AS-0.1101 C-ohjelmoinnin peruskurssi / Tentti 09.05.2007 / Aki Hiisilä

Vastaa viiteen kysymykseen! Tentin arvosteluasteikko on 0 – 30 pistettä. Kaikkien kysymysten painoarvo on sama (6 pistettä/tehtävä). Ohjelmointitehtät tulee kirjoittaa C-kielellä hyvää ohjelmointityyliä noudattaen.

Palauta kolme konseptia, siten että ensimmäisellä konseptilla on vastaukset tehtäviin 1 ja 2, toisella konseptilla vastaukset tehtäviin 3 ja 4, sekä kolmannella konseptilla vastaukset tehtäviin 5 ja 6.

Tehtävä 1

Mitä alla oleva ohjelma tulostaa?

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

int main(void)
{
  int **m = NULL;

  m = (int **)malloc(3 * sizeof(int *));
  m[0] = (int *)malloc(9 * sizeof(int));

  for(int i = 0; i < 9; i++){
    m[0][i] = (i + 1) * 10;
  }

  m[1] = m[0] + 2;
  m[2] = m[1] - 1;

  printf("a: %d\n", m[0][0]);
  printf("b: %d\n", m[1] - m[0]);
  printf("c: %d\n", m[2][2]);
  printf("d: %d\n", *(m[0]));
  printf("e: %d\n", (*((*(m + 0)) + 1)) + 2);
  printf("f: %d\n", (m[0] + (m[1] - m[2]))[3]);

  free(m[0]);
  free(m);

  return EXIT_SUCCESS;
}
```

Tehtävä 2

Kirjoita ohjelma, joka laskee kolmion piirin. Ohjelmalle annetaan syötteenä kolme koordinaattipistettä muodossa **x;y**. Nämä kolme pistettä ovat kolmion kulmapisteet.

Esimerkki ohjelman toiminnasta:

```
Point 1 (in format x;y)
1.2;3.4
Point 2 (in format x;y)
-1;-2
Point 3 (in format x;y)
0;0
Perimeter: 11.67
```

Huomautus 1: Tehtävä pitää funktioilla jakaa sopivasti osiin ja tietojen välitys on tehtävä parametreilla. Globaaleja muuttujia ei saa käyttää.

Huomautus 2: Syöttötiedoille ei tarvitse tehdä mitään järkevyystarkastuksia. Voidaan siis olettaa, että koordinaatit annetaan aina muodossa **x;y**.

Huomautus 3: Kolmion piiri on sen sivujen pituuksien summa

Tehtävä 3

Alla oleva C-kielinen ohjelma lukee positiivisia mittausarvoja kunnes sille annetaan syötteenä negatiivinen arvo. Tämän jälkeen ohjelma laskee mittausarvojen keskiarvon. Ohjelmassa on joitakin virheitä. Etsi ja perustele virhekohdat, sekä kerro miten korjaisit virheet.

```
 1: #include <stdlib.h>
 2: #include <stdio.h>
 3:
 4: double average(int *array, size_t size);
 5:
 6: int main(void)
 7: {
 8:   int *measurements;
 9:   size_t len;
10:   int meas = 1;
11:
12:   while(meas > 0){
13:
14:     if(len == 0){
15:       measurements = (int *)malloc(sizeof(int));
16:     }
17:     else{
18:       realloc(measurements, (len + 1) * sizeof(int));
19:     }
20:
21:     scanf("%d", &meas);
22:     len++;
23:     measurements[len] = meas;
24:   }
25:
26:   printf("Average: %lf\n", average(measurements, len));
27:   free(measurements);
28:   return EXIT_SUCCESS;
29: }
30:
31: double average(int *array, size_t len)
32: {
33:   int sum;
34:   double avg;
35:
36:   for(size_t i = 0; i < len; i++){
37:     sum += array[i];
38:   }
39:   avg = sum / len;
40:   return avg;
41: }
```

Tehtävä 4

Projektissa käsitellään erilaisia asioita. Asian käsittelyssä tarvittavat asian tiedot sisältyvät tietotyyppiin **TItem**. Koska asioita ei pystytä käsittelemään siinä tahdissa kuin niitä tulee käsiteltäväksi, ne laitetaan asioiden jonoon. Tätä varten on olemassa valmiina abstrakti tietotyyppi asiajono, joka muodostuu tietotyypistä **TQueue** ja sen käsittelyyn tarkoitetuista funktioista:

```
/* Alustaa jonon */
TQueue queueConstruct();

/* Laittaa asian jonoon */
void enqueue(TQueue *queue, TItem item);

