STRUKTURE PODATAKA I ALGORITMI

Predavanje 01

INFORMACIJE O KOLEGIJU

Nastavnici

- Predavanja:
  - dr.sc. Goran Đambić
  - goran.djambic@racunarstvo.hr
  - Konzultacije: bilo kada, uz prethodnu najavu e-mailom
- Vježbe:
  - Ja (redovni)
  - Mario Žagar (izvanredni)
    - mario.zagar@racunarstvo.hr

Algebra visoka škola za primijenjeno računarstvo

O kolegiju

- Ciljevi kolegija:
  - Upoznavanje s osnovnim strukturama podataka
  - Upoznavanje s osnovnim algoritmima
  - Implementacija nekih struktura i algoritama na računalu
- Kolegij nosi **5 ECTS** bodova (otprilike 30 sati / bod)
  - 30 sati predavanja (15 tjedana po 2 sata)
  - 30 sati vježbi (15 tjedana po 2 sata)
  - 90 sati samostalnog rada (15 tjedana po 6 sati)
    - **Količina samostalnog rada znatno ovisi o stečenom znanju iz kolegija Programiranje**

Algebra visoka škola za primijenjeno računarstvo

Potpis

- Potpis je uvjet za izlazak na ispit
  - Tko ne dobije potpis, mora sljedeće godine ponovno upisati kolegij
- Uvjeti za potpis su dolasci na predavanja i vježbe
- Redovni studenti:
  - 70% predavanja (21 sat)
  - 80% vježbi (24 sati)
- Izvanredni studenti:
  - 60% predavanja i vježbi (36 sati)

Algebra visoka škola za primijenjeno računarstvo

Prikupljanje bodova i ocjene

- Na kolegiju je moguće skupiti najviše **100 bodova** :
  - Prisutnost na nastavi: najviše **3 boda**
  - Blic testovi: najviše **6 bodova**
  - Domaće zadaće: najviše **7 bodova**
  - 3 međuispita: najviše **84 bodova**
  - Usmenog ispita nema

$$\sum = 100$$

- Ocjene:
  - 92,01 – 100,00 bodova: izvrstan (5)
  - 75,01 – 92,00 bodova: vrlo dobar (4)
  - 58,01 – 75,00 bodova: dobar (3)
  - 50,01 – 58,00 bodova: dovoljan (2)

Algebra visoka škola za primijenjeno računarstvo

Ishodi učenja

- Svaki kolegij je podijeljen na više **ishoda učenja**
  - Ishod učenja opisuje što student treba znati, razumjeti i moći napraviti nakon što je uspješno završio proces učenja:
    - **Minimalni** su za ocjenu dovoljan (2)
    - **Željeni** su za ocjene od dobar (3) do izvrstan (5)
  - Obično kolegiji imaju od 4 do 9 ishoda učenja
  - **SPA ima 6 ishoda učenja**

Algebra visoka škola za primijenjeno računarstvo

Ishodi učenja za SPA

Ishod	MINIMALNI ISHODI UČENJA (Po uspješnom završetku predmeta, student će moći)	ŽELJENI ISHODI UČENJA (Uspješan student bi trebao moći)
I1	Opisati što je algoritam, objasniti zapis algoritma, analizirati jednostavnije algoritme te definirati rekurzivne algoritme	Usporediti postojeće algoritme, analizirati složenije algoritme te rješavati probleme upotrebom rekurzije
I2	Opisati i koristiti jednostavne strukture podataka (lista, stog, red)	Kreirati rješenja bazirana na jednostavnijim strukturama podataka (lista, stog, red)
I3	Opisati i koristiti složenije strukture podataka (stablo, gomila, prioritetni red)	Kreirati rješenja bazirana na složenijim strukturama podataka (stablo, gomila, prioritetni red)
I4	Opisati i koristiti algoritme sortiranja	Konstruirati rješenja bazirana na algoritmima sortiranja
I5	Opisati i koristiti algoritme pretraživanja	Konstruirati rješenja bazirana na algoritmima pretraživanja
I6	Definirati tehnike adresiranja te prepoznati primjene tehnika adresiranja	Koristiti tehnike adresiranja za rješavanje problema

Algebra visoka škola za primijenjeno računarstvo

Ishodi učenja i provjere znanja

	M1	M2	M3	Blicevi	Domaće zadaće	Dolaznost	MAX
I1	14			2			16
I2	14				3		17
I3		14		2			16
I4		14			2		16
I5			14	2			16
I6			14		2		16
Izvan ishoda						3	3
Ukupno	28	28	28	6	7	3	100

