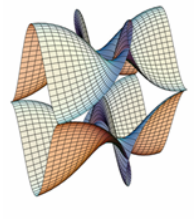

Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu

RAČUNARSKI PRAKTIKUM I

Vježbe 01 - Uvod

v2018/2019.

Sastavio: Zvonimir Bujanović

Gradivo i način polaganja

- Gradivo: osnove jezika C++.

- Način polaganja (detalji na predavanjima):
  - 4 zadaće (4 * 6 bodova)
  - 2 kolokvija (35 + 35 bodova)
  - prisustvo na nastavi / zalaganje (6 bodova)

- Uvjeti za prolaz:
  - ukupno $\geq$ 45 bodova;
  - minimalno po 10 bodova na svakom od kolokvija.
- Moguće je popravljati samo jedan od kolokvija.

- Točnost zadaća se provjerava automatski. Vaše rješenje mora u potpunosti točno raditi za pojedini test da biste dobili bod.
- Kolokviji se pišu na računalu. Točnost se ne provjerava automatski.

Alati koje koristimo

- Službeni operacijski sustav na ovom kolegiju je GNU/Linux. (Vježbati možete i pod Windows-ima.)
- Službeni kompajler je g++ (verzija instalirana u praktikumima).

- Editor teksta: Visual Studio Code, Atom, gedit, Geany, ...

- Nećemo koristiti CodeBlocks da bismo bolje razumjeli proces kompajliranja programa koji se sastoji od više datoteka. Ovo je vrlo važno za zadaće!

Linux -- komandna linija

Većinu ovih naredbi možete izvesti i “klikanjem miša”. Kompajliranje i kretanje po direktorijima svakako treba savladati i unutar terminala.

Unutar terminala:

- Kreiranje novog direktorija – `mkdir`

```
mkdir RP1
```

- Promjena trenutnog direktorija – `cd`

```
cd RP1
cd ..
```

- Izlistavanje sadržaja direktorija – `ls -l`

```
ls -l
```

Linux -- komandna linija

Unutar terminala:

- Brisanje postojeće datoteke – `rm`

```
rm program.cpp
```

- Brisanje *praznog* direktorija – `rmdir`

```
rmdir RP1
```

Linux -- komandna linija

- Kompajliranje programa – iz terminala:

```
g++ program.cpp -std=c++11 -o prog
```

- Pokretanje programa – iz terminala:

```
./prog
```

Standardi jezika C++

- Jezik C++ standardiziran je u više navrata: postoje standardi C++98, C++11, C++14, C++17, a u pripremi je C++2a.
- Na ovom kolegiju koristimo standard C++11.
- Ovaj standard automatski je podržan od verzije 6.1 kompajlera g++. Kod tih kompajlera možemo ispustiti opciju `-std=c++11`:

```
g++ program.cpp -o prog
```

- U praktikumima je instaliran g++ verzije 5.4.1 i potrebna je opcija `-std=c++11`.
- Velika većina naših programa će raditi i po C++98 (upozorit ćemo na iznimke).
- Verziju kompajlera možete doznati pomoću naredbe

```
g++ --version
```

Zadatak 1

1. Napravite direktorij RP1 i unutar njega poddirektorije Vjezbe01, Vjezbe02 i Vjezbe03.
2. Pomoću odabranog editora teksta napišite program koji ispisuje poruku “Hello world!” i spremite ga pod imenom hello.cpp u direktorij RP1/Vjezbe01.
3. U terminalu otiđite u taj direktorij i kompajlirajte program tako da se izvršna verzija zove hello.
4. Pokrenite program.
5. Obrišite izvršnu verziju programa.

Umjesto ponovnog tipkanja naredbi, u terminalu možete koristiti strelicu prema gore.

Naredba cout

Ispisivanje teksta i varijabli pomoću naredbe `printf` u C/C++:

```c
#include <stdio.h>

int main( void )
{
   int a = 5; char str[10] = "abcde";

   printf( "Evo ga: a=%d, ", a );
   printf( "str=%s\nbla", str );

   return 0;
}
```

Rezultat:

```
Evo ga: a=5, str=abcde
bla
```

Naredba cout

Ispisivanje teksta i varijabli pomoću naredbe `cout` u C++:

```
#include <iostream>

using namespace std;

int main( void )
{
   int a = 5; char str[10] = "abcde";

   cout << "Evo ga: a=" << a << ", ";
   cout << "str=" << str << endl << "bla";

   return 0;
}
```

Rezultat:

```
Evo ga: a=5, str=abcde
bla
```

Naredba cin

Želimo učitati:

```
string 3 2.71
```

Pomoću naredbe `scanf` u C/C++:

```c
#include <stdio.h>

int main( void )
{
   int a = 5; char str[10]; float f;

   scanf( "%s %d %f", str, &a, &f );

   return 0;
}
```

Naredba cin

Želimo učitati:

```
string 3 2.71
```

Pomoću naredbe `cin` u C/C++:

```
#include <iostream>

using namespace std;

int main( void )
{
    int a = 5; char str[10]; float f;

    cin >> str >> a >> f;

    return 0;
}
```

using namespace std

- Funkcije iz standardne C++ biblioteke (`cin`, `cout`, ...) se nalaze u "prostoru funkcija" `std`.
- Ako ne navedemo `using namespace std`, trebamo pisati ovako:

```
#include <iostream>

int main( void )
{
    int a = 5; char str[10]; float f;

    std::cin >> str >> a;
    std::cout << "bla bla" << std::endl;

    return 0;
}
```

