

Vježbe –2. dio - Zadaci

Nizovi karaktera

Za kopiranje stringa (niz *i* u niz *ime*) koristite funkciju `strcpy_s` ili `strncpy_s`. Funkcija `strcpy_s` kopiraju sadržaj na adresi izvorišnog niza karaktera, uključujući i terminirajući null-karakter u lokaciju određenu parametrom odredište.

Ove funkcije predstavljaju sigurnosno proširenje za ranije korištene funkcije `strcpy` i `strncpy`. Ranija verzija `strcpy` nije imala mogućnost utvrđivanja da li je string koji se kopira prevelik za odredišni buffer. Funkcija `strcpy_s` uzima veličinu odredišnog buffera kao parametar kako bi mogla utvrditi hoće li doći do prelijevanja buffera. Sintaksu funkcije `strcpy_s` čine tri parametra:

- o prvi parametar predstavlja *destination* (pokazivač na odredišni niz karaktera)
- o drugi parametar predstavlja veličinu odredišnog niza
- o treći parametar predstavlja *source* (pokazivač na izvorni niz karaktera)

Da bi se izračunao drugi parametar, uobičajeno je funkcijom `strlen` najprije izmjeriti potrebnu veličinu odredišnog niza. Funkcija `strlen` prima samo jedan parametar, tj niz karaktera kojem se mjeri veličina. Osim veličine izvornog niza funkciji `strlen` se dodaje i još jedan element za null-karakter kao u sljedećem primjeru.

Primjer rada sa funkcijom `strcpy_s`:

```
void main()
{
    char d[20];
    char s[] = "ovo je neki string";
    int velicina = strlen(s) + 1;
    strcpy_s(d,velicina, s);
    cout << d << endl; //ispis: ovo je neki string
}
```

Slijedi isti primjer ali bez korištenja niza *s* i varijable *velicina*:

```
void main()
{
    char d[20];
    strcpy_s(d,strlen("Ovo je neki string")+1, "Ovo je neki string");
    cout << d << endl; //ispis: ovo je neki string
}
```

Funkcijom `strncpy_s` navodi se maksimalni broj karaktera koji će se kopirati iz *source* u *destination* – ovo ćemo koristiti ako je niz *destination* možda manji od stringa koji se nalazi u nizu *source*.

Ako bi pokušali kopirati niz karaktera u *destination*, a niz *destination* je manji od potrebnog prostora doći će do greške u *run-time*-u aplikacije

Primjer rada sa funkcijom `strncpy_s`:

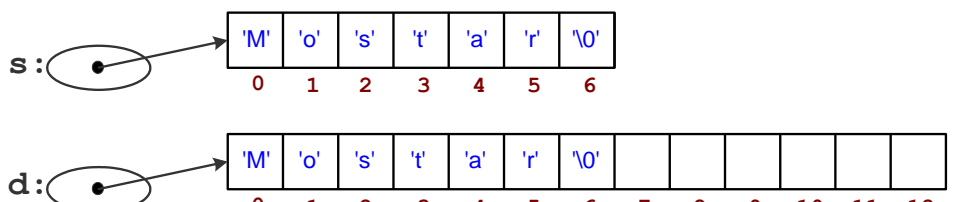
```
void main()
{
    char d[20];
    strncpy_s(d,"Ovo je neki string", 3);
    cout << d << endl; //ispis: ovo je neki string
}
```

```
}}
```

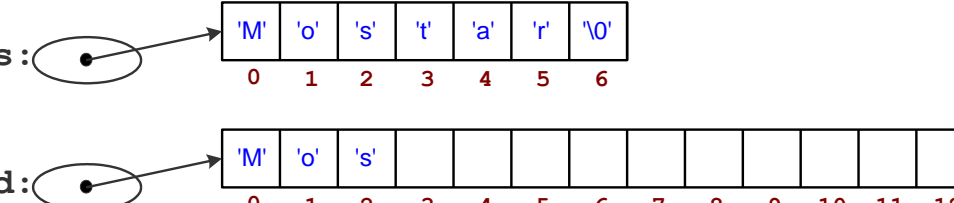
Za funkcije `strcpy_s` i `strncpy_s` nije bitno da li se koriste statički ili dinamički nizovi za izvor i destinaciju. U sljedećim primjerima nije bitno da li `s` predstavlja statički ili dinamički niz, isto vrijedi za `d`.

