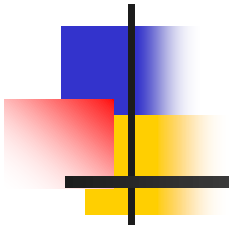
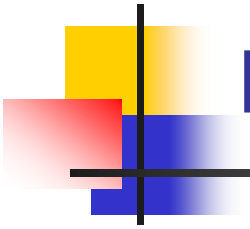


PROGRAMIRANJE I

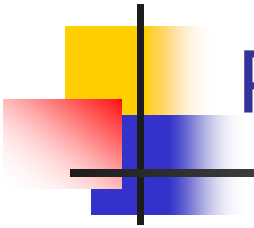
P-10: Modularizacija

prof. dr **Dražen Brđanin**
2023/24



P-10: Modularizacija

- **Sadržaj predavanja**
  - principi modularizacije koda
  - organizacija modularizovanog projekta
  - povezanost identifikatora



Principi modularizacije koda

Modularizacija

- **Modularizacija je postupak kojim se izvorni kôd softverskog projekta organizuje kao kolekcija softverskih modula**
- **Svaki modul smješten je u zasebnoj datoteci**
- **Modularizacija je posebno važna kod razvoja većih softverskih projekata**
- **Dobre strane modularizacije:**
  - **omogućava timski razvoj** – različiti članovi tima mogu istovremeno da razvijaju različite module
  - **omogućava višestruku upotrebu** (engl. *reuse*) **istog koda** – iste module moguće je koristiti u različitim projektima
  - **olakšava otkrivanje grešaka i održavanje softvera**

Osnovni principi modularizacije

- **Prilikom modularizacije treba primjenjivati sljedeće principe:**
  - **funkcionalna dekompozicija**
    - **modul treba da sadrži logičke cjeline** – npr. sve funkcije koje se odnose na rad sa polinomima predstavljaju jednu funkcionalnu, logički povezanu cjelinu, pa ih sve treba smjestiti u jedan modul
  - **razdvajanje interfejsa i implementacije**
    - **interfejs (deklaracije funkcija) smješta se u zaglavlje (modul.h)**
    - **implementacija (definicije funkcija) smješta se u programsku datoteku (modul.c)**

Principi modularizacije koda

Primjena osnovnih principa modularizacije

monolitni program

interfejs

```
#include <stdio.h>
void f();
```

glavna funkcija

```
int main()
{
   f();
   return 0;
}
```

implementacija

```
void f()
{
   // ...
}
```

modularizovani projekat

```
// fajl: modul.h
void f();
```

modul sa interfejsom

```
// fajl: main.c
#include <stdio.h>
#include ”modul.h”
int main()
{
   f();
   return 0;
}
```

modul sa glavnim programom

```
// fajl: modul.c
#include <stdio.h>
void f()
{
   // ...
}
```

modul sa implementacijom

Principi modularizacije koda

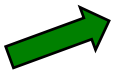
Primjena osnovnih principa modularizacije

```
#include <stdio.h>
```

```
void fA();
void fB();
```

```
int main()
{
   fA();
   return 0;
}
```

```
void fA()
{
   fB();
}
void fB()
{
   // ...
}
```



```
// fajl: a.h
void fA();
```

```
// fajl: b.h
void fB();
```

```
// fajl: main.c
#include <stdio.h>
#include ”a.h”
int main()
{
   fA();
   return 0;
}
```

```
// fajl: a.c
#include <stdio.h>
#include ”b.h”
void fA()
{
   fB();
}
```

```
// fajl: b.c
#include <stdio.h>
void fB()
{
   // ...
}
```

Zaglavlje “a.h” sadrži deklaracije funkcija koje se odnose na funkcionalnost A

Zaglavlje “b.h” sadrži deklaracije funkcija koje se odnose na funkcionalnost B

Modul “main.c” sadrži glavnu funkciju i direktivu za uključivanje zaglavlja “a.h” (jer je za prevođenje main neophodna deklaracija fA)

Modul “a.c” sadrži definicije funkcija koje se odnose na funkcionalnost A i direktivu za uključivanje zaglavlja “b.h” (jer je za prevođenje fA neophodna deklaracija fB)

Organizacija modularizovanog projekta

Datoteke zaglavlja (.h)

- **Za svaku funkcionalnu cjelinu (realizovanu jednom ili više srodnih funkcija) kreira se po jedno zaglavlje**
- **Zaglavlje (može da) sadrži:**
  - **pretprocesorske direktive za uključivanje neophodnih modula** (drugo zaglavlje uključuje se ako je neophodna neka deklaracija ili definicija koja je sadržana u tom modulu)
  - **definicije korisničkih tipova (koji se koriste za realizaciju date funkcionalnosti)**
  - **deklaracije funkcija**

modul.h

```
#include <stdio.h>

#include ”z1.h”
#include ”z2.h”

#ifndef _MODUL_H
#define _MODUL_H

typedef ... TIP;

void f1();
TIP *f2();

#endif
```