Algebra visoka škola za primijenjeno računarstvo

Općenito o ispitima

- Na svakom kolegiju vrijedi **pravilo 3 + 1**
  - To znači da student mora položiti ispit iz najviše 4 izlaska
    - 3 redovna izlaska
    - 1 komisijski izlazak
  - Vremenski rok za položiti ispit je **12 mjeseci** od dana upisa
  - Ako student padne na komisiji ili istekne navedenih 12 mjeseci, **mora ponovno upisati kolegij**
- Vodite računa o rokovima prijave i odjave ispita na InfoEduci
  - Ako niste prijavili ispit, ne možete pristupiti ni pismenom niti usmenom dijelu

Algebra visoka škola za primijenjeno računarstvo

Pismeni ispit iz SPA

- Za vrijeme trajanja semestra pišu se 3 međuispita:
  - Prvi međuispit pokriva ishode učenja I1 i I2
  - Drugi međuispit pokriva ishode učenja I3 i I4
  - Treći međuispit pokriva ishode učenja I5 i I6
- Treći međuispit se smatra i **predrokom**
  - To znači da je izlazak na njega jednak izlasku na bilo koji rok (padom na njemu ostaje vam 2 + 1)
- Ako ne položite ispit na predroku, svi bodovi vam ostaju
  - Na sljedećim rokovima birate koje ishode želite popravljati
  - Novoostvareni bodovi uvijek zamjenjuju dotadašnje

Blic testovi i domaće zadaće

- Dio bodova donose i blic testovi i domaće zadaće
- **Blic testovi**
  - Najčešće pitanja na zaokruživanje
  - Pišu se na papiru na predavanjima ili vježbama i traju otprilike 10 minuta
  - Neće biti unaprijed najavljivani
- **Domaće zadaće**
  - Zadaju se na vježbama
  - Rok za predaju riješenih zadaća su obično sljedeće vježbe
- Blic testove i domaće zadaće nije moguće popravljati

Pregled tema

- Okvirni pregled tema koje ćemo obraditi:
  - Analiza složenosti algoritama
  - Primjena rekurzije
  - Lista, stog, red
  - Stablo (binarno stablo), gomila, prioritetni red
  - Algoritmi sortiranja (Selection sort, Insertion sort, Bubble sort, Heapsort, QuickSort, MergeSort)
  - Sekvencijalno i binarno pretraživanje
  - Osnovne tehnike adresiranja
  - Raspršeno (eng. *hash*) adresiranje

Algebra visoka škola za primijenjeno računarstvo

Literatura

- Osnovna:
  - Strukture podataka i algoritmi – priručnik, Visoka škola za primijenjeno računarstvo, 2009.
- Dodatna:
  - Weiss: Data Structures and Algorithm Analysis in C, Addison Wesley, 1997
  - Sedgwick: Algorithms in C…, Addison-Wesley, 2001
  - T. Cormen,C. E. Leiserson, R. L. Rivest: Introduction to algorithms, 2nd edition, MIT Press, 2001.
  - E. Horowitz, S. Sahni, S. Rajasekaran: Computer Algorithms / C++, Computer Science Press, New York, 1997.
  - Knuth: Fundamental Algorithms, Addison-Wessley, 1973

Algebra visoka škola za primijenjeno računarstvo

PONAVLJANJE

Zadatak

- Ponoviti gradivo kolegija Programiranje

Algebra visoka škola za primjenjeno računarstvo

ORGANIZACIJA PROJEKTA

Uvodni zadatak

- Riješimo sljedeći zadatak: definirati tip podataka koji omogućava čuvanje podataka o širini i visini pravokutnika. Omogućiti množenje pravokutnika skalarom, te izračun površine, opsega, dijagonale. Omogućiti i iscrtavanje pravokutnika zvjezdicama. U glavnom programu napraviti pravokutnik te demonstrirati rad svih njegovih operacija.
  - Kako ćemo definirati novi tip podataka, ako znamo da moramo čuvati širinu i visinu te neke operacije?
  - Što će biti privatno, a što javno?
  - Kako ćemo postaviti širinu i visinu?

