Programiranje 2

Bibliotečke funkcije bsearch i qsort

Milena Vujošević Janičić
Jelena Graovac

www.matf.bg.ac.rs/~milena
www.matf.bg.ac.rs/~jgraovac

Pregled

Pregled

Pokazivači na funkcije (obnavljanje)

```
void povecaj1(int a[], int n, int b[]) {
 int i;
 for (i = 0; i < n; i++)
   b[i] = a[i] + 1;
}
void pomnozi2(int a[], int n, int b[]) {
 int i;
 for (i = 0; i < n; i++)
   b[i] = a[i] * 2;
}
void parni0(int a[], int n, int b[]) {
 int i;
 for (i = 0; i < n; i++)
   b[i] = a[i] % 2 == 0 ? 0 : a[i];
}
```

Pokazivači na funkcije (obnavljanje)

```
void transformacija(int a[], int n, int b[], int (*f) (int)) {
 int i;
 for (i = 0; i < n; i++)
   b[i] = (*f)(a[i]);
}
int uvecaj1(int x) { return x + 1; }
int pomnozi2(int x) { return 2 * x; }
int parni0(int x) { return x % 2 == 0 ? 0 : x; }

/*poziv funkcije map*/
transformacija(a, N, b, &uvecaj1);
transformacija(a, N, b, &pomnozi2);
transformacija(a, N, b, &parni0);
```

Pokazivači na funkcije (obnavljanje)

- Pokazivači na funkcije se razlikuju po tipu funkcije na koje ukazuju (po tipovima argumenata i tipu povratne vrednosti).
- Deklaracija promenljive tipa pokazivača na funkciju se vrši tako što se ime promenljive kojem prethodi karakter * navede u zagradama kojima prethodi tip povratne vrednosti funkcije, a za kojima sledi lista tipova parametara funkcije.

```
double *a(double, int);   /*Funkcija koja vraca pokazivac na
                            double i prima dva argumenta
                            double i int*/
double (*b)(double, int); /*Pokazivac na funkciju koja vraca
                            double i prima dva argumenta
                            double i int*/
```

Pokazivači na funkcije (obnavljanje)

Moguće je kreirati i nizove pokazivača na funkcije. Ovi nizovi se mogu i incijalizovati (na uobičajeni način). U primeru

```
int (*nf[3]) (int) = {&povecaj1, &pomnozi2, &parni0};
```

nf predstavlja niz od 3 pokazivača na funkcije koje vraćaju int, i primaju argument tipa int.
Funkcije čije se adrese nalaze u nizu se mogu direktno i pozvati. Na primer, naredni kôd ispisuje vrednost 4:

```
printf("%d", (*nf[0])(3));
```

Pregled

Generička funkcija provere sortiranosti niza

- Ako je zadata relacija poretka (ili strogog poretka), onda se može proveriti da li je niz uređen (ili sortiran) u skladu sa tom relacijom. Na primer, naredna funkcija proverava da li je niz tipa int uređen u skladu sa relacijom $\leq$ (parametar n daje broj elemenata niza i takav je obično tipa `size_t`):

```
int sortiran(const int a[], size_t n){
  int i;
  for (i = 0; i < n - 1; i++)
    if (!(a[i] <= a[i + 1]))
      return 0;
  return 1;
}
```

Generička funkcija provere sortiranosti niza

- Niz nekog brojevnog tipa, dakle, može biti uređen na različite načine.
- Niz tipa int može, na primer, biti uređen i neopadajuće po zbiru svojih cifara.
- Sledeća funkcija proverava da li je niz uređen na takav način (podrazumeva se da funkcija `int zbir_cifara(int)` vraća zbir cifara svog parametra)

```
int sortiran(const int a[], size_t n){
  int i;
  for (i = 0; i < n - 1; i++)
    if (!(zbir_cifara(a[i]) <= zbir_cifara(a[i + 1])))
      return 0;
  return 1;
}
```

