AS-0.1101 C-ohjelmoinnin peruskurssi
Tentti 27.10.2009, 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.

Kirjoita edes opiskelijanumerosi selkeällä käsialalla tenttipapereihin.

1. Tehtävä

Toteuta ohjelma, joka lukee käyttäjältä 100 liukulukuarvoa. Luettuaan nämä arvot ohjelma kysyy käyttäjältä kaksi liukulukua raja-arvoiksi, joista ensimmäinen kuvaa alarajaa ja toinen ylärajaa. Tämän jälkeen ohjelma tulostaa kaikki luvut jotka ovat näiden annettujen raja-arvojen välillä sekä näiden lukujen keskiarvon. Jaa ohjelmasi järkevästi ainakin kolmeen funktioon. Voit olettaa syötteen olevan aina kelvollista, eikä ohjelmasi tarvitse esittää minkäänlaisia kehotteita käyttäjälle.

2. Tehtävä

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

printf-määre %zu on käytännössä etumerkitön kokonaisluku.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(void)
6  {
7      int array[8] = {1, 2, 4, 8, 16, 32, 64, 128};
8      int* ptr_array[4];
9      int *dyn_array = malloc(sizeof(int) * 8);
10
11     ptr_array[0] = array;
12     ptr_array[1] = array + 7;
13     ptr_array[2] = array + 3;
14     ptr_array[3] = dyn_array;
15
16     printf("A: %zu\n", ptr_array[1] - ptr_array[2]);
17     printf("B: %d\n", ptr_array[1][2] - ptr_array[2][1]);
18
19     for (int* ptr = dyn_array;ptr - dyn_array < 8;ptr++)
20         *ptr = array[ptr - dyn_array] * 2;
21
22     printf("C: %d\n", ptr_array[3][4]);
23     printf("D: %d\n", **ptr_array + dyn_array[0]);
24
25     ptr_array[2][0] = *dyn_array;
26
27     printf("E: %d\n", array[3]);
28
29     free(dyn_array);
30 }
```

3. Tehtävä

Toteuta filterInvalid()-funktio, joka ottaa parametrinaan taulukon `unsigned char` tyyppisiä arvoja ja taulukon koon `size_t` tyyppisenä muuttujana. Funktio "poistaa" taulukosta kaikki sellaiset arvot joissa ei ole parillista määrää 1-bittejä tiivistämällä taulukon siten, että sen alussa ovat kaikki kelvolliset arvot (sellaiset joissa on parillinen määrä 1-bittejä). Funktio palauttaa tiivistetyn taulukon koon, eli monessako alkuperäisen taulukon alkiossa oli parillinen määrä 1-bittejä. Alkioiden alkuperäinen järjestys on säilytettävä.

Esimerkiksi taulukko: 0x05, 0x04, 0x12, 0x31, 0x0F tiivistyy: 0x05, 0x12, 0x0F

```
1 #include <stddef.h>
2 size_t filterInvalid(unsigned char*, size_t);
```

4. Tehtävä

Ohessa on 12 väittämää C-kielestä, joista vain osa pitää paikkansa. Vastaa valitsemiisi kohtiin ja kerro pitääkö väittämä paikkansa vai ei, vastauksia ei tarvitse perustella mitenkään. Tehtävä arvostellaan kohdittain siten, että oikea vastaus +0.5 p, väärä vastaus -0.5p ja ei vastausta 0p. Tehtävän maksimipistemäärä on 5 ja minimipistemäärä 0.

I	`printf("duck\n");` on sama kuin: `fprintf(stdout, "duck\n");`
II	`void*`-tyyppiseen muuttujaan voi sijoittaa funktio-osoittimen
III	Funktionkaltainen makro: `#define PRODUCT(P1, P2) ((P1) * (P2));` on toteutettu väärin
IV	`struct {int i;} array[10]`-taulukon loppukohta voidaan ilmaista oikein `NULL`-arvolla
V	C-kielessä kaksi samantyyppistä tietuetta voi sijoittaa toisiinsa
VI	Ehtolause: `if (10.0 < length < 20.0)` on järkevä
VII	`(0xA4 + 4) >> 2` on 0x54
VIII	`struct s {int first; void* second;}` määrittelee tietueen s
IX	Jos on määritelty `double array[4];` niin `(&array[4] - array);` on 4
X	C-kielessä (0.5 + 1/4) on 0.75
XI	`sizeof`-operaattorilla ei ole mahdollista selvittää staattisen taulukon kokoa
XII	`void (*ptr)(void*) = free;` on laillinen C-kielessä

5. 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 "stdlib.h"
2
3  /* arg and return value is always at least void** */
4  void* function1(void *arg)
5  {
6      void **ptr = arg;
7      size_t s;
8      for (s = 0;ptr[s];s++);
9      void **p = malloc(sizeof (*ptr) * (s + 1));
10     if (!p)
11         return NULL;
12     for (s = 0;ptr[s];s++)
13         p[s] = ptr[s];
14     p[s] = ptr[s];
15     return p;
16 }
17
18 void function2(const char *s, size_t *t)
19 {
20     *t = 0;
21     while (*(s + *t))
22         (*t)++;
23 }
24
25 void function3(char *ptr)
26 {
27     char *s = ptr;
28     while (*s)
29        s++;
30     for (s--;ptr < s;ptr++, s--)
31     {
32         char t = *ptr;
33         *ptr = *s;
34         *s = t;
35     }
36 }
```

6. Tehtävä

Toteuta funktio findFunctionPointer()-joka ottaa kaksi parametria. Ensimmäinen parametri on taulukko funktio-osoittimia ja toinen parametri on funktio-osoitin, joka ottaa parametrinaan yhden taulukon alkion ja palauttaa kokonaislukuna esitetyn totuusarvon. Funktio käy läpi funktio-osoitintaulukkoa alusta alkaen ja palauttaa ensimmäisen funktio-osoittimen jolle toisena parametrina annettu funktio palauttaa toden arvon. Funktio-osoitintaulukon loppua ilmaisee NULL arvo. Mikäli etsittävää funktiota ei löydy taulukosta, funktio palauttaa NULL arvon. Kaikki taulukon funktiot ovat tyyppiä:

```
1 double (*)(int, int, char);
```

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 fclose(FILE *fp);
FILE *fopen(const char *path, const char *mode);
int fscanf(FILE *stream, const char* format, ...);
int sscanf(const char *str, const char* format, ...);
int fgetc(FILE *stream);
char *fgets(char *str, int size, 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* calloc(size_t nmemb, 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);

char *strstr(const char *haystack, const char *needle);

size_t strxfrm(char *dest, const char *src, size_t n);

void* memset(void *s, int c, size_t n);
```