Deklaracija struktura

Deklaracije strukture `tocka` u C-u:

```
typedef struct
{
    int x, y;
} tocka;
...
tocka P, Q;
```

Deklaracije strukture `tocka` u C++-u:

```
struct tocka
{
    int x, y;
};
...
tocka P, Q;
```

Zadatak 2

1. Deklarirajte strukturu `tocka`.
2. Napišite funkciju koja prima dvije točke i vraća njihovu udaljenost.
3. Napišite funkciju koja prima točku A i vraća točku centralno simetričnu točki A s obzirom na ishodište.
4. U `main`-u sa tipkovnice učitajte koordinate dviju točaka, izračunajte njihovu udaljenost pomoću funkcije i ispišite ju na ekran.

Razdvajanje programa u više datoteka

- Tip podatka `tocka` i opisane funkcije mogu nam trebati više puta.
- Ponovno pisanje deklaracije tipa i implementacija funkcija
  $\rightsquigarrow$ utrošak vremena, povećana mogućnost pogreške.

- Umjesto toga, razdijelit ćemo program na:
  1. sučelje – datoteka koji sadrži deklaraciju tipa podatka `tocka` i pripadnih funkcija.
  2. implementaciju – datoteka koja sadrži sam kod funkcija koje pripadaju tipu podatka `tocka`.
  3. klijentski dio – datoteka koja sadrži glavni dio programa ("main"). Klijentski dio samo koristi tip podatka `tocka` i pripadne funkcije; deklaracije doznaje iz sučelja.

- Ako nam u nekom drugom programu zatreba `tocka`, iskoristit ćemo ranije napisano sučelje i implementaciju. Treba promijeniti samo klijentski dio.
- Ako program koristi više tipova (npr. `tocka` i `duzina`), često za svaki od tipova postoji po jedna datoteka za njegovo sučelje i jedna za implementaciju.

Sučelje

`tocka.h` – sadrži samo deklaracije tipova i funkcija.

```
struct tocka
{
    int x, y;
};

void ispisiTocku( tocka P );
tocka simetricnaTocka( tocka P );
float udaljenost( tocka P, tocka Q );
```

Napomene:

- Ako u ovoj datoteci koristite neke vanjske tipove varijabli, dodajte potrebne `#include` direktive na vrh.
- Ako koristite standardnu biblioteku $\rightsquigarrow$ `using namespace std`. (Preporuka: u `.h` datotekama umjesto toga je bolje koristiti `std::cout` i slično.)

Implementacija

`tocka.cpp` – sadrži implementaciju funkcija iz `tocka.h`.

```
#include <iostream>
#include "tocka.h" // sucelje!

using namespace std;

void ispisiTocku( tocka P )
{
    cout << "(" << P.x << ", ";
    cout << P.y << ")";
}

tocka simetricnaTocka( tocka P ) { ...kod... }
float udaljenost( tocka P, tocka Q ) { ...kod... }
```

Klijentski program

`main.cpp` – koristimo tip `tocka` i funkcije.

```
#include <iostream>
#include "tocka.h" // sucelje!

using namespace std;

int main( void )
{
    tocka P, Q;

    cin >> P.x >> P.y >> Q.x >> Q.y;
    cout << "d(P,Q)=" << udaljenost( P, Q ) << endl;
    tocka simP = simetricnaTocka( P );
    ispisiTocku( simP );

    return 0;
}
```

Kompajliranje i linkanje

- Kompajliraju (prevode u strojne instrukcije) se samo .c/.cpp datoteke (ne i .h).
- Za uspješno kompajliranje moraju biti poznate deklaracije svih tipova i funkcija koje se koriste u pojedinoj datoteci – tome služe `#include` direktive.

- Proces kompajliranja provodimo dodavanjem opcije `-c`:

```
g++ tocka.cpp -std=c++11 -c
g++ main.cpp -std=c++11 -c
```

- Nastale su datoteke `tocka.o` i `main.o`.

Kompajliranje i linkanje

- Linkanje (povezivanje u izvršnu verziju) rezultira programom kojeg možemo pokrenuti iz terminala; linkaju se .o datoteke.

- Za uspješno linkanje nužno je da vrijedi:
  1. funkcija main je u točno jednoj .o datoteci;
  2. poznate su implementacije svih funkcija koje se koriste u svim .o datotekama.

- Uz ranije napravljeno kompajliranje, linkanje radimo ovako:

```
g++ main.o tocka.o -std=c++11 -o program
```

- Kompajliranje i linkanje možemo provesti odjednom:

```
g++ main.cpp tocka.cpp -std=c++11 -o program
```

Zadatak 3

- Napišite sučelje i (neku, bilo koju) implementaciju apstraktnog tipa podatka `stack` (stog cijelih brojeva).
- Sučelje mora biti takvo da se donji klijentski program uspješno kompajlira i izvršava:

```
#include "stack.h"

int main( void )
{
    stack S;

    makeNull( &S );
    push( &S, 3 ); push( &S, 5 );
    int a = top( S ); pop( &S );

    return 0;
}
```