Rješenje – klasa

```
class pravokutnik {
    private:
        int sirina;
        int visina;
    public:
        void inicijaliziraj(int sirina, int visina) {
             this->sirina = sirina;
             this->visina = visina;
        }
        void mnozi_skalarom(int skalar) {
             sirina *= skalar;
             visina *= skalar;
        }
        int povrsina() { return sirina * visina; }
        int opseg() { return 2 * sirina + 2 * visina; }
        double dijagonala() { return sqrt(sirina*sirina + visina*visina); }
        void iscrtaj() {
             for (int i = 0; i < visina; i++) {
                 for (int j = 0; j < sirina; j++) {
                     if (i == 0 || i == visina - 1) cout << "*";
                     else {
                         if (j == 0 || j == sirina - 1) cout << "*";
                         Else cout << ' ';
                     }
                 }
                 cout << endl;
             }
        }
};
```


Rješenje – main

```
pravokutnik p;
p.inicijaliziraj(10, 5);
cout << p.povrsina() << endl;
cout << p.opseg() << endl;
cout << p.dijagonala() << endl;
p.iscrtaj();
p.mnozi_skalarom(2);
p.iscrtaj();
```

Problem broj jedan

- Ovo rješenje radi, ali nije jednostavno za održavanje
  - U netrivijalnim programima često imamo više (desetaka, stotina) klasa i struktura
- Veće zadatke je teško riješiti unutar jedne .cpp datoteke
- Često se C++ projekt dijeli na više datoteka:
  - .h datoteke (zaglavlja) sadržavaju sučelja
  - .cpp datoteke sadržavaju implementacije
- Kompajliraju se samo .cpp datoteke i to na sljedeći način:
  - Na mjesto #include se iskopira sadržaj iz .h datoteke
  - Svaki .cpp se kompajlira neovisno o ostalima
  - Na kraju se svi .cpp linkaju

Rješenje problema broj jedan (1/2)

- Kako bismo bolje organizirali naš kôd, klasu ćemo izdvojiti u dvije nove datoteke:
  - pravokutnik.h će sadržavati samo sučelje klase
  - pravokutnik.cpp će sadržavati implementaciju klase
- U sučelju, metode samo najavimo prototipom, primjerice:

```
void inicijaliziraj(int sirina, int visina);
```

- U implementaciji, ispred naziva metode moramo pisati kojoj klasi pripada, primjerice:

```
void pravokutnik::inicijaliziraj(int sirina, int visina) {
  …
}
```

Rješenje problema broj jedan (2/2)

- Nakon reorganizacije, moramo na ispravan način povezati datoteke:
  - U pravokutnik.cpp ćemo uključiti zaglavlje pravokutnik.h
    - Jer tu definiramo implementaciju tog sučelja
  - U program.cpp ćemo uključiti zaglavlje pravokutnik.h
    - Jer tu koristimo novi tip podataka
- Pravilo:
  - Kad uključujemo standardna zaglavlje, koristimo razlomljene zagrade, primjerice: `#include <string>`
  - Kad uključujemo naša zaglavlje, koristimo dvostruke navodnike, primjerice: `#include "pravokutnik.h"`

Problem broj dva

- Promijenimo početak datoteke program.cpp tako da glasi:

```
#include <string>
#include <string>
#include "pravokutnik.h"
```

- Možemo li ovo kompajlirati?
- Promijenimo početak datoteke program.cpp tako da glasi:

```
#include <string>
#include <string>
#include "pravokutnik.h"
#include "pravokutnik.h"
```

- Možemo li ovo kompajlirati? Zašto ne?

Algebra visoka škola za primijenjeno računarstvo

Rješenje problema broj dva – datoteke zaglavlja

- Zbog kopiranja .h datoteka u .cpp datoteke mogu nastati problemi ako se ista .h datoteka više puta uključi u isti .cpp
  - Izravno ili posredno preko drugih datoteka
- Problem se jednostavno rješava korištenjem *include guard*-ova u .h datotekama

```
#ifndef _MOJAKLASA_H_ // Ako ovaj .h nije uključen.
#define _MOJAKLASA_H_ // Označi da je sad uključen.
class MojaKlasa { … };
#endif
```


KONSTRUKTORI I DESTRUKTORI

Problem broj tri

- Što ako napravimo sljedeće:

```
pravokutnik p;
cout << p.povrsina() << endl;
cout << p.opseg() << endl;
cout << p.dijagonala() << endl;
cout << p.iscrtaj();
p.mnozi_skalarom(2);
cout << p.iscrtaj();
```

- Zašto dobijemo takve brojke?
- Ima li smisla postojanje pravokutnika kojemu nisu definirani širina i visina?
- Možemo li kako spriječiti korištenje takvih neinicijaliziranih pravokutnika?