Slijede primjeri kopiranja stringova pomoću funkcije `strcpy_s` i `strncpy_s` (nemojte obraćati pažnju na alokaciju i način unosa stringa, obraćajte pažnju na alokaciju u memoriji - dijagram) :

Slučaj a: niz *destination* je veći od niza *source*

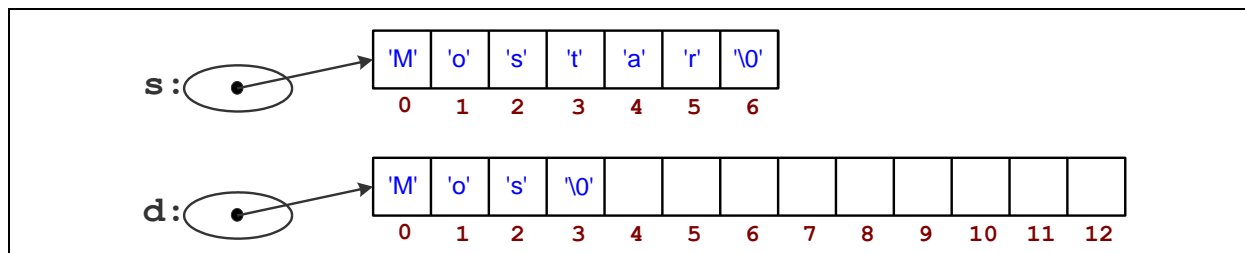
| | |
|--|--|
| s je dinamički niz d je statički niz | d je dinamički niz s je statički niz |
| char* s = new char[7]; cin >> s; char d[13]; strcpy_s(d, strlen(s)+1, s); | char s[7]; cin >> s; char* d = new char[13]; strcpy_s(d, strlen(s)+1, s); |
|  <p>s: 'M' 'o' 's' 't' 'a' 'r' '\0'</p> <p style="text-align: center;">0 1 2 3 4 5 6</p> <p>d: 'M' 'o' 's' 't' 'a' 'r' '\0' </p> <p style="text-align: center;">0 1 2 3 4 5 6 7 8 9 10 11 12</p> | |

Slučaj b: korištenje funkcije `strncpy_s`

| | |
|--|---|
| s je dinamički niz d je statički niz | d je dinamički niz s je statički niz |
| char* s = new char[7]; cin >> s; char d[13]; strncpy_s(d, s, 3); | char s[7]; cin >> s; char* d = new char[13]; strncpy_s(d, s, 3); |
|  <p>s: 'M' 'o' 's' 't' 'a' 'r' '\0'</p> <p style="text-align: center;">0 1 2 3 4 5 6</p> <p>d: 'M' 'o' 's' </p> <p style="text-align: center;">0 1 2 3 4 5 6 7 8 9 10 11 12</p> | |

Ovdje je potrebno dodati karakter `'\0'` na kraj stringa. Slijedi ispravak.

| | |
|---|---|
| char* s = new char[7]; cin >> s; char d[13]; strncpy_s(d, s, 3); d[3] = '\0'; | char s[7]; cin >> s; char* d = new char[13]; strncpy_s(d, s, 3); d[3] = '\0'; |
|---|---|



Slučaj c: niz *destination* je manji od niza *source*, ali nije manji od *stringa* iz *source*

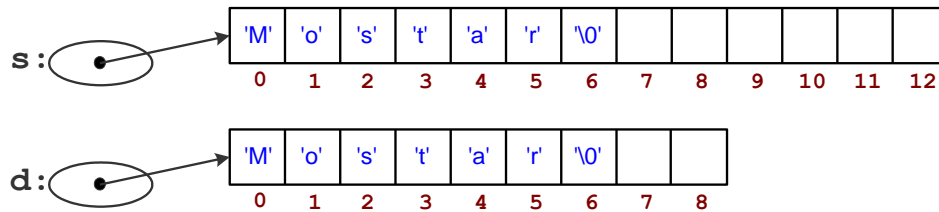
Da li će doći do greške?

```
char* s = new char[13];
cin >> s;
char d[9];

strcpy_s(d, strlen(s)+1, s);
```

```
char s[13];
cin >> s;
char* d = new char[9];

strcpy(d, strlen(s)+1, s);
```



Ovdje neće doći do greške ako je korisnik unio manje od 9 karaktera, jer funkcija `strcpy_s` ne kopira čitav niz, ona kopira karakter po karakter dok ne dođe do karaktera čiji je ASCII broj 0.

Funkcija `strcpy_s` ne zna i ne može da zna koliko je dug niz `s`. Inače vrijedi:

Sve funkcije koje rade sa stringom (niz tipa `char`) rade sa pokazivačem na taj niz. Kraj stringa (a ne kraj niza) predstavlja karakter '\0'. Nemoguće je saznati dužinu nekog niza. Čak funkcija `strlen` računa dužinu *stringa* pomoću karaktera '\0'. Primjer:

Pitanje: Šta će ispisati sljedeći kôd?

```
char p [20];

cout << strlen(p) << endl;
```

Odgovor:

- Rezultat je nepredvidljiv. Funkcija `strlen` će kao parametar primiti adresu od prvog elementa niza koji je tipa `char`. Zatim će brojati korake od prvog elementa pa dok ne pronađe vrijednost 0 u RAM-u. Broj koraka od *prvog elementa* do *vrijednost '\0'* će biti izlaz iz ove funkcije.