/* Ottaa asian jonosta */
TItem dequeue(TQueue *queue);

/* Testaa onko jono tyhjä (palauttaa 1) vai ei (palauttaa 0) */
int isEmpty(const TQueue *queue);
```

Nyt kuitenkin halutaan priorisoida asioiden käsittelyä. Tätä varten tarvitaan asioiden prioriteettijono, jossa asialle annetaan prioriteetti silloin, kun se laitetaan jonoon. Mahdolliset prioriteetit ovat 0, 1, 2, 3, ..., n - 1, missä n on prioriteettijonon alustuksen yhteydessä annettava luku. Asioita prioriteettijonosta pois otettaessa ne tulevat prioriteetin mukaan kuten prioriteettijonosta kuuluu (eli korkeamman prioriteetin asia aina ennen matalamman prioriteetin asiaa).

Eräs kätevä tapa toteuttaa prioriteettijono on käyttää n (n on prioriteettien määrä) kappaletta tavallisia jonoja ja laittaa asiat prioriteetin mukaan sisäisesti omiin jonoihinsa. Ulos otettaessa asia otetaan aina korkeimman prioriteetin jonosta, jossa asioita on.

Tehtävänä on **toteuttaa asioiden prioriteettijono** ylläkuvatulla periaatteella eli **määritellä tietotyyppi TPriQueue ja toteuttaa seuraavat funktiot sen käyttöön**:

```
/* Alustaa prioriteettijonon (n on prioriteettien määrä) */
TPriQueue constructPriQueue(int n);

/* Laittaa asian prioriteettijonoon */
void pqEnqueue(TPriQueue *pq, TItem item, int priority);

/* Ottaa asian prioriteettijonosta */
Titem pqDequeue(TPriQueue *pq);

/* Testaa onko prioriteettijono tyhjä (palauttaa 1) vai ei (palauttaa
0) */
int pqIsEmpty(const TPriQueue *queue);
```

Huomautus: Tässä prioriteetti on sitä korkeampi, mitä suurempi arvo sillä on.

Tehtävä 5

Mitä alla oleva ohjelma tulostaa, kun oletetaan, että yksi tavu on kahdeksan bittiä?

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

typedef unsigned char (*opfunc)(unsigned char, unsigned char);

unsigned char f1(unsigned char a, unsigned char b)
{
  return (a & b);
}
unsigned char f2(unsigned char a, unsigned char b)
{
  return (a | b);
}
unsigned char f3(unsigned char n, unsigned char byte)
{
  printf("%d: ", n);

  for(unsigned char mask = 0x80; mask; mask >>= 1){
    printf("%c", (byte & mask) ? '1' : '0');
  }
  printf("\n");
  return 1;
}

int main(void)
{
  unsigned char b1 = 0xf1;
  unsigned char b2 = 0x66;
  opfunc fa[] = {f1, f2, f3};

  f3(1, f1(b1, b2));

  f3(2, f2(b1, b2));

  f3(3, f1(b1 << 1, b2 >> 1) ^ f2(b1 >> 1, b2 << 1));

  fa[2](4, fa[0](b1, b2) && fa[1](b1, b2));

  fa[2](6, fa[0](b1, b2) || fa[2](5, fa[0](fa[0](b1, b2),
                                           fa[1](b1, b2))));

  b1 = f1(f2(f3(b1, b2), f2(b1, b2)), f1(b1, b2));

  return EXIT_SUCCESS;
}
```

Tehtävä 6

Toteuta seuraavat merkkijonoja käsittelevät funktiot:

```
/** Kääntää merkkijonon toisinpäin (esim. ABC -> CBA).
 *  @param str Merkkijono joka käännetään.
 */
void reverse(char *str);

/** Lisää (liittää) merkkijonon n merkkijonon str eteen.
 *  @param str Osoitin merkkijonoon.
 *  @param n Merkkijono, joka lisätään merkkijonon str eteen.
 *  @return 0 jos onnistui, 1 virhetilanteissa.
 */
int addToFront(char **str, const char *n);

/** Järjestää merkkijonon str merkit aakkosjärjestykseen.
 *  Oletetaan, että merkkijonon kaikki merkit ovat pieniä kirjaimia
 *  (a, b, c, ..., z)
 *  @param str Merkkijono jonka merkit järjestetään.
 */
void sortChars(char *str);
```

Huomautus 1: Funktioiden pitää pitää huoli siitä, että merkkijonolle on varattuna aina täsmälleen oikea määrä muistia.

Huomautus 1: Voit olettaa, että aakkoset ovat merkistössä peräkkäin