# Uvod

**Ciljevi:**

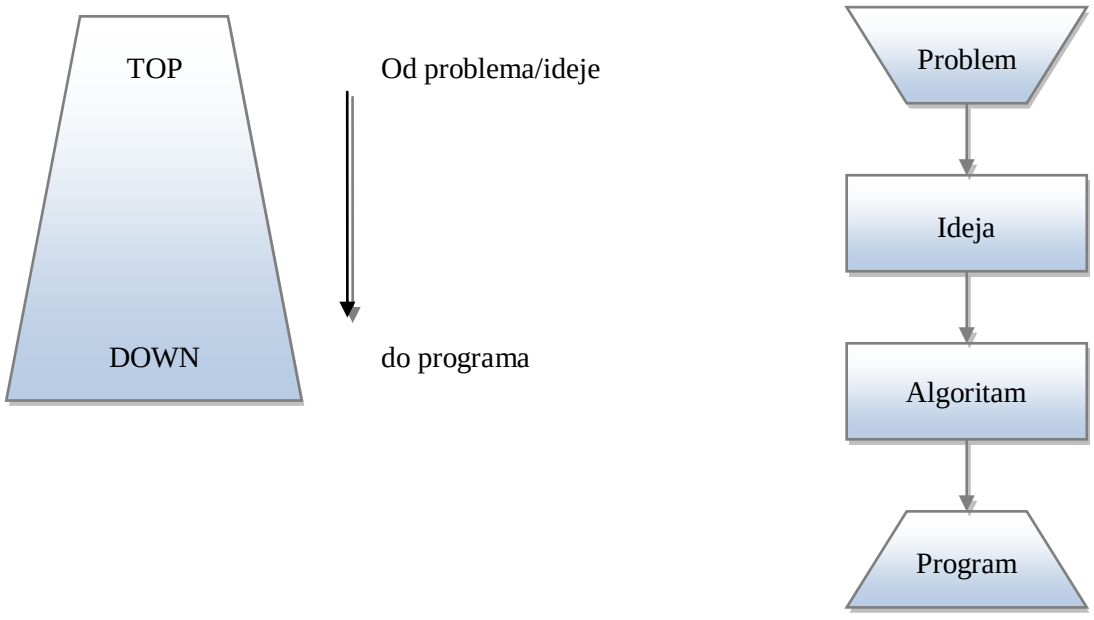
- proučiti i shvatiti pojam programiranja
- shvatiti put problem-ideja-algoritam-program
- shvatiti svaki od pojmova iz prethodne točke zasebno
- izraditi primjer izrade problema, ideja, algoritma i programskog rješenja

**Pregled lekcije**

Na samom početku trebamo se kratko osvrnuti na sam naziv kolegija – Programiranje. Što to znači programirati?

Kratko i jasno možemo reći da je programiranje postupak osmišljavanja i izrade računalnih programa. Programiranje se smatra jednom od najtemeljnijih informatičkih disciplina i svaki informatičar bi trebao znati programirati.

Možemo također reći da je programiranje put od problema i ideje njegovog rješavanja do konačne realizacije rješenja u obliku računalnog programa.




Slika 1: Pristup programiranju

U početku se pred programera postavlja neki problem. Programer razmišlja o problemu, skicira elemente problema i pronalazi ideju za njegovo rješenje. Nakon osmišljene ideje, pristupa se izradi algoritma tj. detaljnog opisa niza konačnog broja koraka koji vode rješenju problema u određenom vremenskom intervalu. Po izradi algoritma pristupa se pisanju programskog koda putem kojeg se izrađeni algoritam pretvara u programski kod tj. u računalni program.

