RAČUNALNIŠTVO I
1998/99

Kolokvij / rešitve

4. sušca 1999

Kolokvij sestoji iz devetih nalog, katerih skupno število točk je **100**. Vrednosti posameznih nalog so naznačne v spodnji tabeli.

1. naloga: definicija funkcije	__ /	6
2. naloga: parametri funkcije	__ /	12
3. naloga: Črni Jaka	__ /	12
4. naloga: nepridiprav pri kartah	__ /	9
5. naloga: izpisovanje kart	__ /	5
6. naloga: in še enkrat izpisovanje kart	__ /	15
7. naloga: razvrščanje kart	__ /	14
8. naloga: niso samo karte skrivnostne, ampak tudi programi	__ /	7
9. naloga: pa poglejmo, kdo je zmagal pri kartah	__ /	20
SKUPNO	__ /	100

Čas pisanja je 1 ura in pol (90 minut).
Na kolokviju je dovoljena uporaba zapiskov a nikakor „dobrososedske“ pomoči.
Predno pričnete pisati, spodaj vpišite svoje ime in priimek ter se podpišite.

Ime in Priimek	Podpis
(tiskano)	(študent)

Veliko uspeha!

1. Definicija funkcije sestoji iz glave, pogodbe in telesa.

   NALOGA:

   (a) Iz česa sestoji glava definicije funkcije? Napišite primer glave.

   - *tip rezultata*
   - *ime funkcije*
   - *parametrov*

```
int VsotaVektorja(int vektor[100])
```

   (b) Iz česa sestoji pogodba funkcije? Na kratko opišite pomen posameznih členov pogodbe.

   - *opis: opisuje kaj funnkcija dela*
   - *parametri: kakšne parametre funkcija potrebuje, njihovo vlogo in kako so prenešeni (oziroma, ali so vhodni ali izhodni)*
   - *rezultat: kakšen je rezultat in kako se izračuna*
   - *predpogoji: kakšne pogoje mora zagotoviti klicatelj, da funkcija pravilno deluje*
   - *popogoji: kakšne pogoje zagotavlja funkcija ob koncu izvajanja, če je klicatelj zagotovil predpogoje*
   - *okolje: morebitno dodatno okolje, ki ga funkcija uporablja*

2. Vsaka funkcija ima *vhodne* in *izhodne* parametre. Sicer prenašamo parametre v funkcijo bodisi po referenci (naslovu) ali po vrednosti.

   NALOGA:

   (a) Katera je najbolj bistvena razlika med prenosom parametrov po referenci in po vrednosti?
   Če parametru v funkciji spremenimo vrednost, se v klicočem programu pri prenosu po referenci (naslovu) ta sprememba odraža, medtem ko se pri prenosu po vrednosti ne.

   (b) Kakšen prenos se uporablja za vhodne parametre?
   Načeloma po vrednosti, lahko po referenci.

   (c) Kakšen prenos se uporablja za izhodne parametre?
   Po referenci ali kot rezultat.

   (d) Za vsakega od parametrov v spodnji glavi funkcije določite ali je prenešen po vrednosti ali po referenci.

   ```
   void NekajNaredi (int par1, int par2[100]);
   ```

   - ***par1*** – *po vrednosti*
   - ***par2*** – *po referenci*

3. Pri igri s kartami imenovani ,,Črni Jaka" lahko vzemamo karte z vrha kupa, dokler njihova vsota ne preseže vrednosti 21. Zmaga tisti igralec, katerega vsota je najbližja vrednosti 21 a je ne presega. V tej nalogi malce spremenimo pravila in vlečemo karte dokler njihova vsota ne preseže vrednosti 21. Kot rezultat funkcija vrne razliko med vsoto izvlečenih kart in vrednsotjo 21.

