssipalautelomake on auki NG:ssä. Kurssipalautteen antamisesta saa yhden lisäpisteen kurssiarvosteluun.

Kirjoita edes opiskelijanumerosi selkeällä käsialalla tenttipapereihin.

1. Tehtävä

Vastaa seuraaviin kysymyksiin annetun ohjelmakoodin perusteella.

A) Funktio `reverseArray()` tekee oletuksia parametreistaan.
Ilmaise nämä oletukset yhtenä tai useampana `assert()`-lausekkeena.

B) Mitkä olisivat seuraavien lausekkeiden arvot `reverseArray()`-funktiossa?

I) `sizeof(array);`

II) `sizeof(*array);`

III) `sizeof(array[0]);`

C) Vaihtamalla pelkästään funktion `reverseArray()` ensimmäisen parametrin tyypin, voisiko funktiolla kääntää:

I) Taulukon `char` arvoja?

```
void reverseArray(char* array, int size);
```

II) Taulukon `double` arvoja?

```
void reverseArray(double* array, int size);
```

III) Taulukon `char*` arvoja?

```
void reverseArray(char** array, int size);
```

D) Mitä funktio tekisi jos `for`-lauseesta poistettaisiin aaltosulut?

```
for (size_t i = 0;i < ((size_t)size) / 2;i++)
  SWAP(array[i], array[size - i - 1]);
```

E) Mitä funktio tekisi jos `for`-lauseen ehdosta poistettaisiin kahdella jakaminen?

```
for (size_t i = 0;i < (size_t)size;i++)
{
  SWAP(array[i], array[size - i - 1]);
}
```

```
1  #include <stddef.h>
2
3  #define SWAP(L, R) int tmp = (L); (L) = (R); (R) = tmp
4
5  void reverseArray(int* array, int size)
6  {
7    for (size_t i = 0;i < ((size_t)size) / 2;i++)
8    {
9      SWAP(array[i], array[size - i - 1]);
10   }
11 }
```

2. Tehtävä

Toteuta funktio `sortBits()`, joka järjestää `unsigned int` tyyppisen arvon bitit.

Bitit järjestetään siten, että 1-bitit tulevat arvon eniten merkitsevään (vasempaan) päähän. Vastaavasti 0-bitit tulevat arvon vähiten merkitsevään (oikeaan) päähän.

Funktio palauttaa järjestetyn arvon.

```
unsigned int sortBits(unsigned int value);
```

Esimerkkinä funktion toiminnasta: `sortBits(0x1248)` palauttaa arvon `0xF0000000`

3. Tehtävä

Ohessa on funktioita jotka käsittelevät yksinkertaista yksisuuntaista listarakennetta.
Funktioiden toteutus valitettavasti on kaukana parhaasta mahdollisesta ja niissä on muutamia virheitä.