## Problem

Problem je zadatak koji želimo riješiti pomoću računala. Prilikom zadavanja problema moraju se poštivati sljedeća pravila:

- problem mora biti zadan općenito i pokrivati sve slučajeve koji se mogu dogoditi u domeni problema koji se zadaje
- problem mora biti zadan jasno i nedvosmisleno

Da bi problem bio dobro strukturiran njegov opis mora sadržavati sljedeće:

- detaljan opis ulaznih podataka
- opis rezultata i veze između rezultata i ulaznih podataka

Opis ulaznih podataka:

- ulazni se podaci zadaju općenito, tako da obuhvaćaju sve moguće slučajeve
    - broj podataka
    - tip podataka
    - dodatna ograničenja nad podacima

Opis rezultata:

- rezultati se zadaju jasnim opisom onoga u što se žele pretvoriti ulazni podaci
    - tip podataka
    - odnos rezultata prema ulaznim podacima

Za rezultat možemo reći da su to izlazni podaci koji se dobiju iz ulaznih podataka primjenom algoritma.

Primjer – Opis problema

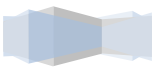
Korisnik unosi neki cijeli broj s tipkovnice (ulazni podaci). Uneseni broj se povećava za 1 i ispisuje se na ekran (rezultat i opis veze ulaz-rezultat).

## Ideja

Ideja predstavlja rezultat procesa proučavanja problema, skiciranja pojedinih elemenata problema i iznalaženja mogućnosti njegovog rješenja. Ideja često ne dolazi odjednom već se polako razvija, ponekad i kroz duže razdoblje ukoliko se radi o velikim projektima i složenim problemima. S druge strane, u nekim slučajevima je ideja u potpunosti trivijalna jer je i sam problem trivijalan.

Primjer – Ideja





Ideja je stvoriti spremnike u koje ćemo pohraniti ulazne podatke, izvršiti traženu operaciju te nakon toga ispisati rezultat korisniku na ekran.

## Algoritam

Algoritam je jasno definiran niz koraka koji vode to rezultata. Možemo također reći da algoritam transformira ulazne podatke u izlazne tj. u konačni rezultat.

Algoritam treba biti definiran prema sljedećim pravilima:

- treba imati 0 ili više ulaza (**Ulaz**)
- treba imati 1 izlaz (ili više) (**Izlaz**)
- svaka instrukcija algoritma treba biti nedvosmislena (**Definitnost**)
- u algoritmu nakon izvršenja svake instrukcije treba biti nedvosmisleno jasno koja se instrukcija sljedeća izvodi (**Determinantnost**)
- algoritam mora biti takav da računalo ili čovjek nakon konačnog broja koraka u konačnom vremenu dođe do rezultata (**Efektivnost**)

Postoji nekoliko načina na koji se algoritam može opisati:

- prirodni jezik
- pseudokod (formatirani opis u prirodnom jeziku)
- kod u programskom jeziku (formalni jezik za opis algoritma na računalu)

Izrada algoritma se zasniva na sljedećem:

- metoda izrade algoritma koja se najčešće koristi naziva se top-down metoda
- izvan računalstva ova se metoda naziva metoda konkretizacije
- algoritam se opiše u 3-7 općenitih instrukcija i u svakom sljedećem koraku se usredotočuje na svaku od opisanih instrukcija koja se potom konkretizira dok god je to potrebno
- za definiciju algoritma se koristi pseudokod koji se detaljizira sve dok se ne može izravno prevesti u kod nekog od programskih jezika

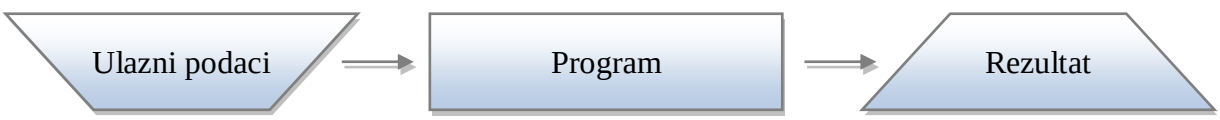
Primjer – Algoritam

```
1. ulaz: pozitivni cijeli broj A
2. izlaz: broj A uvećan za 1
3. ispiši "A="
4. učitaj cijeli broj A
5. uvećaj A za 1
6. ispiši broj A
```