   NALOGA: Spodaj je napisano telo funkcije, ki skrbi za vlečenje kart. V telesu funkcije je napisano le *kaj* počne, vi pa dopišite *kako* to naredi – t.j. napišite kodo funkcije. Pri tem upoštevajte, da je na vrhu kupa kart `karte[0]`, naslednja karta je `karta[1]` itd. Karta `i` ima vrednost `karta[i]`.

```
int VleciKarte(int karte[52]) {
  int karta;
  int vsota;

                   /* nastavi zacetno vsoto na 0 */
  vsota= 0;
                   /* dokler vsota kart ne preseze 21 */
  for (karta= 0; vsota <= 21; karta++)
                      /* izvleci karto */
                      /* in pristej njeno vrednost vsoti */
    vsota+= karte[karta];
                   /* vrni vsoto zmanjsano za 21 */
  return (vsota - 21);
} /* VleciKarte */
```

4. Sosedov Miha bi rad malce goljufal pri igri „Črni Jaka“. Nepridiprav jo kani goljufati tako, da bo sem ter tja pokukal na poljubno mesto v kup kart, ki je predstavljen kot tabela 52-ih celih števil (glej nalogo 3), ter se tako lažje odločil, ali naj še vleče karte ali ne. Zato potrebuje funkcijo (podprogram) `Kukaj`, ki bo prebrala s tipkovnice mesto, na katero bo Miha pokukal in bo nato vrnila vrednost karte na tem mestu.

   NALOGA:

   (a) Kakšne vhodne in kakšne izhodne parametre ima funkcija `Kukaj`?

   - *vhodni: karte (tabela)*
   - *izhodni: vrednost_karte (pokukane)*

   (b) Napišite glavo funkcije `Kukaj`.

   int Kukaj(int karte[52]

5. Pri naši igri s kartami potrebujemo tudi funkcijo, ki bo smiselno izpisovala karte, glede na njihovo vrednost. Zato definiramo naslednjo funkcijo:

```
void IzpisiKarto(int karta) {
/* Desc:   Funkcija izpise na stdout opis slike na karti, ki ima dano
           vrednost.
   Parms:  karta - vrednost karte (po vrednosti)
   Result: -
   Pre:    -
   Post:   opis karte je izpisan na stdout
*/

  if (karta == 14)      printf("as");
  else if (karta == 13) printf("kralj");
  else if (karta == 12) printf("dama");
  else if (karta == 11) printf("fant");
  else if (karta == 10) printf("desetica");
  else if (karta == 9)  printf("devetica");
  else if (karta == 8)  printf("osmica");
  else if (karta == 7)  printf("sedmica");
  else if (karta == 6)  printf("sestica");
  else if (karta == 5)  printf("petica");
  else if (karta == 4)  printf("stirica");
  else if (karta == 3)  printf("trojka");
  else                  printf("dvojka");

} /* IzpisiKarto */
```

NALOGA: Prepišite zgornjo funkcijo z uporabo stavka `switch`.

```
void IzpisiKarto(int karta) {
  switch (karta) {
  case 14: printf("as");       break;
  case 13: printf("kralj");    break;
  case 12: printf("dama");     break;
  case 11: printf("fant");     break;
  case 10: printf("desetica"); break;
  case 9:  printf("devetica"); break;
  case 8:  printf("osmica");   break;
  case 7:  printf("sedmica");  break;
  case 6:  printf("sestica");  break;
  case 5:  printf("petica");   break;
  case 4:  printf("stirica");  break;
  case 3:  printf("trojka");   break;
  default: printf("dvojka");
  }
} /* IzpisiKarto */
```

6. Nepridiprav Miha želi pri svojem nečednem poslu uporabiti našo funkcijo `IzpisiKarto` iz naloge 5. Toda Miha bi rad najprej preveril kako program deluje. Pričakuje, da bo naslednji testni program:

```
int main(void) {

  int karta;

  for (karta= 2; karta < 20; karta++) {
    IzpisiKarto(karta);
    printf("\n");
  }

  return 1;
} /* main */
```

izpisal:

```
dvojka
trojka
stirica
petica
sestica
sedmica
osmica
devetica
desetica
fant
dama
kralj
as
dvojka
trojka
stirica
petica
sestica
```

saj meni, da se najprej izpise prvi niz kart, nato drugi in tako naprej. Toda izpiše se:

```
dvojka
trojka
stirica
petica
sestica
sedmica
osmica
devetica
desetica
```

```
fant
dama
kralj
as
dvojka
dvojka
dvojka
dvojka
dvojka
```

NALOGA:

(a) Zakaj, opišite, je prišlo do razlike med obema izpisoma?

Ker se v funkciji `IzpisiKarto` izpiše `dvojka` za vsako vrednost, ki je manjša od 3 in večja od 14.

(b) Kdo je ,,kriv" za napako (obkrožite pravilen odgovor):

- $\boxed{\text{mi, ki smo programirali IzpisiKarto}}$;
- Miha, ki je uporabil našo funkcijo.