Generička funkcija provere sortiranosti niza

- Primetimo da su prethodne dve funkcije veoma slične i razlikuju se samo u liniji u kojoj se proverava da li su dva susedna elementa u ispravnom poretku. Te provere mogu se izdvojiti u funkcije, te se pokazivač na njih može koristiti kao dodatni argument funkcije sortiran.

```
int u_poretku1(int x, int y){
return (x <= y);
}

int u_poretku2(int x, int y){
return (x >= y);
}

int u_poretku3(int x, int y){
return (zbir_cifara(x) <= zbir_cifara(y));
}
```

Generička funkcija provere sortiranosti niza

```
int sortiran(const int a[], size_t n, int (*p)(int, int)){
  int i;
  for (i = 0; i < n-1; i++)
    if (!p(a[i], a[i+1]))
      return 0;
  return 1;
}
```

Provera da li je niz sortiran na jedan ili na drugi način:

```
sortiran(a, 10, u_poretku1)
sortiran(a, 10, u_poretku2)
```

Generička funkcija provere sortiranosti niza

- Štaviše, forma funkcije sortiran može da se uopšti ne samo za bilo koju vrstu poređenja, nego i na bilo koji tip niza.
- Zato će parametar koji prima niz da bude tipa `void*`. Pošto takav parametar nosi samo informaciju o adresi početka niza, a ne i o lokacijama pojedinačnih elemenata, biće potreban i parametar koji predstavlja veličinu jednog elementa niza (u bajtovima).

```
int sortiran(const void* a, size_t n, size_t velicina, int (*p)(const void *x, const void *y)){
  int i;
  for (i = 0; i < n-1; i++)
    if (!p(a+i*velicina,a+(i+1)*velicina))
      return 0;
  return 1;
}
```

Generička funkcija provere sortiranosti niza

- U pozivu `p(a+i*velicina,a+(i+1)*velicina)`, funkciji p šalju se adrese i-tog i i+1-tog elementa niza i ona vrši njihovo poređenje. Ova funkcija p ima generički prototip (da bi se uklopila u generičku formu funkcije sortiran), ali je njena definicija nužno prilagođena jednom konkretnom tipu. Na primer, za tip int, ona može biti definisana na sledeći način.

```
int u_poretku_int(const void *x, const void *y){
return (*(int*)x <= *(int*)y);
}
```

- U navedenom kodu, podrazumeva se da pokazivači x i y pokazuju na neke elemente niza koji je tipa int. Ovi pokazivači se, međutim, ne mogu derefencerirati jer je njihov tip `const void *`. Potrebno ih je najpre kastovati u tip `int *`, a tek onda derefencirati (i obraditi vrednosti tipa int na koje ukazuju).

Generička funkcija provere sortiranosti niza

- Analogno se može definisati funkcija za poređenje niski koja vraća 1 ako je prva niska manja ili jednaka od druge:

```
int u_poretku_niske(const void *pa, const void *pb){
return (strcmp(*(char**)pa, *(char**)pb) <= 0);
}
```

Generička funkcija provere sortiranosti niza

```
int main(){
int b1[] = { 1, 2, 3, 4, 5 };
int b2[] = { 3, 2, 3, 4, 5 };
char* b3[] = { "ab", "da", "za" };
if (sortiran(b1, sizeof(b1)/sizeof(b1[0]), sizeof(b1[0]), u_poretku_int))
  printf("Niz b1 je sortiran.\n");
else
  printf("Niz b1 nije sortiran.\n");
if (sortiran(b2, sizeof(b2)/sizeof(b2[0]), sizeof(b2[0]), u_poretku_int))
  printf("Niz b2 je sortiran.\n");
else
  printf("Niz b2 nije sortiran.\n");
if (sortiran(b3, sizeof(b3)/sizeof(b3[0]), sizeof(b3[0]), u_poretku_niske))
  printf("Niz b3 je sortiran.\n");
else
  printf("Niz b3 nije sortiran.\n");
return 0;
}
```

Pregled

Bibliotečke funkcije bsearch i qsort

- Algoritmi pretrage (linerana ili binarna) pretražuju niz po nekom ključu pretrage, vršeći poređenje elemenata niza sa datim ključem.
- Pretraga se uvek vrši po istom algoritmu, razlika je samo u tipu niza koji se pretražuje i u načinu poređenja elemenata.
- Algoritam sortiranja sortira niz vršeći poređenje elemenata na neki način koji zavisi od tipa elemenata u nizu, i razmenu elemenata niza.