## Program

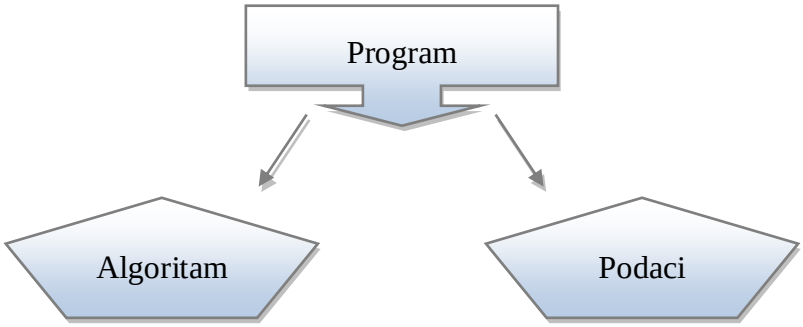
Računalni program možemo definirati kao konačan niz instrukcija računalu koje pretvaraju ulazne podatke u željeni rezultat.






Slika 2: Definicija računalnog programa

Računalni program se realizira kodiranjem algoritma i korištenjem pripadajućih podataka. Dakle možemo postaviti jednadžbu Program = Struktura podataka + Algoritam (N. Wirth)..




Slika 3: Jednadžba programa

Pojam algoritma smo već objasnili a što se tiče podataka možemo reći da je podatak jednostavna i neobrađena činjenica koja ima neko značenje (npr. broj, slovo, ...).

Podatke u programu dijelimo na:

- ulazne podatke
- međurezultate
- izlazne podatke

Svakom podatku treba definirati tip podataka koji predstavlja domenu iz koje taj podatak može poprimiti vrijednost. Uobičajene domene podataka su:

- cijeli brojevi
- pozitivni cijeli brojevi (prirodni brojevi)
- decimalni brojevi
- slova
- riječi
- logičke vrijednosti (da/ne, yes/no, `true`/`false`)