Zaglavlja “z1.h” i “z2.h” pripadaju istom projektu i nalaze se u istom folderu kao i modul.h pa se zato navode pod navodnim znakovima

Da se u kompletnom projektu neka definicija ne bi ponovila više puta (prevodilac bi to prijavio kao grešku), pretprocesorskom direktivom #ifndef provjerava se da li je već ranije definisano simboličko ime _MODUL_H i ako nije – definiše se _MODUL_H

```
#ifndef _MODUL_H
#define _MODUL_H
...
#endif
```

U slučaju da u više modula bude uključen fajl “modul.h”, neće biti višestrukih definicija tipa TIP

Organizacija modularizovanog projekta

Glavni program (main.c)

- **Glavni program sadrži:**
  - **pretprocesorske direktive za uključivanje neophodnih modula** (sva potrebna zaglavlja)
  - definicije koje nisu sadržane u uključenim zaglavljima
  - **glavnu funkciju**

main.c

```
#include <stdio.h>

#include ”z1.h”
#include ”z2.h”

int main()
{
   // ...
   return 0;
}
```

Datoteke sa implementacijom funkcija (.c)

- **Svaka implementaciona datoteka sadrži:**
  - **pretprocesorske direktive za uključivanje neophodnih modula**
  - **definicije funkcija**

modul.c

```
#include <stdio.h>
#include <stdlib.h>

#include ”z1.h”
#include ”z2.h”

void f1()
{
   // ...
}
TIP *f2()
{
   TIP *p;
   // ...
   return p;
}
```

Povezanost identifikatora

Definicija povezanosti identifikatora

- **Povezanost identifikatora**
- vezana je za fazu povezivanja (linkovanja) programa
- **koristi se da za određivanje međusobnog odnosa objekata u različitim modulima**
- **daje odgovor na pitanje da li je i kako moguće objekte definisane u jednom modulu koristiti u nekom drugom modulu**, npr. **da li neki globalni identifikator u dva različita modula predstavlja istu ili dvije različite promjenljive**

- **Jezik C razlikuje identifikatore:**
  - **sa spoljašnjom povezanošću**
  - **sa unutrašnjom povezanošću**
  - **bez povezanosti**

Identifikatori bez povezanosti

- Identifikatori bez povezanosti nisu vidljivi prilikom procesa povezivanja i mogu se potpuno nezavisno ponavljati u različitim funkcijama, u istom ili različtim modulima.
- Svaka deklaracija identifikatora bez povezanosti ujedno je i njegova definicija i ona određuje jedinstveni nezavisni objekat.
- **Bez povezanosti su:**
  - **automatske lokalne promjenljive,**
  - **statičke lokalne promjenljive,**
  - **parametri funkcija,**
  - **lokalno definisani korisnički tipovi**
  - **labele**

Npr.

```
int main()
{
   int i;
   return 0;
}
```

Promjenljiva i je lokalna automatska pa i predstavlja identifikator bez povezanosti

Povezanost identifikatora

Spoljašnja povezanost identifikatora

- **Spoljašnja povezanost identifikatora omogućava da se isti objekat (promjenljiva/konstanta/funkcija) koristi u različitim modulima.**
- **Sve deklaracije identifikatora sa spoljašnjom povezanošću u čitavom projektu sa više modula odnose se na jedinstveni objekat** (tj. sve pojave ovakvog identifikatora u različitim modulima odnose se na isti objekat)
- U čitavom projektu može da postoji samo jedna jedina definicija objekta sa spoljašnjom povezanošću (ako je u pitanju promjenljiva, može da postoji samo jedna definiicija sa inicijalizacijom, sve ostalo moraju biti deklaracije)

Spoljašnju povezanost imaju globalne funkcije (ako nisu kvalifikovane sa static)

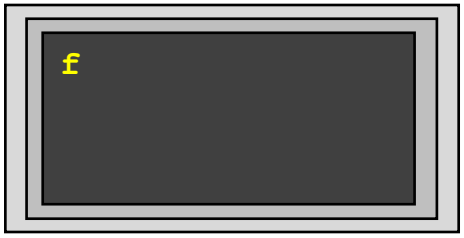
m1.c

```
#include <stdio.h>
void f()
{
   printf("f\n");
}
```

m2.c

```
#include <stdio.h>
void f();
int main()
{
   f();
   return 0;
}
```

Funkcija f je globalna funkcija (i nije kvalifikovana sa static) pa je dostupna u svim modulima u okviru datog projekta