(c) Kako popraviti napako – odpraviti razliko? Navedite dve bistveno različni možnosti.

- *V funkcijo `IzpisiKarto` dodamo predpogoj:*

$$\texttt{Pre:} 2 \leq \texttt{karta} \leq 14$$

- *V funkciji nadomestimo zadnji `else` stavek s stavkom:*

```
else if (karta == 2) printf("dvojka");
```

7. Če imamo n kart jih lahko razvrstimo v vrsto na $n!$ načinov, kjer je funkcija $n!$ definirana kot $1 \cdot 2 \cdots n$.

   NALOGA: Napišite funkcijo `Fakulteta`, ki bo izračunala $n!$ s pomočjo *rekurzije*.

```
int Fakulteta(int n) {

  if (n == 0) return 1;
  else        return n * Fakulteta(n-1);

} /* Fakulteta */
```

8. Imamo program:

```
#include <stdio.h>

int Nekaj (int a, int b) {

  if (b == 0)  return a;
  else         return Nekaj(b, a % b);

} /* Nekaj */

int main (void) {

  printf("%d \n", Nekaj(77, 22));
  return 0;

} /* main */
```

NALOGA: Kaj izpiše program?

11

DODATNA NALOGA: Znate opisati bolj splošno, kaj dela program?

Program je Evklidov algoritem za iskanje največjega skupnega deljitelja števil `a` in `b`.

9. Igra kart je zaključena in čas je, da vidimo, kdo je največkrat zmagal. Za to bomo uporabili *tračni diagram.* Tračni diagram sestoji iz trakov, katerih dolžina je sorazmerna velikosti količino, ki jo predstavljajo. Včasih pa so lahko količine, ki jih želimo predstaviti s tračnim diagramom prevelike in moramo omejiti dolžino trakov. V naslednjem primeru so v prvem stolpcu zapisane velikosti količin, ki jim sledijo tračni diagrami, ki so omejeni na 7 znakov. Pri tem je trak, ki smo ga morali „odrezati“, označen z znakom `X` na zadnjem mestu.

```
4 0000
9 000000X
7 0000000
0
2 00
```

NALOGA: Napišite podprogram `TracniDiagram`, ki bo izrisal tašen tračni diagram.

```
void TracniDiagram (int stKolicin, int kolicine[MAX_KOLICIN],
                    int najvecjaDolzina) {
/* Desc:    Funkcija izrise tracni diagram za stKolicin kolicin, ki so
            shranjene v tabeli kolicine. Pri tem uposteva, da je
            najdaljsi trak diagrama lahko dolg najvecjaDolzina znakov.
            i-ta vrstica diagrama predstavlja i-to kolicino iz tabele
            kolicin kolicine (0 <= i < stKolicin). Vrstica se pricne s
            stevilsko vrednostjo i-te kolicine, ki jo izpisana na 5
            mest natancno. Sledi presledek in nato trak sestojec iz
            znakov 0 in dolzine enake velikosti i-te kolicine. Ce je
            ta kolicina vecja od najvecjaDolzina, je trak dolg
            najvecjaKolicina znakov, od katerih je zadnji znak X.
   Parms:   stKolicin - stevilo kolicin v tabeli kolicine (po vrednosti)
            kolicine - tabela kolicin (po referenci)
            najvecjaDolzina - najvecja dolzina traku (po vrednosti)
   Result:  -
   Pre:     (0 <= stKolicin < MAX_KOLICIN) AND
            (0 < najvecjaKolicina) AND
            (FOR 0 <= i < stKolicin: kolicine[i] >= 0)
   Post:    (tracni diagram je izpisan) AND
            (kolicine' = kolicine)
*/
```

```
  int igralec;          /* zaporedni igralec */
  int stolpec;          /* izpisani stolpec */

                                          /* za vsakega igralca */
  for (igralec= 0; igralec < stKolicin; igralec++) {
    printf("%5d ", kolicine[igralec]);    /* stevilo tock */
                                          /* in se trak */
    for (stolpec= 0;
         (stolpec < najvecjaDolzina) &
         (stolpec <= kolicine[igralec]);
         stolpec++)
      printf("O");
    if (stolpec <=  kolicine[igralec]) { /* cel trak narisan? */
      if (stolpec < kolicine[igralec])   /* ups, trak predolg */
        printf("X");
      else
        printf("O");
    }
  }

} /* TracniDiagram */
```