Konstruktor

- Svaka struktura i klasa može imati konstruktor
  - Metoda koja se poziva prilikom izrade objekta
  - Koristi se za postavljanje početnog stanja objekta
  - Naziv jednak nazivu strukture/klase, ali bez povratne vrijednosti (nema čak niti `void`)
    - Može ih biti više s različitim parametrima
    - Konstruktor bez parametara se naziva *defaultni* konstruktor

Korištenje konstruktora

- Konstruktor se automatski poziva pri kreiranju objekta
- Ako struktura/klasa ima definirano više konstruktora, parametri koje proslijedimo određuju koji će biti pozvan
  - Kao i kod poziva svake funkcije, parametri idu u zagradu
  - Iznimka: ako želimo koristiti *defaultni* konstruktor, ne smijemo pisati zagrade
- Primjeri pozivanja konstruktora:

```
pravokutnik p1;
pravokutnik p2(4);
pravokutnik p3(6, 8);
pravokutnik p4[5];
pravokutnik* p5 = new pravokutnik(4, 6);
```


Rješenje problema broj tri

```
class pravokutnik {
        …
   public:
        pravokutnik(int sirina, int visina);
        pravokutnik(int dimenzije);
        …
};
…
int main() {
   pravokutnik p(10, 5);
   cout << p.povrsina() << endl;
   cout << p.opseg() << endl;
   cout << p.dijagonala() << endl;
   cout << p.iscrtaj();
   p.mnozi_skalarom(2);
   cout << p.iscrtaj();
   return 0;
}
```

Destruktor

- Svaka struktura i klasa može imati destruktor
  - Metoda koja se poziva prilikom uništenja objekta
    - Završetkom funkcije
    - Pozivom `delete`
  - Naziv jednak nazivu strukture/klase s tildom ispred, nema povratne vrijednosti, nema parametara

```
struct pravokutnik {
   …
   ~pravokutnik() {
      cout << "Destruktor je pozvan" << endl;
   }
};
```


KLASA STRINGSTREAM

Klasa `stringstream`

- Do sada smo koristili sljedeće tijekove:
  - `cin`, `cout`, `ifstream`, `ofstream`
- Klasa `stringstream` predstavlja ulazno/izlazni tijek prema međuspremniku znakova:
  - Uključimo zaglavlje: `#include <sstream>`
  - Napravimo varijablu: `stringstream sstr;`
  - U nju upisujemo kao u `cout`: `sstr << "XY" << 22 << endl;`
  - Iz nje čitamo kao iz cin: `sstr >> broj;`
  - Iz nje uzmemo string: `string s = sstr.str();`
  - Čistimo njegov sadržaj:
```
sstr.str("");
sstr.clear();
```


Problem broj četiri i njegovo rješenje

- Želimo omogućiti da iscrtavanje ide ili u konzolu ili u tekstualnu datoteku
  - Kako ćemo to napraviti
- Ideja: promijenimo metodu iscrtaj() tako da ne iscrtava izravno u konzolu, već da vrati string koji predstavlja crtež pravokutnika
  - Moramo promijeniti pravokutnik.h
  - Moramo promijeniti pravokutnik.cpp
  - Moramo promijeniti program.cpp

JOŠ MALO O DATOTEKAMA

Rad s datotekama

- Čitanje i pisanje iz tekstualnih i binarnih datoteka je vrlo česta operacija
- Postupak:
  1. Otvori datoteku (tekstualni ili binarno)
  2. Provjeri uspjeh otvaranja; ako nije uspjelo, završi izvođenje programa!
  3. Koristi datoteku:
     1. Tekstualnu kao tijek (`cout`, `cin`, `getline`)
     2. Binarnu kao niz bajtova (`read()`, `write()`)
  4. Zatvori datoteku

Provjera uspjeha čitanja (ili: kad je kraj datoteke?)

- Za čitanje iz tekstualne datoteke koristimo:

```
getline(dat1, ime[, separator]);
dat1 >> broj;
```

  - Ako čitanje nije uspjelo (npr. ako smo došli do kraja datoteke), operacija čitanja će "vratiti" `false`
- Za čitanje iz binarne datoteke koristimo:

```
dat.read(adresa_bajta, broj_bajtova);
```

  - Ako čitanje nije uspjelo (npr. ako smo došli do kraja datoteke), operacija čitanja će uspješno završiti, ali će "postaviti" tijek na `false`

Zadnjih par savjeta

- Q: Kako bismo proslijedili tijek, vector, pravokutnik, … u funkciju?
  - A: Referencom
- Q: Kako bismo pretvorili `string` u `int` ili `double`?
  - A: `atoi()`, `atof()` vraćaju dobiveni broj ili 0 ako se string ne može pretvoriti u broj
  - A: možemo koristiti `stringstream`

Algebra visoka škola za primijenjeno računarstvo

Primjeri

1. Analizirajte datoteku Co2_emisije.csv tako da za Hrvatsku ispišete ukupnu emisiju CO2 od 1960. do 2008 (sve skupa 49 godina). Separator je točka-zarez.
2. Analizirajte datoteku banke.bin tako da ispišete sve banke i njihove VBDI-jeve.

Duljina u bajtovima	Opis
2	n = duljina naziva banke
n	naziv banke
7	VBDI banke