Tunnista nämä virheet ja kuvaile **lyhyesti** miten korjaisit ne.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <assert.h>
4
5  typedef struct node
6  {
7    int value;
8    struct node* next;
9  } Node;
10
11 /* Lisää uuden solmun listan ensimmäiseksi solmuksi. */
12 Node* listPrepend(Node* list, int value)
13 {
14   Node* new = malloc(sizeof(new));
15   new->value = value;
16   new->next = list;
17   return new;
18 }
19
20 /* Lisää uuden solmun listan viimeiseksi solmuksi. */
21 Node* listAppend(Node* list, int value)
22 {
23   if (!list)
24     return listPrepend(list, value);
25   Node* ptr = list;
26   while (ptr)
27     ptr = ptr->next;
28   ptr->next = listPrepend(list, value);
29   return list;
30 }
31
32 /* Poistaa listasta pelkästään ensimmäisen solmun jossa on
33  *  parametrina saatu arvo. */
34 void listRemove(Node** list, int value)
35 {
36   assert(list);
37   while (*list)
38     if ((*list)->value == value)
39     {
40       Node* tmp = (*list)->next;
41       free (*list);
42       *list = tmp;
43     }
44     else
45       list = &(*list)->next;
46 }
47
48 /* Laskee listan pituuden. */
49 size_t listLen(Node* list)
50 {
51   assert(list->next);
52   size_t n;
53   while (list)
54   {
55     list = list->next;
56     n++;
57   }
58   return n;
59 }
```

4. Tehtävä

Toteuta funktio `strTransform()`, joka suorittaa yksinkertaisen ehdollisen muutoksen merkkijonolle.

Funktio ottaa kolme parametria:

1. Merkkijonon `const char*` tyyppisenä.
2. Predikaattifunktion (funktio-osoittimen) joka ottaa parametrinaan yhden `int` tyyppisen merkin ja palauttaa totuusarvon (`int`).
3. Muutosfunktion (funktio-osoittimen) joka ottaa parametrinaan yhden `int` tyyppisen merkin ja palauttaa muutetun merkin `int` tyyppisenä.

Funktio palauttaa uuden dynaamisesti (`malloc`) varatun merkkijonon.

Funktio muodostaa tämän merkkijonon siten, että niille parametrimerkkijonon merkeille joille predikaattifunktio on tosi, merkkijonoon sijoitetaan muutosfunktion merkille palauttama arvo.

Vastaavasti ne parametrimerkkijonon merkit joille predikaattifunktio on epätosi, sijoitetaan sellaisenaan.

Toteuta lisäksi yksinkertainen `main()` funktio, jossa muunnat `strTransform()` funktiota käyttäen jonkin merkkijonon kaikki pienet kirjaimet isoiksi kirjaimiksi.

- Voit olettaa että `malloc()`-, `calloc()`- ja `realloc()`-funktiot onnistuvat aina.
- Tehtävän funktio-osoittimet ovat yhteensopivia `ctype.h` funktioiden kanssa (minireferenssi).

5. Tehtävä

Selitä **lyhyesti** ja korkeintaan kahdella lauseella mitä nämä funktiot tekevät. Älä kuvaa funktion sisäistä toimintaa, vaan kerro mitä funktio tekee syötteelleen tai millaisen arvon funktio laskee syötteistään.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <assert.h>
4
5  size_t function1(const char* p)
6  {
7    assert(p);
8    size_t n = 0;
9    while (*(p++))
10     n++;
11   return n;
12 }
13
14 char* function2(FILE* in)
15 {
16   assert(in);
17   static char buffer[512];
18   size_t n = 0;
19   while (n < 511)
20   {
21     int ch = fgetc(in);
22     if (ch == EOF || ch == '\n')
23       break;
24     buffer[n++] = ch;
25   }
26   buffer[n] = '\0';
27   return buffer;
28 }
29
30 void function3(int* a, size_t n, int* p1, int* p2)
31 {
32   assert(a && n && p1 && p2);
33   *p1 = *a;
34   *p2 = *a;
35   for (size_t i = 0;i < n;i++)
36   {
37     if (*a < *p1)
38       *p1 = *a;
39     else if (*a > *p2)
40       *p2 = *a;
41     a++;
42   }
43 }
```

6. Tehtävä

Toteuta oheiset kolme funktiota, jotka käsittelevät taulukoita yksinkertaisia `Entry` tietueita.

