Struktura programa

Slide 1

Blokovska struktura jezika (1):

Jedan blok naredbi čini svaki niz naredbi omeđen vitičastim zagradama (npr. tijelo funkcije).

Programski jezik C dozvoljava da se u svakom bloku deklariraju varijable. Deklaracija varijabli unutar bloka mora prethoditi prvoj izvršnoj naredbi u bloku (C90).

Primjer:

```
if(n>0) {
   int i;   /* deklaracija varijable */
   for(i=0; i<n; ++i)
      ....
}
```

Slide 2

Programiranje (C) Ak. god. 2003/04

Blokovska struktura jezika (2):

- Varijabla definirana unutar nekog bloka vidljiva je samo unutar tog bloka.
- Varijabla definirana izvan bloka vidljiva je u unutarnjem bloku ako u njemu nije definirana varijabla istog imena.

Primjer:

```
int main(void) {
  int x,y;
   ......
   if (x>0)
   {
      double y; /* int y nije vidljiv u bloku */
      ....      /* int x je vidljiv u bloku   */
   }
}
```

Slide 3

Blokovska struktura jezika (3):

Formalni argument funkcije vidljiv je unutar funkcije i nije definiran izvan nje. Doseg formalnog argumenta je dakle isti kao i doseg varijable definirane na početku funkcije.

```
int x,y;
......
void f(double x) {
   double y;   /* int x i int y   */
      ....     /* nisu vidljivi   */
}              /* unutar funkcije */
int main(void)
{ ....
}
```

Slide 4

M. Jurak pred6.tex

Atributi varijable

Sve varijable imaju tri atributa: tip, doseg (eng. *scope*) i vijek trajanja.

- Prema tipu imamo varijable tipa `int`, `float`, `char` itd.
- Po dosegu varijable se dijele na lokalne i globalne.
- Prema vijeku trajanja mogu biti automatska i statička.

Slide 5

Automatske varijable (1):

- Svaka varijabla kreirana unutar nekog bloka (dakle unutar neke funkcije) koja nije deklarirana s ključnom riječi `static` je automatska varijable.
- Automatske varijable se kreiraju na ulasku u blok u kome su deklarirane i uništavaju na izlazu iz bloka. Memorija koju je automatska varijabla zauzimala oslobadja se za druge varijable.

Primjer:

```
......
void f(double x) {
   double y=2.71;
   static double z;
      ....
}
```

Slide 6

Programiranje (C) Ak. god. 2003/04

Automatske varijable (2):

- Automatske varijable mogu se inicijalizirati (kao što je to slučaj s varijablom `y`). Inicijalizacija se vrši pri svakom novom ulazu u blok u kome je varijabla definirana.
- Automatska varijabla koja nije inicijalizirana na neki način na ulasku u blok u kome je definirana dobiva nepredvidljivu vrijednost.
- Inicijalizaciju je, osim konstantnim izrazom, moguće izvršiti i izrazom koji nije konstantan, kao u ovom slučaju:

```
void f(double x, int n) {
   double y=n*x;
      ....
}
```

Slide 7

Identifikatori memorijske klase

`auto`, `extern`, `static` i `register`.

Identifikator memorijska klase postavlja se u dekelaraciji varijable prije identifikatora tipa varijable, tako da je opći oblik deklaracije varijable:

```
identifikator_mem_klase tip_varijable ime_varijable;
```

Na primjer,

```
extern double l;
static char polje[10];
auto int *pi;
register int z;
```

Slide 8

M. Jurak pred6.tex

- `auto` : deklarira automatsku varijablu. Budući da su sve varijable definirane unutar nekog bloka, bez ključne riječi `static` automatske, a sve varijable definirane izvan svih blokova statičke, to se ključna riječ `auto` ne mora nikad koristiti.
- Identifikator memorijske klase `register` može se primijeniti samo na automatske varijable. Na primjer

```
f(register int m, register long n)
{
   register int i;
   ........
}
```

- `register` sugerira prevodiocu da varijablu smjesti u registar mikroprocesora umjesto u memoriju.

Slide 9

Statičke varijable (1):

- Kreiraju se na početku izvršavanja programa i uništavaju se tek nakon završetka programa.
- Ukoliko nisu inicijalizirane eksplicitno prevodilac će ih inicijalizirati nulom.
- Statičke je varijable moguće inicijalizirati samo konstantnim izrazima.

```
int f(int j)
{
   static int i=j;  /* greska */
   ........
}
```

- Varijabla definirana izvan svih funkcija je statička. Varijabla deklarirana u nekom bloku s ključnom riječi `static` je statička.

Slide 10

Programiranje (C) Ak. god. 2003/04

Statičke varijable (2):

- Statička varijabla deklarirana unutar nekog bloka inicijalizira se samo jednom i to pri prvom ulazu u blok.