Pitanje: Šta bi ispisale sljedeće dvije linije kôda kad bi se dodale u prethodna tri primjera nakon kopiranja stringa `s` u `d`?

```
cout << strlen(s) << endl;
cout << strlen(d) << endl;
```

Odgovor:

| a) | b) | c) |
|--------------------------------|--------------------------------|--------------------------------|
| strlen(s) = 6 strlen(d) = 6 | strlen(s) = 6 strlen(d) = 3 | strlen(s) = 6 strlen(d) = 6 |

Primjer nedozvoljenog kopiranja:

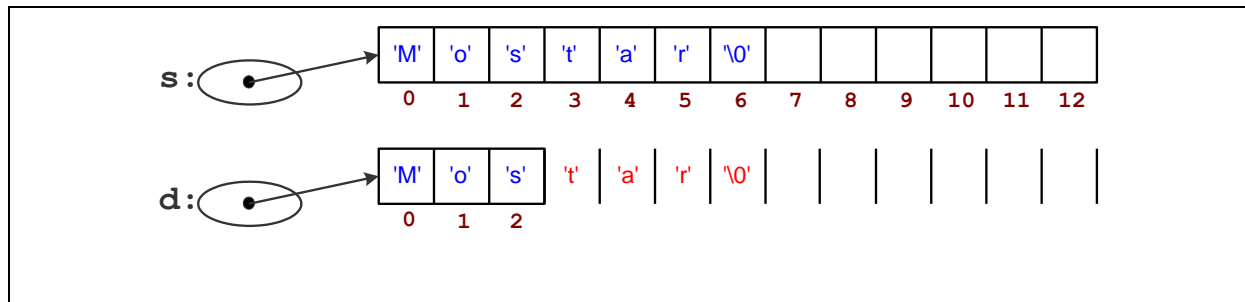
niz *destination* je manji od *stringa* iz *source*

```
char* s = new char[13];
cin >> s;
char d[3];

strcpy_s(d, strlen(s)+1, s);
```

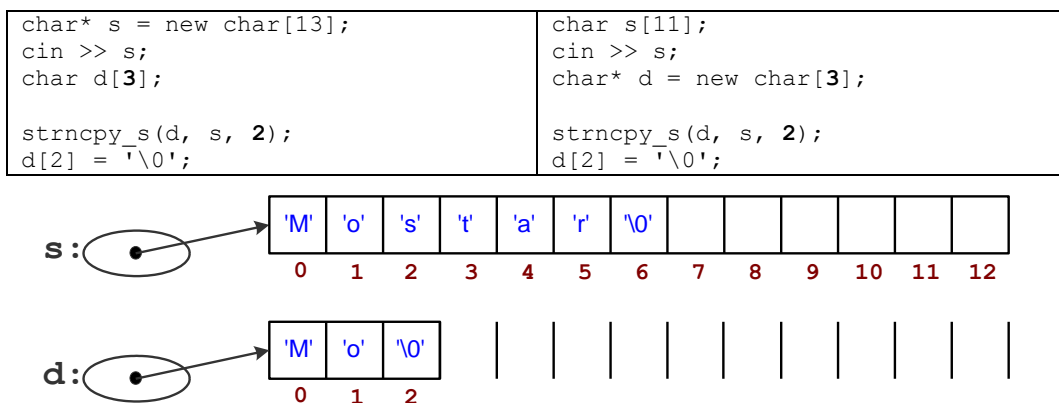
```
char s[11];
cin >> s;
char* d = new char[3];

strcpy_s(d, strlen(s)+1, s);
```



Ako pretpostavimo da nam je *destination* niz konstante dužine, koristit ćemo funkciju `strncpy_s` da bi izbjegli moguću grešku koja nastaje prilikom kopiranja stringa u niz koji nije dovoljno veliki. Nakon korištenja funkcije `strncpy_s` potrebno je dodati karakter za kraj stringa. U ovom primjer karakter '\0' treba dodati na zadnju indeks poziciju, tj. 2.

Slijedi ispravak prethodne greške koja se javlja jer je niz *destination* manji od *stringa* iz *source*:



U svim *string*-funkcijama moguće umjesto imena niza tipa `char` koristiti string napisan između navodnih znakova. Slijede primjeri:

```
int d;
d = strlen("ovo je neki string");
cout << d; //ispisuje se broj 18
```

```
char n[100];
strcpy_s(n, strlen("ovaj string smjestamo u niz n")+1, "ovaj string smjestamo u niz n");
cout << n << endl;
```

Za svaki string koji je napisan između navodnih znakova, alocirat će se privremeni niz u koji će smjestiti napisani string.