Bibliotečke funkcije bsearch i qsort

- Poređenje u oba slučaja se može izdvojiti u zasebnu funkciju
- Može se definisati algoritam tako da može da se koristi za proizvoljne tipove podataka.
- Zavaljujući tome, postoje bibliotečke funkcije bsearch i qsort koje se mogu primeniti za pretragu/sortiranje proizvoljnih nizova.

Funkcija bsearch

- Funkcija `bsearch` vrši binarnu pretragu niza za koji se podrazumeva da je sortiran

```
void *bsearch(const void *x, void *array, int n,
              int s, int (*comp)(const void *, const void *));
```

- Funkcija `bsearch` vraća adresu elementa u nizu koji je jednak traženom elementu, ili NULL ukoliko element nije pronađen.
- Prvi argument je pokazivač na ključ pretrage.
- Drugi argument je adresa početka niza.
- Četvrti argument je veličina jednog elementa u nizu.

Funkcija bsearch

- Poslednji argument je pokazivač na funkciju poređenja koja definiše na koji način se vrši pretraživanje po ključu.
- Funkcija poređenja mora da ima naredni potpis:

```
int uporedi(const void* key, const void* pelem)
```

- Prvi argument ove funkcije je ključ pretrage, a drugi je element niza.

Funkcija bsearch

- Povratna vrednost funkcije poređenja je
  - 0 — ukoliko su elementi jednaki,
  - vrednost manja od nule — ukoliko je ključ pretage manji od elementa u nizu
  - vrednost veća od nule — ukoliko je ključ pretrage veći od elementa u nizu sa kojim se vrši poređenje.
- Da bi se koristila funkcija `bsearch`, neophodno je definisati funkciju poređenja.

Funkcija bsearch

Primer poziv-a funkcije bsearch:

```
int niz[] = {1, 3, 5, 7, 8, 9, 13, 16, 18};
int* p = NULL;
int n = sizeof(niz) / sizeof(int);
int x = 16;
p = (int*)bsearch(&x, niz, n, sizeof(int), &uporedi);
if(p==NULL) printf("Broj nije pronadjen!\n");
else printf("Broj je pronadjen na poziciji %d.\n", p-niz);
```

Funkcija bsearch

- U slučaju pretrage niza celih brojeva, funkcija poređenja se definiše na sledeći način:

```
int uporedi(const void* key, const void* pelem) {
  int kljuc = *(int *)key;
  int element_niza = *(int *)pelem;

  if(kljuc < element_niza) return -1;
  else if(kljuc > element_niza) return 1;
  else return 0;

  /* return kljuc - element_niza; */
}

/* skracen zapis */
int uporedi(const void* key, const void* pelem) {
  return *(int *)key - *(int *)pelem;
}
```

Funkcija qsort

- Funkcija qsort vrši sortiranje niza algoritmom brzog sortiranja

```
void qsort(void *array, int n,
           int s, int (*comp)(const void *, const void *));
```

- Funkcija nema povratnu vrednost
- Prvi argument je adresa početka niza.
- Drugi argument je veličina niza.
- Treći argument je veličina jednog elementa u nizu.
- Četvrti argument je pokazivač na funkciju poređenja.

Funkcija qsort

- Poslednji argument je pokazivač na funkciju poređenja koja definiše na koji način se vrši poređenje elemenata u nizu.
- Funkcija poređenja mora da ima naredni potpis:

```
int uporedi(const void* pa, const void* pb)
```

- Oba argumenta ove funkcije su elementi niza!