Povezanost identifikatora

Spoljašnja povezanost identifikatora

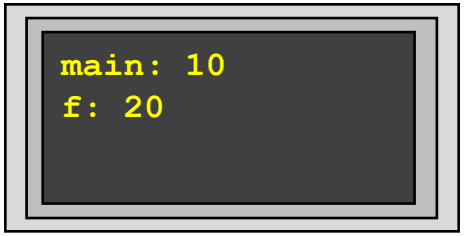
Spoljašnju povezanost imaju globalne promjenljive
(ako nisu kvalifikovane sa static)

m1.c

```
#include <stdio.h>
int n=10;
void f()
{
   printf("f: %d\n", n);
}
```

m2.c

```
#include <stdio.h>
void f();
int n;
int main()
{
   printf("main: %d\n", n);
   n=20;
   f();
   return 0;
}
```



U modulu m1.c navedena je definicija promjenljive n (u pitanju je definicija, jer uključuje i inicijalizaciju promjenljive).

U modulu m2.c navedena je deklaracija promjenljive n.

Pošto je n identifikator sa spoljašnjom poveznošću, riječ je o istoj promjenljivoj, koja se tokom životnog vijeka nalazi u data segmentu.

Početna vrijednost promjenljive n određena je definicijom u modulu m1.c.

Deklaracija promjenljive n u m2.c predstavlja tzv. "uslovnu definiciju" (engl. *Tentative definition*) – ako u nekom drugom modulu nije definisana istoimena promjenljiva, onda bi ova deklaracija predstavljala definiciju.

Povezanost identifikatora

Spoljašnja povezanost identifikatora

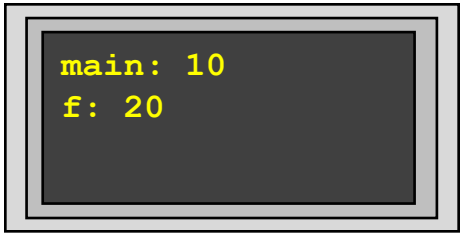
Spoljašnju povezanost imaju lokalne promjenljive kvalifikovane sa extern

m1.c

```
#include <stdio.h>
int n=10;
void f()
{
   printf("f: %d\n", n);
}
```

m2.c

```
#include <stdio.h>
void f();
int main()
{
   extern int n;
   printf("main: %d\n", n);
   n=20;
   f();
   return 0;
}
```



U funkciji main() navedena je deklaracija promjenljive n, kojom je n kvalifikovana kao eksterna (spoljašnja).

Kvalifikacijom extern specifikuje se da se identifikator n odnosi na globalnu promjenljivu koja je definisana u modulu m1.c.

Deklaracije sa extern ne mogu da sadrže inicijalizaciju, npr. NIJE DOZVOLJENO

```
extern int n=20;
```

Da promjenljiva n u funkciji main nije eksplicitno kvalifikovana kao spoljašnja (tj. da je izostavljen extern), tada bi u pitanju bila definicija lokalne promjenljive n koja bi maskirala globalnu promjenljivu n)

Povezanost identifikatora

Unutrašnja povezanost identifikatora

- **Unutrašnju povezanost imaju identifikatori koji su vidljivi samo unutar jednog modula**
- **Unutrašnju povezanost imaju globalni objekti (promjenljive i funkcije) kvalifikovani sa static**
- **Identifikatori sa unutrašnjom povezanošću nisu dostupni u drugim modulima** (stvarni doseg je nivoa datog modula) **i u drugim modulima nezavisno mogu da se koriste istoimeni identifikatori koji reprezentuju druge objekte**

m1.c

```
#include <stdio.h>
static int x=10;
int y=100;
static int f()
{
  printf("f: ");
  printf("x=%d ", x);
  printf("y=%d\n", y);
}
int g()
{ f(); }
```

m2.c

```
#include <stdio.h>
int g();
int x, y;
int main()
{
  printf("main: ");
  printf("x=%d ", x);
  printf("y=%d\n", y);
  g();
  return 0;
}
```

Identifikator y je sa spoljašnjom povezanošću i reprezentuje jednu globalnu promjenljivu koja je vidljiva u čitavom projektu (definisana je u m1.c, početna vrijednost je 100).

U m1.c definisana je globalna promjenljiva x, koja je kvalifikovana sa static, pa ima unutrašnju povezanost, tj. vidljiva je samo u m1.c.

Promjenljiva x, definisana u m2.c, ima spoljašnju povezanost. Pošto u drugim modulima nije definisana istoimena promjenljiva sa spoljašnjom povezanošću, ova deklaracija predstavlja definiciju (poč. vrijednost 0).