`Entry` on tietue jossa on kaksi kenttää: nimi `char*` merkkijonona ja yksi `int`-tyyppinen arvo.

```
1  #include <stdio.h>
2  #include <stddef.h>
3
4  struct Entry
5  {
6    char* name; // Tietueen nimi
7    int value;  // Tietueen arvo
8  };
9
10 /* Tulostaa taulukon Entry tietueita annettuun virtaan.
11  * Jokainen tietue tulostetaan omalle rivilleen muodossa:
12  *  nimi : arvo
13  * Parametrit järjestyksessä:
14  *  Virta johon tietueet tulostetaan
15  *  Taulukko Entry tietueita
16  *  Taulukossa olevien tietueiden lukumäärä */
17 void printEntries(FILE* out, const struct Entry* entries, size_t n);
18
19 /* Etsii taulukosta Entry tietueen annetun nimen perusteella.
20  * Palauttaa osoittimen ensimmäiseen tietueeseen jolla on annettu nimi.
21  * Palauttaa NULL mikäli annetun nimistä tietuetta ei löydy taulukosta.
22  * Parametrit järjestyksessä:
23  *  Taulukko Entry tietueita
24  *  Taulukossa olevien tietueiden lukumäärä
25  *  Haettavan tietueen nimi */
26 const struct Entry* findEntry(const struct Entry* entries, size_t n,
27                               const char* search);
28
29 /* Kopioi taulukon Entry tietueita.
30  * Palauttaa luodun kopion.
31  * Taulukko syväkopioidaan, eli esimerkiksi alkuperäisen taulukon
32  *  poistaminen ei saa vaikuttaa kopioon mitenkään.
33  * Parametrit järjestyksessä:
34  *  Taulukko Entry tietueita
35  *  Taulukossa olevien tietueiden lukumäärä */
36 struct Entry* copyEntries(const struct Entry* entries, size_t n);
```

- Voit olettaa että `malloc()`-, `calloc()`- ja `realloc()`-funktiot onnistuvat aina.

C99 minireferenssi

Tentissä tarvittavat tietotyypit ja niiden koot

Tyyppi	Koko tavuina	Lukualue	I/O muotoilumääre
`char`	1	-127 ... 127	%c
`unsigned char`	1	0 ... 255	%c
`int`	4	-2,147,483,647 ... 2,147,483,647	%d
`unsigned int`	4	0 ... 4,294,967,295	%u %x %X
`long`	8	Riittävän laaja ($-2^{63}...2^{63}-1$)	%ld
`size_t`	8	Riittävän laaja ($0...2^{64}-1$)	%zu
`double`	8	Riittävän laaja	%lf
`void*`	8	Kaikki (objekti-)osoitinmuuttujat	%p

Operaattoripresedenssi korkeimmasta matalimpaan ja assosioituvuus

`() [] . → ++ --`	vasemmalta oikealle	postfix ++ ja --
`++ -- + - ! ~ (tyyppi) * & sizeof`	oikealta vasemmalle	prefix ++ ja --, * ja & muistioperaattoreina
`* / %`	vasemmalta oikealle	* aritmeettisena operaattorina
`+ -`	vasemmalta oikealle	
`<< >>`	vasemmalta oikealle	
`< <= > >=`	vasemmalta oikealle	
`== != & ^ | && ||`	vasemmalta oikealle	& loogisena operaattorina
`?:`	oikealta vasemmalle	
`= op=`	oikealta vasemmalle	kaikki sijoitusoperaatiot
`,`	vasemmalta oikealle	pilkku operaattorina

C99 varatut sanat

`auto`	`break`	`case`	`char`	`const`	`continue`	`default`	`do`
`double`	`else`	`enum`	`extern`	`float`	`for`	`goto`	`if`
`inline`	`int`	`long`	`register`	`restrict`	`return`	`short`	`signed`
`sizeof`	`static`	`struct`	`switch`	`typedef`	`union`	`unsigned`	`void`
`volatile`	`while`	`_Bool`	`_Complex`	`_Imaginary`			

Tentissä tarvittavat standardikirjaston funktiot

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, ...);
FILE* fopen(const char* path, const char* mode);        int fclose(FILE* fp);
int getchar(void);                                      int fgetc(FILE *stream);
int putchar(int ch);                                    int fputc(int ch, FILE* stream);
int puts(const char* str);                              int fputs(const char* str, FILE* stream);
char* fgets(char* str, int size, FILE* stream);
```

stdlib.h

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

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);
char *strstr(const char* haystack, const char* needle);     size_t strlen(const char *str);
size_t strxfrm(char* dest, const char* src, size_t n);      void* memset(void* s, int c, size_t n);
```