# Upute za natjecatelje

Sastavni dio kompleta zadataka su ove upute te uvodna stranica na kojoj se nalaze mnogi bitni podaci o zadacima i molimo vas da i jedno i drugo vrlo pažljivo pročitate. Na ostalim stranicama se nalaze četiri zadatka. Prilikom rješavanja zadataka preporučuje se korištenje olovke i papira za skiciranje i razradu algoritma.

## Provedba županijskog natjecanja

Nakon završetka natjecanja, članovi županijskih povjerenstava će poslati izvorne kodove vaših rješenja na automatsku evaluaciju. Nakon završetka evaluacije za vašu županiju, rezultati evaluacije biti će dostupni putem sustava za evaluaciju na adresi `https://informatika.fer.hr`.

Ako smatrate da vaša rješenja nisu ispravno evaluirana ili imate bilo kakvih drugih pitanja ili primjedbi obratite se (samostalno ili u dogovoru sa mentorom) državnom povjerenstvu na email adresu `dp@infokup.hr`. Obratite se čak i ako ste svjesni da ste napravili neku od pogrešaka spomenutu u ovim uputama.

Žalbe na rezultate evaluacije možete uputiti direktno županijskom povjerenstvu (koje će vas obavijestiti o roku i načinu predaje žalbi) ili državnom povjerenstvu na adresu `dp@infokup.hr`.

Rok za predaju žalbi državnom povjerenstvu ističe u ponoć na dan natjecanja. Rok žalbi državno povjerenstvo može dodatno produžiti u slučaju većih problema sa zadacima ili sa sustavom za evaluaciju.

## Ulazni i izlazni podaci

Kod svakog pojedinog zadatka obratite pozornost na poglavlja *Ulazni podaci* i *Izlazni podaci*. Tu su definirana pravila vezana za format ulaznih i izlaznih podataka koji mora biti strogo poštovan kako bi vaša rješenja bila točno evaluirana.

Ulaz i izlaz treba se odvijati preko standardnog ulaza i standardnog izlaza. Vaš program sa standardnog ulaza mora očekivati samo ulazne podatke, a na standardni izlaz mora ispisivati samo izlazne podatke. Ako vaš program bude čekao na unos nečeg drugog osim ulaznih podataka ili ispisivao nešto drugo osim izlaznih podataka (npr. `Unesite brojeve...`, `Rješenje je...` i slično), nećete dobiti bodove za taj zadatak, jer evaluator to ne očekuje. Za ilustraciju i bolje razumijevanje pogledajte poglavlje *Primjeri pravilno napisanih programa*.

Vaši programi ne smiju pristupati nikakvim datotekama niti ih kreirati, kršenje ovog pravila može rezultirati gubitkom bodova za taj zadatak. Vaši programi ne smiju pokretati nove procese ili koristiti više od jedne dretve (threada), kršenje ovog pravila može rezultirati gubitkom bodova za taj zadatak.

Prilikom rješavanja nekog zadatka i testiranja njegovog rješenja preporučuje se korištenje *operatora redirekcije ulaza* kako ne biste više puta nepotrebno unosili podatke preko tipkovnice. Na primjer, od ulaznih podataka za neki od test primjera iz teksta zadatka možete napraviti tekstualnu datoteku i testirati vaš program tako da ga pokrećete putem komandne linije na sljedeći način (pretpostavimo da se zadatak zove janje):

```
janje < janje.txt
```

Znak `<` je operator redirekcije ulaza i sve što se nalazi u datoteci janje.txt bit će proslijeđeno vašem programu na isti način kao da je bilo uneseno preko tipkovnice.

## Bodovanje

Obratite pozornost na poglavlje *Bodovanje* u kojem će kod nekih zadataka biti definirano parcijalno bodovanje za pojedine test podatke i/ili skup test podataka koji će zadovoljavati dodatna ograničenja. Dakle, čak i ako neki zadatak ne znate ili ne stignete riješiti u potpunosti, možete dobiti unaprijed poznati

broj bodova za ispravno parcijalno rješenje.

Zadaci ne nose jednak broj bodova. Lakši i brže rješivi zadaci nose manje bodova, a teži zadaci za čije je rješavanje potrebno više vremena, znanja i koncentracije nose više bodova.