Zadatak 1:

Koristeći pokazivač na niz karaktera u dinamičkoj memoriji, napravite program koji će u taj niz smjestiti ime i prezime (u jedan niz) uneseno sa tastature. Program treba zahtijevati i unos mjesta rođenja koji se treba smjestiti u drugi niz, koji se ne kreira u dinamičkoj memoriji. Ta dva niza trebate iskopirati (koristeći funkciji `strcpy_s`) u treći niz koji je kreiran u dinamičkoj memoriji. Reciklirajte oba izvorna niza.

Za spajanje nizova koristite funkciju `strcat_s`.

Veličina niza trećeg niza smije biti samo onoliko velika koliko je to neophodno.

Rješenje se nalazi na stranici 11.

Zadatak 2:

Analizirajte sljedeći program.

```
1: void main()
2: {
3:     char* str1 = "ovo je prvi string u statickoj memoriji";
4:     char str2[] = "ovo je drugi string u statickoj memoriji";
5:
6:     char* p1;
7:     char* p2;
8:     p1 = str1; //p1 pokazuje na string str1
9:     p2 = str2; //p2 pokazuje na string str2
10:
11:     cout << str1 << endl;
12:     cout << str2 << endl;
13:
14:     cout << p1 << endl;
15:     cout << p2 << endl;
16: }
```

Linija br. 11 i br. 14 će ispisati poruku "ovo je prvi string u statickoj memoriji" jer `str1` i `p1` pokazuju na niz u statickoj memoriji.

Linija br. 12 i br. 15 će ispisati poruku "ovo je drugi string u statickoj memoriji" jer `str2` i `p2` pokazuju na niz u statickoj memoriji.

Sva četiri pokazivača (`str1`, `str2`, `p1` i `p2`) pokazuju na prvi elemenat niza. Pitanje:

- Zašto onda nije potrebno dereferencirati te pokazivače ?
- Zašto se ne ispiše samo prvi elemenat (karakter) niza ?

Objekat `cout` klase `iostream` je tako definisan da, kada prima adresu nekog elementa niza, ispisuje karaktere (naredne elemente niza) sve dok ne naiđe na karakter `'\0'`.

Zadatak:

Dovršite naredni program tako što ćete ...

- a) ispisati dati string od trećeg karaktera
- b) ispisati prvi karakter stringa
- c) ispisati zadnji karakter stringa
- d) ispisati zadnjih 10 karaktera stringa

```
1: void main()
2: {
3:     char* str1 = "ovo je prvi string u statickoj memoriji";
    ...
}
```

Pomoć:

- Ako derefenciramo adresu nekog elementa niza onda će se ispisati samo taj karakter.
- Ako proslijedimo adresu nekog elementa niza onda će se ispisati svi uzastopni karakteri do '\0'.
- U zadacima **a** i **d** treba proslijediti adresu, dok u zadacima **b** i **c** treba dereferencirati neku adresu da bismo dobili vrijednost (karakter) tog elementa.

Rješenje se nalazi na stranici 11.

Zadatak 3:

Deklarišite dva stringa. Zatim ih iskopirajte (spojite) u treći i ispišite ga na ekran.

- a) koristite funkciju `strcat_s`
- b) bez korištenja funkcije `strcat_s`

Rješenje se nalazi na stranici 11.

Zadatak 4:

Implementirajte funkciju `dodjeli_str` koja će alocirati niz u dinamičkoj memoriji i kopirati ulazni string u taj niz.

```
1: void main()
2: {
3:     char* str1;
4:     str1 = dodjeli_str("Ovo je neki string");//funkcija vrši alokaciju i kopiranje
5:
6:     cout << str1 << endl;
7:
8:     delete [] str1; //dealokacija
9: }
```

Rješenje se nalazi na stranici 13.

Ova funkcija nam može malo olakšati rad sa dinamičkom memorijom. Analizirajte sljedeća dva programa koja rade istu stvar. U prvom nije korištena funkcija `str1`, dok u drugom programu jeste.

```
1: void main()
2: {
3:     char* str1;
4:
5:     str1 = new char[strlen("Ovo je neki string") + 1];
6:     strcpy_s(str1, strlen("Ovo je neki string")+1, "Ovo je neki string");
7:     cout << str1 << endl;
8:
9:     char* str2 = new char[strlen(str1) + 1];
10:    strcpy_s(str2, strlen(str1)+1, str1);
11:    cout << str2 << endl;
12:
13:    delete [] str1;
14:    delete [] str2;
15: }
```

```
1: void main()
2: {
3:     char* str1;
4:
5+6:     str1 = dodjeli_str("Ovo je neki string");
7:     cout << str1 << endl;
8:
9+10:    char* str2 = dodjeli_str(str1);
11:    cout << str2 << endl;
12:
13:    delete [] str1;
14:    delete [] str2;
15: }
```