Funkcija qsort

- Povratna vrednost funkcije poređenja je
  - 0 — ukoliko su elementi jednaki,
  - vrednost manja od nule — ukoliko je prvi argument manji od drugog,
  - vrednost veća od nule — ukoliko je prvi argument veći od drugog.
- Da bi se koristila funkcija `qsort`, neophodno je definisati funkciju poređenja.

Funkcija qsort

Na primer, upotreba funkcije qsort za sortiranje niza celih brojeva:

```
int niz[] = {1, 5, 2, 4, 3, 11, 12, 9, 6};
int n = sizeof(niz) / sizeof(int);
qsort(niz, n, sizeof(int), &uporedi);
```

Niz može da bude statički/dinamički alociran, elementi niza mogu da su inicijalizovani ili da se unose sa standardnog ulaza/iz datoteke...
Funkcija uporedi:

```
int uporedi(const void* pa, const void* pb) {
  return *(int *)pa - *(int *)pb;
}
```

Funkcija qsort

Može se sortirati niska karaktera. Na primer:

```
char niska[]="Informatika";
qsort(niska,sizeof(niska)/sizeof(char),sizeof(char),&cmp);
```

Za nisku karaktera, funkcija poređenja može se definisati na sledeći način:

```
int cmp(const void* pa, const void* pb) {
  return *(char *)pa - *(char *)pb;
}
```

Ukoliko želimo opadajuće sortiran niz:

```
return *(char *)pb - *(char *)pa;
```

Funkcija qsort

Sortiranje niza pokazivača (nacrtati memoriju!):

```
char* reci[] = {"do", "while", "if", "switch",
                "break", "continue"};
int n = sizeof(reci)/sizeof(char*);
qsort(reci, n, sizeof(char*), &uporedi);
```

Funkcija uporedi za leksikografsko sortiranje:

```
int uporedi(const void* pa, const void* pb) {
  /*Element niza je pokazivac na nisku.
  Funkciji se prosledjuje adresa jednog elementa niza,
  dakle adresa pokazivaca na nisku.*/
  char* s1 = *(char**)pa;
  char* s2 = *(char**)pb;
  return strcmp(s1,s2);
}
```

Funkcija qsort

Sortiranje niza pokazivača:

```
char* reci[] = {"do", "while", "if", "switch",
                "break", "continue"};
int n = sizeof(reci)/sizeof(char*);
qsort(reci, n, sizeof(char*), &uporedi);
```

Funkcija uporedi za sortiranje po dužini reči:

```
int uporedi(const void* pa, const void* pb) {
  /*Element niza je pokazivac na nisku.
  Funkciji se prosledjuje adresa jednog elementa niza,
  dakle adresa pokazivaca na nisku.*/
  char* s1 = *(char**)pa;
  char* s2 = *(char**)pb;
  return strlen(s1) - strlen(s2);
}
```

Funkcija qsort

Funkcija uporedi za sortiranje po dužini, ako su iste dužine, onda leksikografski:

```
int uporedi(const void* pa, const void* pb) {
  char* s1 = *(char**)pa;
  char* s2 = *(char**)pb;
  int n1 = strlen(s1);
  int n2 = strlen(s2);
  if(n1==n2) return strcmp(s1,s2);
  else return n1-n2;
}
```

Funkcija qsort

- Ukoliko imamo niz nizova, na primer

```
char niz_nizova[][10]={"do", "while", "if", "switch",
                       "break", "continue"};
```

  Važe drugačija pravila, nacrtati memoriju, napisati funkciju poređenja.
- Koristeći qsort, sortirati nizove struktura po različitim kriterijumima...

Pregled

Literatura

- Slajdovi su pripremljeni na osnovu knjige
  Filip Marić, Predrag Janičić: Programiranje 1
  i šestog poglavlja knjige
  Predrag Janičić, Filip Marić: Programiranje 2
- Za pripremu ispita, slajdovi nisu dovoljni, neophodno je koristiti knjigu!