Da bi program koji rješava problem iz nekog zadatka dobio maksimalni broj bodova, primijenjeni algoritam mora biti valjan i efikasan tj. brz. Test podaci su unaprijed osmišljeni i koncipirani na način da će programi koji koriste neke manje efikasne, ali valjane algoritme, također dobiti određeni broj bodova (npr. od ukupno 60 bodova, jako loš i spor algoritam će dobiti npr. 20 bodova, dok će dobar algoritam, ali ne i najbolji dobiti npr. 40 bodova). Programi koji će raditi jako brzo za sve test podatke, ali neće davati točne rezultate, naravno, neće donositi bodove. Znači, valjanost algoritma je na prvom mjestu, a brzina izvršavanja na drugom.

## Evaluacija

Bodove za pojedini test podatak će dobiti samo oni programi koji budu generirali točan rezultat unutar navedenog vremenskog i memorijskog ograničenja, te regularno završe svoje izvođenje. Točnije, program se treba izvršiti do kraja. U programskim jezicima C i C++ to znači do return 0; na kraju funkcije `main` koja treba biti deklarirana kao `int main()`, ili do naredbe `exit(0);`. U programskom jeziku Pascal to znači do `end.` ili do naredbe `halt(0)`.

Ne trebate kreirati izvršnu (exe) datoteku. Sustav za evaluaciju će iz vašeg izvornog kôda kreirati izvršnu datoteku na sljedeći način (pretpostavimo da se zadatak zove janje):

| | |
|---|---|
| **C** | `gcc -DEVAL -static -O2 -o janje janje.c -lm` |
| **C++** | `g++ -DEVAL -static -O2 -o janje janje.cpp` |
| **C++11** | `g++ -DEVAL -std=c++11 -static -O2 -o janje janje.cxx` |
| **Pascal** | `fpc -dEVAL -XS -O2 -o janje janje.pas` |

Računalo na kojem se vrši evaluacija je Linux računalo s Ubuntu 12.04LTS 64-bitnim operativnim sustavom i sljedećim verzijama prevoditelja:. gcc 4.8, g++ 4.8, fpc 2.4.

Preporučamo da svoja rješenja obavezno isprobate sa gcc, g++ i fpc prevoditeljima te sa gore navedenim opcijama, posebno ako koristite neki drugi alat (npr. Microsoft Visual Studio) tijekom natjecanja.

Da bi vaše rješenje bilo uspješno prevedeno morate koristiti samo standardne biblioteke, odnosno ne smijete koristiti naredbe i funkcije specifične za Windowse. Vaša rješenja će se evaluirati na računalu s Intel i5 procesorom radnog takta 3.2GHZ. Kako bi mogli procijeniti jesu li su vaša rješenja dovoljno brza uzmite u obzir da se sljedeća dva programa u Pascal-u i C-u prevedena sa navedenim opcijama izvršavaju na navedenom računalu 1.5 i 0.5 sekundi redom (programi računaju ostatak $2^{26}$-og Fibonaccijevog broja pri dijeljenju sa 1000).

```
program fib;
var a, b, i: longint;
begin
  a := 0;
  b := 1;
  for i:=1 to 1 shl 26 do
  begin
    b := (a + b) mod 1000;
    a := (b - a + 1000) mod 1000;
  end;
  writeln(b);
end.
```

```
#include <stdio.h>
int main() {
  int a = 0;
  int b = 1;
  for (int i=0; i<(1<<26); i++) {
    b = (a+b)%1000;
    a = (b-a+1000)%1000;
  }
  printf("%d\n", b);
  return 0;
}
```

## Primjeri pravilno napisanih programa

*Zadatak*: Napišite program koji će zbrojiti i oduzeti dva cijela broja.

*Ulaz*: U prvom retku se nalaze dva cijela broja A i B, međusobno odvojena jednim razmakom.

*Izlaz*: U prvi redak ispišite zbroj, a u drugi redak razliku brojeva A i B.