Zadatak 5:

Implementirajte funkciju `dodaj_str` koja će prvom stringu dodati drugi string na isti način kao funkcija `strcat` s tim da ova funkcija vrši ponovnu alokaciju prvog niza.

```
1: void main()
2: {
3:     char* str1 = dodjeli_str("Prvi niz. ");
4:     dodaj_str(str1, "Ovo je drugi niz");
5:
6:     cout << str1 << endl;
7:     delete [] str1;
8: }
```

Rješenje se nalazi na stranici 13.

Dodatak:

Da bi ova funkcija ispravno radila neophodno je da `str1` pokazuje na neki string. On mora pokazivati barem na prazan string u dinamičkoj memoriji, tj. na karakter `'\0'`.

```
1: void main()
2: {
3:     char* str1 = new char;
4:     *str = '\0';
5:     dodaj_str(str1, "Ovo je drugi niz");
6:
7:     cout << str1 << endl;
8:     delete [] str1;
9: }
```

...ili jednostavnije - koristeći funkciju `dodjeli_str`:

```
1: void main()
2: {
3:     char* str1 = dodjeli_str("");
4:     dodaj_str(str1, "Ovo je drugi niz");
5:
6:     cout << str1 << endl;
7:     delete [] str1;
8: }
```

Pomoć:

- neka funkcija prima dva niza, `a` i `b` (`a` će biti `str1`, `b` će biti `"Ovo je drugi niz"`)

- funkcija `dodaj_str` treba alocirati novi niz u koji mogu stati oba niza (`a` i `b`)
- iskopirajte nizove `a` i `b` u novi niz
- dealocirajte stari niz `a`
- neka `a` pokazuje na novi niz
- da bi izmjena pokazivača `a` uticala i na pokazivač `str1` potrebno je da prvi parametar bude po referenci, ali da tip podatka ostane tipa `char*`

Zašto referenca nad pokazivačem?

Ako ne koristimo referencu alocirat će se prostor za dvije varijable (pokazivača) `a` i `str1`. Oba pokazivača će pokazivati na istu adresu (prvi element niza). Ako mijenjamo vrijednost elemenata niza u funkciji onda će ta promjena uticati i u glavnom programu i ako nismo koristili referencu, a to najčešće i želimo. Problem nastaje ako želimo da `a` pokazuje na novi niz i mi želimo da i `str1` pokazuje na taj novi niz. Ako ne koristimo referencu onda će samo `a` pokazivati na taj niz, a `str1` će ostati ne promijenjen. Ako koristimo referencu onda će i `a` time i `str1` pokazivati na novi niz. *To zamislite na sljedeći način:* Ako koristimo referencu onda imamo jedan pokazivač (sa dva imena: `a` i `str1`), a ako ne koristimo referencu onda imamo dva pokazivača (pokazivač `a` i pokazivač `str1`, mi možemo mijenjati samo smjer pokazivača `a`, jer je on formalni parametar funkcije).

Pitanje:

Koja je razlika između sljedeća dva zaglavlja funkcije?

```
void test(int* A)
{
    //...
}
```

```
void test(int B[])
{
    //...
}
```

Odgovor:

Nema nikakve razlike. `A` i `B` predstavlja pokazivače. Ovu drugu metodu koriste programeri koji 'strahuju' od pokazivača (ne shvatajući da su `A` i `B` pokazivači).

Isto vrijedi i za dvodimenzionalne nizove:

```
void test(int** C)
{
    //...
}
```

```
void test(int* D[])
{
    //...
}
```

Zadatak 6:

Ponovo definišite funkciju `"void dodjeli_str(char* &d, char* b)"` koja radi istu stvar kao funkcija `"char* dodjeli_str(char* s)"` s tim da izlaz bude pomoću reference (izlaznog parametra) a ne pomoću naredbe `return`.

Nemojte brisati prvu definiciju funkcije, jer je moguće imati više funkcija sa istim imenom a koje se razlikuju po broju ili tipu parametara. Takve funkcije se zovu *overloaded* funkcije. Kompajler će pozivati onu definiciju funkcije kojoj odgovaraju proslijeđeni parametri.