Primjer – Program

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

int main () {
    int A;
    cout << ''Unesite cijeli broj: '' << endl;
    cin >> A;
    A = A + 1;
    cout << ''Rezultat iznosi '' << A << endl;
    return 0;
}
```

UKLJUČUJEMO BIBLIOTEKU FUNKCIJA

UKLJUČUJEMO PROSTOR FUNKCIJSKIH IMENA

POČETAK MAIN FUNKCIJE

NAREDBA ZA ISPIS NA ZASLON, NALAZI SE U IOSTREAM BIBLOTECI

VARIJABLA A TIPA INT

ZBRAJAMO VARIJABLU A I BROJ 1 TE REZULTAT SPREMAMO U VARIJABLU A

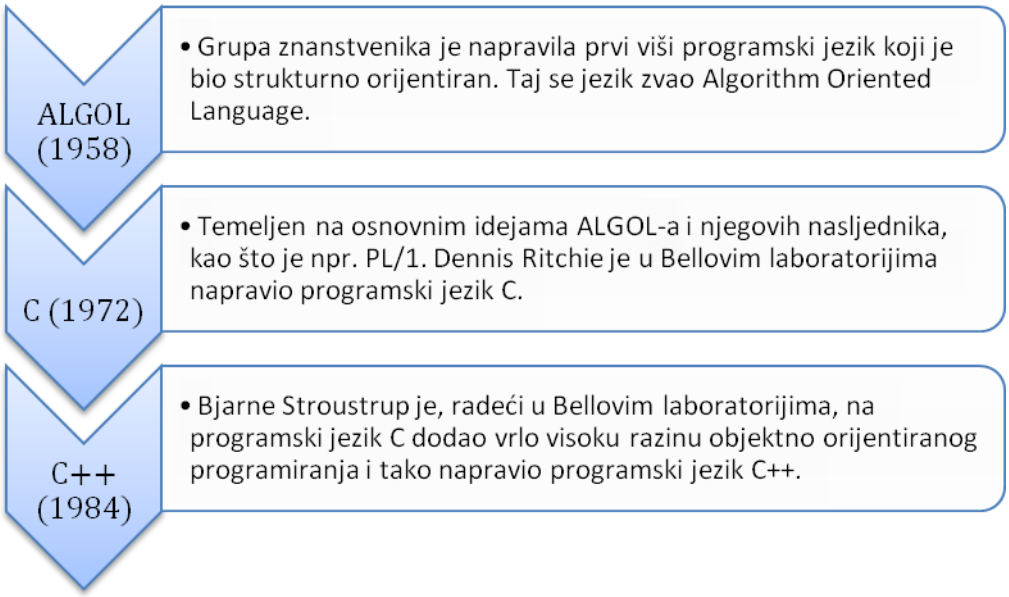
NAREDBA ZA UNOS PODATAKA U VARIJABLU, DAKLE OVDJE UNOSIMO BROJ U VARIJABLU A

ISPISUJEMO SADRŽAJ VARIJABLE A

SIGNAL OPERACIJSKOM SUSTAVU DA JE PROGRAM ZAVRŠIO BEZ PROBLEMA TE DELIMITER ';' KOJI OZNAČAVA KRAJ NAREDBE

KRAJ MAIN FUNKCIJE

## Povijest programskog jezika C++




**Slika 4: Povijesni razvoj programskog jezika C++**

C je programski jezik koji je prvenstveno bio namijenjen za izradu operacijskih sustava. Stoga su mnogi koncepti koje je ALGOL grupa propagirala žrtvovani kako bi se postigla što veća efikasnost jezika. Osim same strukturne čistoće, koja je iz ALGOL-a prenesena u Pascal (Niklaus Wirth, 1971). Svojstva koja je programski jezik C imao jesu

- Niska razina – Mogućnost pisana efikasnih programa i time korištenje programskog jezika C u slučajevima u kojima se prije njega moralo posezati za assemblerom,
- Minimalnost – Zbog efikasnosti u samoj jezgri programskog jezika se nalaze samo najosnovnije naredbe, dok se ostale naredbe po potrebi uključuju u program iz unaprijed pripremljenih biblioteka funkcija,
- Portabilnost – Programski kod napisan u programskom jeziku C može se bez većih izmjena prevoditi i izvršavati na različitim računalnim platformama.

C++ je prvi objektno orijentirani jezik koji je ušao u široku primjenu, C++ je postao de facto standard za objektno orijentirane proceduralne programske jezike. Koncepti koji su dodani programskom jeziku C jesu

- Klase – Koje sadržavaju opis podataka, ali i metode (operacije) nad objektima klase
- Objekti – Instance pojedine klase, u kojima se konkretizira struktura podataka opisana u klasi. Dakle, objekti sadrže podatke opisane u klasi, ali ne i instance metoda.

Klase imaju osnovna svojstva definirana objektnim pristupom:

- Nasljeđivanje – Hijerarhijska organizacija klasa, kao i mogućnost prenošenja dijela strukture podataka i metoda s nadređene klase na podređenu
- Učahurivanje – Definiranje metoda pomoću kojih se vrše sve potrebne operacije nad objektima klase

- Privatnost – Mogućnost definiranja internih podataka klase, koji se ne vide iz okoline objekta, već su dostupne isključivo samom objektu
- Polimorfizam – Mogućnost imenovanja različitih funkcija (s različitim brojem i tipovima podataka) istim imenom.

## Zadaci za vježbu

1. Definirajte jedan problem, ideju, algoritam i program po želji. Program možete opisati vlastitim riječima umjesto kodom.

2. Definirajte Algoritam za sljedeći problem: Korisnik unosi jedan cijeli broj. Broj se umanjuje za 1 i korisniku se ispisuje rezultat.

3. Odgovorite na pitanje što je to problem.

4. Odgovorite na pitanje što je to program.

## Programski primjeri za laboratorijske vježbe

U ovom dijelu nema programskih primjera.