www.rajak.rs
skolarajak@gmail.com programerska.radionica@gmail.com
www.facebook.com/Skola.Rajak

Програмерска радионица

Програмски језик C

Додатни материјал

Аутори:

Ким Новак, Себастиан Новак

www.rajak.rs
skolarajak@gmail.com programerska.radionica@gmail.com
www.facebook.com/Skola.Rajak

Предговор

Идеја овог писаног материјала је да прошири оно што је речено на предавањима, никада неће излазити ван оквира области за које је предвиђен, односно читање и усвајање и овог градива није неопходно за праћење наставе програмерске радионице.

Такође, на предавањима се неће тражити ништа што није на њима већ речено, како нико не би био оштећен.

Насупрот томе, при излагању градива у овим материјалима, подразумеваће се да је градиво са предавања усвојено и понављање градива са предавања ће бити минимално.

За оне који су заинтересовани и желе да науче више, овај материјал ће служити као помоћ при савладавању сложенијих проблема.

Све примедбе, критике, утиске, сугестије као и уочене грешке (лексичке или везане за градиво) можете послати мејлом на programerska.radionica@gmail.com .

Унапред захвални за Ваш труд:

Аутори

Садржај

www.rajak.rs
skolarajak@gmail.com programerska.radionica@gmail.com
www.facebook.com/Skola.Rajak

1.0 Низови

Низ представља неки скуп елемената који су истог типа. Ти елементи се у меморији налазе један до другог. Сви чланови се у меморији налазе један иза другог. Зато нам је за приступ адреси довољан само први члан низа јер ће сви остали бити на n-том растојању од њега. За сада, рећи ћемо да је број елемената низа нека унапред дефинисана вредност која се не може променити. То је зато јер морамо унапред обезбедити меморију у рачунару коју ће низ заузети. (Код динамичког низа ће то бити другачије).

1.1 Декларација и иницијализација низова

Низ дефинишемо тако што прво наводимо ког типа су **сви** елементи низа, затим дајемо име низу и у [] пишемо величину низа(колико елемената ће садржати низ). Наш низ ће изгледати овако:

```
int niz[10];
```

Односно општи облик

```
tip imeNiza[brojElemenata];
```

Вредности елемената низа можемо унапред да одредимо или да питамо корисника да унесе вредност. Када унапред стављамо вредност сваког елемента низа то изгледа овако:

```
int niz[5]; //niz od 5 elemenata gde su svi elementi tipa int
niz[0]=1;  //prvi clan ce imati vrednost 1
niz[1]=2; //drugi clan ce imati vrednost 2
niz[2]=3;
niz[3]=4;
niz[4]=5;
```

Оно што се налази у угластим заградама зовемо индекс и он представља место елемента у низу. Индексирање низа увек почиње од 0 у *C*-у. Односно индекс иде од 0 до n-1 где n представља број елемената низа.

У случају да желимо да корисник унесе вредности елемената низа:

```
int niz[30];
int n;
printf(“Unesite broj elemenata niza, ne veci od 30\n”); //dajemo korisniku mogucnost da ne unosi svih 30 clanova nego na primer samo 15 od 30 kojih smo zauzeli u memoriji
scanf(“%d”, n); //ocitavamo vrednost za broj elemenata niza
for (i=0;i<n;i++){
printf(“niz[%d]=”, i);
scanf(“%d”, &niz[i]);
} //ovim su unete vrednosti svih elemenata niza
```

1.2 Тражење максимума(минимума) и сортирање низа

Сваком елементу низа смо доделили неку вредност. Ако желимо да пронађемо највећи или најмањи елемент низа потребно је да прво прогласимо неки елемент низа за максимум или минимум. Ми ћемо увек узимати први елемент за максимум или минимум јер тако не морамо да упоређујемо оне пре њега. Дакле проглашавамо први(нулти) елемент низа за максимум.

```
int a[30];
int max;

 a[0]=max;

for(i=1;i<30;i++){
        if(a[i]>max){
                max=a[i];
        }
}

printf(“%d”, max);
```

За минимум би исто прогласили први елемент низа за најмањи и упоређивали га са осталим члановима само би у овом случају услов био да ли је i-ти члан низа мањи од првог(претходно проглашеног минимума). Ако јесте он постаје нови минимум и наставља се даље проверавање да ли постоји елемент који је мањи и од њега.

Постоје разне методе сортирања низа. На предавању смо имали пример који ради исто што и bubble sort. На пример имамо низ елемената и желимо да их поређамо тј. Уредимо по растућем или опадајућем редоследу.

Упоређиваћемо елемент на позицији i са елементом који је одмах иза њега на позицији j.
За ово ће нам бити потребне две for петље једна која ће бројати индекс i и другу која ће се односити на индекс j. Ако је j-ти елемент бећи од i-тог потребно је да замене места. Да се не би изгубила вредност једног или другог елемента, биће нам потребна додатна помоћна променљива коју ћемо назвати temp (наговештава да служи само за привремено смештање неке вредности у њу). Замена ако је i-ти елемент већи од j-тог односно од оног са његове десне стране(следећи елемент) се постиже на следећи начин:

```
for(i=0;i<7;i++){ // овде ће i ићи до претпоследњег члана јер последњи члан нема суседа са десне стране(нема следећег елемента јер је он последњи)
      for(j=i+1;j<8;j++){ // j ће ићи до последњег елемента
          if(X[j]<X[i]){ //ако је i-ти елемент већи од j-тог треба да замене места
              temp = X[j]; //вредност j-тог се смешта у помоћну променљиву
              X[j] = X[i]; //j-том елементу се додељује вредност i-тог
              X[i] = temp;// i-ти елемент добија вредност j-тог која је била смештена у помоћну променљиву
          }
      }
   }
```

Овим смо заменили места та два елемента.