Nemojte vršiti dealokaciju da ne bi program padao ako programu proslijedimo 'divlji' pozivač kao *destination*. Ova funkcija radi uz pretpostavku da je *destination* niza već dealociran (ako je bio alociran). Ako to nije slučaj, program će i dalje raditi, ali dolazi do neznatnog curenja memorije koje možemo zanemariti. Ako pretpostavimo da je naša funkcija `dodjeli_str2` dio standardnih funkcija programskog jezika onda propust koji dolazi pri curenju memorije nije naš (programera koji napravio funkciju) nego drugog krajnjeg programera kao korisnika naše funkcije, ako je u dokumentaciji funkcije naglašeno da funkcija ne vrši dealokaciju *destination* niza.

Rješenje se nalazi na stranici 14.

Zadatak 7:

- a) Definišite funkciju `spoji_str` koja će kreirati izlazni string sastavljen od dva ulazna stringa. Izlazna vrijednost neka bude pomoću naredbe `return`.

Rješenje se nalazi na stranici 14.

- b) Ponovo definišite funkciju `spoji_str` koja će kreirati izlazni string sastavljen od tri ulazna stringa. Izlazna vrijednost neka bude pomoću naredbe `return`.

Rješenje se nalazi na stranici 14.

Zadatak 8:

Dovršite program:

```
#include <iostream>
using namespace std;

char crt[] = "\n-----\n";

void OslobodiMemoriju(char * &tekst)
{
    //dealocirati tekst
}
void Informacije(char * tekst)
{
    int razmaci = 0, brojevi = 0, velika = 0, mala = 0, interpunkcijski = 0;

    //...

    cout << crt << "\t\t:INFO: " << crt;
    cout << crt << "Tekst: " << tekst << crt;
    cout << "Niz ima " << strlen(tekst) << " karaktera." << crt;
    cout << "Razmaka: \t\t\t" << razmaci << endl;
    cout << "Brojeva: \t\t\t" << brojevi << endl;
    cout << "Velikih slova: \t\t\t" << velika << endl;
    cout << "Malih slova: \t\t\t" << mala << endl;
    cout << "Interpunkcijskih znakova: \t" << interpunkcijski;
```

```

        cout << crt << "Info: Informacije prikazane...." << crt;
    }

    void DodajTekst(char * &tekst)
    {
        //stari tekst + razmak + novi tekst
    }

    void Pretraga(char * tekst)
    {
        //...
    }

    void UnosTeksta(char *& tekst)
    {
        //...
    }

    int PrikaziMeni() {
        int izbor = 1;
        do {
            cout << crt << "\t\t::MENI::" << crt;
            cout << "1. Unos novog teksta" << endl;
            cout << "2. Dodavanje teksta" << endl;
            cout << "3. Informacije o tekstu" << endl;
            cout << "4. Pretraga" << endl;
            cout << "5. Zatvori editor" << endl;
            cout << "Unesite vas izbor: ";
            cin >> izbor;
            cin.ignore();
            system("cls");
        } while (izbor < 1 || izbor>5);
        return izbor;
    }

    void main() {
        int izbor = 0;
        char * tekst = NULL;
        do {
            cout << crt << "\t\t::TEKST EDITOR::";
            izbor = PrikaziMeni();
            switch (izbor) {
                case 1:
                    UnosTeksta(tekst); break;
                case 2:
                    DodajTekst(tekst); break;
                case 3:
                    Informacije(tekst); break;
                case 4:
                    Pretraga(tekst); break;
            }
            system("pause>0");
            system("cls");
        } while (izbor != 5);

        if (tekst != NULL)
            OslobodiMemoriju(tekst);

        cout << crt;
    }

```

Dodatni zadaci bez rješenja:

Zadatak 9:

Napisati program koji će učitati niz karaktera i ispisati koliko puta se određeni karakter pojavljuje u tom nizu.

Zadatak 10:

Program koji ispisuje tabelu ASCII karaktera u 19 redova i 5 kolona, počevši od znaka *space*.

Zadatak 11.

Izračunati dužinu niza karaktera i napisati niz karaktera unazad uz pomoć funkcija.

Zadatak 12.

Napisati funkciju izbaci_praznine koja iz niza karaktera str izbacuje sve praznine.

Zadatak 13.

Napisati program koji će omogućiti:

- Unos 2D dinamička niza vodeći računa da su svi elementi **dvocifreni** (ukoliko unos nekog elementa ne zadovoljava uslov, ponavljati unos tog elementa dok se ne zadovolji uslov) – Koristiti funkciju **unos**
- Izvršiti transpoziciju niza tako što će se zamjeniti redovi i kolone – Koristiti funkciju **transpose**

Zadatak 14.

Napišite program prema sljedećim zahtjevima:

- za studente se unosi broj bodova postignutih na ispitu (korisnik unosi broj studenata), bodovi su u rasponu od 0 do 100
- prebrojati i ispisati koliko je studenata bilo uspješno, a koliko neuspješno na ispitu (ako je prag prolaznosti na 60 bodova);
- za uspješne studente (one koji su položili ispit) izračunati prosječan broj bodova;
- ukoliko je više od polovice studenata bilo neuspješno na ispitu ispisati poruku – Potrebno više sati instrukcija!