**Pascal**

```
program p(input,output);
var
  a,b : integer;
begin
  readln(a,b);
  writeln(a+b);
  writeln(a-b);
end.
```

**C**

```
#include <stdio.h>
int main(void) {
  int a,b;
  scanf("%d%d",&a,&b);
  printf("%d\n",a+b);
  printf("%d\n",a-b);
  return 0;
}
```

**C++**

```
#include <iostream>
using namespace std;
int main(void) {
  int a,b;
  cin >> a >> b;
  cout << a+b << endl;
  cout << a-b << endl;
  return 0;
}
```

## Podrška za programski jezik python

Zadatke možete rješavati koristeći programski jezik Python, a podržane su i verzija 2 i verzija 3 jezika. Imajte na umu da nije garantirano da će biti moguće sve zadatke dovoljno efikasno riješiti u Python-u. Naime, vremenska ograničenja su ista za sve jezike, a rješenja u Python-u su vjerojatno puno sporija od ekvivalentnih u drugim dozvoljenim jezicima. Primjerice, program u Python-u koji računaju ostatak $2^{26}$-og Fibonaccijevog broja pri dijeljenju sa 1000 ekvivalentan programima gore se na evaluatoru izvršava 20 sekundi. Za evaluaciju će se koristiti verzije prevoditelja 2.7 odnosno 3.2, a prije izvođenja vaš kod će biti preveden u bytecode naredbom `python -m py_compile janje.py`.

Za zadatke riješene u Python-u potrebno je predati samo izvorni kod. Kako bi evaluator prepoznao koju inačicu Python-a koristite, prva linija u kodu mora točno odgovarati jednom od sljedećeg:

- `#!/usr/bin/python2`
- `#!/usr/bin/python3`

Za testiranje iz komandne linije možete također koristiti operator redirekcije ulaza, međutim potrebno je eksplicitno pozvati odgovarajući prevoditelj. Na primjer:

```
C:\Python27\python janje.py < janje.txt
```

U sljedećoj tablici dani su primjeri pravilno napisanih programa u Python-u za isti zadatak kao gore.

**Python2**

```
#!/usr/bin/python2
a, b = map(int, raw_input().split())
print a+b
print a-b
```

**Python3**

```
#!/usr/bin/python3
a, b = map(int, input().split())
print(a+b)
print(a-b)
```

## Česte pogreške

Donosimo listu čestih i nepotrebnih pogrešaka na natjecanjima ovog tipa. Sve od sljedećeg može rezultirati time da će evaluator dodijeliti nula bodova vašem rješenju.

- Manjak inicijalizacije lokalnih varijabli: U jezicima C i C++ lokalne varijable (što uključuje i varijable deklarirane unutar funkcije main) se ne inicijaliziraju automatski te njihova početna vrijednost nije definirana. Obavezno inicijalizirajte vrijednost svih lokalnih varijabli, inače je moguće da vaš program radi dobro lokalno, a dobije nula bodova na sustavu za evaluaciju.
- Korištenje `uses crt` u programskom jeziku Pascal odnosno `conio.h` u jezicima C i C++ Korištenje sintakse, tipova i funkcija specifičnih za Microsoftove C i C++ prevoditelja, npr. tip podataka `int64`.
- Korištenje sintakse, tipova i funkcija specifičnih za Turbo Pascal ili Borland Delphi.
- Rješenja čekaju na pritisnutu tipku nakon ispisa rezultata.
- Rješenja ispisuju rezultate u krivom formatu, na primjer, ispisuju dva broja svaki u svoj red umjesto u istom redu. Rješenja ispisuju višak podataka na standardi izlaz, na primjer debug informacije.
- Glavna funkcija u jezicima C i C++ je definirana kao void `main()` ili nema `return 0` naredbe.
- Manjak potrebnih include direktiva u jezicima C i C++.
- Korištenje naredbi `itoa` (koje nema pod Linuxom) ili `atoi` (koja radi malo drugačije pod Linuxom). Umjesto njih preporučamo korištenje funkcija `sscanf` i `sprintf`.
- Mijenjanje vrijednosti brojača unutar for petlje u Pascal-u - to nije dozvoljeno u FreePascal-u i uzrokovati će grešku pri prevođenju.
- Pohranjivanje velikih cjelobrojnih vrijednosti u tip integer u Pascal-u. U FreePascalu je integer veličine 16 bita.
- Rješenje alocira premala polja: Ako ne alocirate dovoljno velika polja moguće je da vaš program dobro radi za sve primjere iz zadatka, a dobije nula bodova na evaluatoru. Na primjer, ako sa ulaza trebate pročitati niz od najviše 1000 znakova, u jeziku C potrebno je alocirati polje od najmanje 1001 elementa (`char s[1001]`).
- Rješenje alocira prevelika polja: Nepotrebno glomazna polja mogu uzrokovati da vaše rješenje prekorači memorijsko ograničenje i dobije nula bodova na evaluatoru. Primjerice, u jeziku C polje deklarirano sa `int x[1024][1024][100]` koristi 400MiB memorije.