Dodatno:

Uspješne i neuspješne studente spremati u zasebne dinamičke nizove.

Rješenja

Rješenje zadatka br. 1:

```
1:  #include <iostream>
2:  using namespace std;
3:
4:  void main()
5:  {
6:      char* A = new char[30];
7:      char B[20];
8:
9:      cout << "Unesi ime i prezime: \n";
10:     cin.getline(A, 29); // maksimalna duzina je 29, jer je potrebno jedno mjesto za
11:     //'\\0'
12:     cout << "Unesi mjesto rođenja: \n";
13:     cin.getline(B, 19);
14:     // duzina A + razmak + duzina B + '\\n'
15:     int PotrebnaDuzina = strlen(A) + 1 + strlen(B) + 1;
16:     char* C = new char[PotrebnaDuzina];
17:
18:     strcpy_s(C, PotrebnaDuzina, A); // A kopira u C
19:     strcat_s(C, PotrebnaDuzina, " "); // u C dodaje "razmak", a mogao je biti bilo
20:     //koji tekst
21:     strcat_s(C, PotrebnaDuzina, B); // u C dodaje B
22:
23:     cout << C << endl;
24:     delete[] A;
25:     delete[] C;
26: }
```

Rješenje zadatka br. 2:

```
1:  void main()
2:  {
3:      char* str1 = "ovo je string u statickoj memoriji";
4:
5:      // a)
6:      cout << (str1 + 3) << endl;
7:      // b)
8:      cout << *str1 << endl; //isto kao str1[0]
9:      // c)
10:     int d = strlen(str1);
11:     cout << str1[d-1] << endl; // isto kao *(str1 + d - 1)
12:     // d)
13:     cout << (str1 + d - 10) << endl;
14: }
```

Rješenje zadatka br. 3a:

```
1:  void main()
2:  {
3:      char* str1 = "Ovo je prvi string. ";
4:      char* str2 = "Ovo je drugi string.";
5:      char* str3;
6:
7:      int d1 = strlen(str1);
8:      int d2 = strlen(str2);
9: }
```

```

10:     int duzina = d1 + d2 + 1;
11:     str3 = new char[duzina];
12:
13:
14:     strcpy_s(str3,duzina, str1);
15:     strcat_s(str3,duzina, str2);
16:
17:     cout << str3 << endl;
18:     delete[] str3;
19: }

```

Rješenje zadatka br. 3b:

```

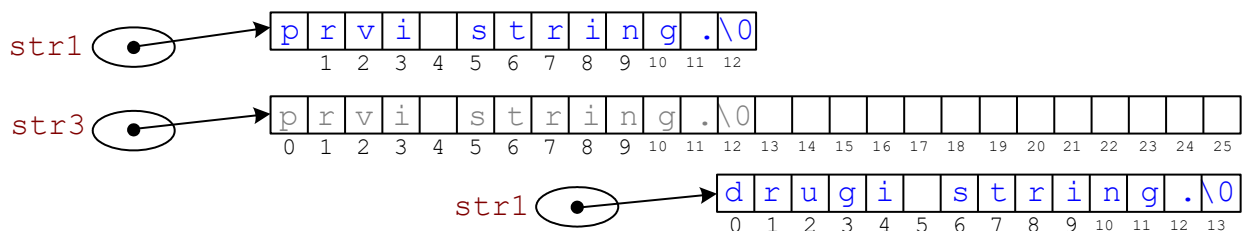
1:  #include <iostream>
2:  using namespace std;
3:
4:  void main()
5:  {
6:      char* str1 = "prvi string.";
7:      char* str2 = "drugi string.";
8:      char* str3;
9:
10:     int d1 = strlen(str1); //12
11:     int d2 = strlen(str2); //13
12:     int duzina = d1 + d2 + 1;
13:     str3 = new char[duzina];
14:
15:     strcpy_s(str3 + 0,duzina, str1);
16:     strcpy_s(str3 + d1,strlen(str2)+1, str2);
17:
18:     cout << str3 << endl;
19:     delete[] str3;
20: }

```

Rješenje ovog programa nije baš od neke koristi. Služi nam samo kao primjer prosljeđivanja adrese nekog elementa funkcijama koje manipuliraju sa stringom.

Broj 0 u liniji br. 12 nema nikakvo značenje.

U liniji br. 13 se u niz elementa koji počinje od elementa sa indeksom 12 kopira drugi string kao na slici:



Karakter '\0' koji se nalazi u elementu sa indeksom 12 će biti zamijenjen sa karakterom 'd' tako da će kraj proširenog stringa `str3` biti na 25-om elementu.

Rješenje zadatka br. 4:

Povratna vrijednost funkcije će biti podatak tipa `char*`, tj. adresa prvog elementa novoalociranog niza. Povratna vrijednost ove funkcije se može dodjeljivati samo pokazivačima.

```

1:  #include <iostream>
2:  using namespace std;
3:
4:  char* dodjeli_str(char* s)
5:  {
6:      char* x = new char[strlen(s) + 1];
7:      strcpy_s(x, strlen(s) + 1, s);
8:      return x;
9:  }
10:
11: void main()
12: {
13:     char* str1;
14:     str1 = dodjeli_str("Ovo je neki string");
15:
16:     cout << str1 << endl;
17:     delete[] str1;
18: }
```

Nemojte ovu funkciju koristiti u `cout` naredbi jer neće se moći deallocirati niz.

```
cout << dodaj_str("Ovaj niz će biti izgubljen.") << endl;
```

Ovdje će se napraviti kopija stringa i ta kopija (koja će biti povratna vrijednost funkcije) će biti izgubljena. Nećemo je moći deallocirati niti ponovo ispisati.

Rješenje zadatka br. 5:

```

1:  #include <iostream>
2:  using namespace std;
3:
4:  char* dodjeli_str(char* s)
5:  {
6:      char* x = new char[strlen(s) + 1];
7:      strcpy_s(x, strlen(s)+1, s);
8:      return x;
9:  }
10:
11: void dodaj_str(char* &a, char* b)
12: {
13:     int d1 = strlen(a);
14:     int d2 = strlen(b);
15:     int duzina = d1+d2+1;
16:     char* x = new char[duzina]; //alociramo novi niz tako da u njega moze
17:     stati niz i niz b
18:     strcpy_s(x, duzina, a); // u novi niz kopiramo a
19:     strcat_s(x, duzina, b); // u novi niz dodajemo b
20:
21:     delete[] a; //alociramo stari niz a
22:     a = x;      //neka a pokazuje na novi niz
23: }
24:
25: void main()
26: {
27:     char* str1 = dodjeli_str("Prvi niz. ");
```

```
26:     dodaj_str(str1, "Ovo je drugi niz");
27:
28:     cout << str1 << endl;
29:     delete[] str1;
30: }
31:
```

Funkcija `dodaj_str` će pasti u sljedećim situacijama:

1. ako prvi niz ili drugi niz nisu alocirani
2. ako je vrijednost prvog ili drugog pokazivača jednaka 'divljoj' ili jednaka NULL
3. ako je string u prvom nizu alociran u deklaraciji, kao npr.:

```
char* str1 = "Neki niz";
dodaj_str(str1, "Ovo je drugi niz");
```

Sljedeći fragment programa će uzrokovati grešku:

```
char* str1 = "Neki niz";
delete [] str1;
```

Rješenje zadatka br. 6:

```
1: void dodjeli_str(char* &a, char* b)
2: {
3:     int d = strlen(b);
4:     a = new char[d + 1];
5:     strcpy_s(a,d+1, b);
6: }
```

Rješenje zadatka br. 7a:

```
1: char* spoji_str(char* a, char* b)
2: {
3:     int duzina = strlen(a) + strlen(b) + 1;
4:     char* x = new char[duzina];
5:     strcpy_s(x,duzina, a);
6:     strcat_s(x,duzina, b);
7:     return x;
8: }
```

Rješenje zadatka br. 7b:

```
1: char* spoji_str(char* a, char* b, char* c)
2: {
3:     int duzina = strlen(a) + strlen(b) + strlen(c) + 1;
4:     char* x = new char[duzina];
5:     strcpy_s(x,duzina, a);
6:     strcat_s(x,duzina, b);
7:     strcat_s(x,duzina, c);
8:     return x;
9: }
```

Kopirajte vaše četiri funkcije sa njihovim prototipovima u jedan fajl i taj fajl snimate u folder gdje se nalaze i drugi *header*-fajlovi ili negdje na disk tako da možete ubuduće koristiti vaše funkcije kao gotove funkcije, npr.:


```
1:  #include <iostream>
2:  using namespace std;
3:  #include "c:\temp\str_funkcije.h"
4:
5:  void main()
6:  {
7:      char* p1 = dodjeli_str("_Fit for FIT");
8:
9:      char* p2;
10:     dodjeli_str(p2, "He says:");
11:     dodaj_str(p2, p1);
12:     cout << "p2 = " << p2 << endl;
13:
14:     char* p3 = spoji_str(p2, " > ", p1);
15:     //...nemojte zaboraviti dealocirati nizove p1, p2 i p3...
```