Primjer. Želimo napisati program koji ispisuje prvih 20 Fibonaccijevih brojeva:

$$F_i = F_{i-1} + F_{i-2}, \quad (i = 3, 4, \ldots) \quad F_1 = F_2 = 1.$$

```
#include <stdio.h>
long fibonacci(int);
int main(void) {
   int i;
   for(i=1;i<=20;i++)
       printf("\n i= %d, F= %ld",i,fibonacci(i));
   return 0; }
```

Slide 11

```
long fibonacci(int i)
{
    static long f1=1, f2=1;
    long f;

    f=(i<3) ? 1 : f1+f2;
    f2=f1; f1=f;
    return f;
}
```

Statičke varijable `f1` i `f2` bit će inicijalizirane jedinicama samo pri prvom pozivu funkcije `fibonacci`. Izmedju svaka dva poziva funkciji `fibonacci` one zadržavaju svoju vrijednost i stoga pri i-tom pozivu funkcije imamo `f1`$= F_{i-1}$ i `f2`$= F_{i-2}$.

Slide 12

M. Jurak pred6.tex

Doseg varijable

- Doseg varijable je područje u kojem je varijabla dostupna (ponekad se kaže da je ime varijable vidljivo).
- Prema dosegu varijable se dijele na lokalne i globalne.
- Svaka varijabla definirana unutar nekog bloka je lokalna varijabla za taj blok. Ona nije definirana izvan tog bloka čak i kad je statička.
- Statička lokalna varijabla postoji za cijelo vrijeme izvršavanja programa ali se može dohvatiti samo iz bloka u kojem je deklarirana.

Slide 13

Globalne varijable (1):

- Varijabla definirana izvan svih funkcija naziva se globalna varijabla.
- Globalna varijable je vidljiva od mjesta deklaracije do kraja datoteke.
- Svaka funkcija može doseći globalnu varijablu u svom dosegu i promijeniti njenu vrijednost. Na taj načni više funkcija može komunicirati bez upotrebe formalnih argumenta.

Primjer:

Slide 14

Programiranje (C) Ak. god. 2003/04

```
#include <stdio.h>
#include <ctype.h>
char string[64];  /* globalna varijabla  */
void ucitaj(void);
void malo_u_veliko(void);
void ispisi(void);

int main(void) { ucitaj(); malo_u_veliko(); ispisi(); return 0;}
void ucitaj() { fgets(string,sizeof(string),stdin); }
void malo_u_veliko() {
    int i; for(i=0;string[i] !='\0';i++)
                 string[i]=toupper(string[i]);
}
void ispisi() { printf("%s\n",string); }
```

Slide 15

Globalne varijable (2):

Varijabla deklarirana izvan svih blokova vidljiva je od mjesta svoje deklaracije do kraja datoteke u kojoj se nalazi. Stoga globalne varijable deklariramo na početku datoteke, prije svih funkcija.

Primjer:

```
int a;
void f(int);
int main(void) {
   ....
}
int b;
void f(int i) {
  .....
}
```

Slide 16

M. Jurak pred6.tex

Program smješten u više datoteka:

- C program može biti smješten u više datoteka. Na primjer, svaka funkcija definirana u programu može biti smještena u zasebnu `.c` datoteku.
- Globalne varijable i funkcije definirane u jednoj datoteci mogu se koristiti u drugoj ako su tamo deklarirane.
- Za deklaraciju objekta koji je definiran u drugoj datoteci koristimo ključnu riječ `extern`.

Primjer:

Slide 17

```
#include <stdio.h> /****** Datoteka 1 ******/
int g(int);
void f(int i)
{ printf("i=%d\n",g(i)); }
int g(int i) {
  return 2*i-1;
}


extern void f(int); /****** Datoteka 2 ******/
int main(void) {
   f(3); return 0;
}
```

Slide 18

Vanjski simboli

- U programu smještenom u više datoteka sve funkcije i globalne varijable mogu se koristiti i u drugim datotekama, ako su tamo *deklarirane*. Stoga kažemo da su imena funkcija i globalnih varijabli vanjski simboli.
- Povezivanje deklaracija vanjskih simbola s njihovim definicijama radi linker.
- Kada funkciju ili globalnu varijablu deklariramo s identifikatorom memorijske klase `static`, ona prestaje biti vanjski simbol i može se dohvatiti samo iz datoteke u kojoj je definirana.

Slide 19

Primjer funkcije deklarirane `static`:

```
#include <stdio.h> /****** Datoteka 1 ********/
static int g(int); /* Nije vidljiva izvan
                      Datoteke 1            */

void f(int i) { printf("i=%d\n",g(i)); }

static int g(int i) { return 2*i-1;  }

extern void f(int); /****** Datoteka 2 *******/
extern int g(int);   /* GRESKA, nije vanjski simbol */

int main(void) {
   f(3); printf("g(2)=%d\n",g(2)); return 0;
}
```

Slide 20

Primjer globalne varijable definirane u jednoj datoteci a korištena u drugoj:

```
#include <stdio.h> /******* Datoteka 1 ********/
int z=3;          /* definicija varijable z  */
void f(int i) {
   printf("i=%d\n",i);
}

extern void f(int); /****** Datoteka 2 ********/
extern int z;      /* Deklaracija varijable z  */
int main(void) {
   f(z);
}
```

Kako prevodilac zna što je deklaracija, a što definicija varijable?

Slide 21

Definicija i deklaracija:

Kod globalnih varijabli treba razlikovati definiciju varijable i deklaraciju varijable.

- Prilikom definicije varijable deklarira se njeno ime i tip te se rezervira memorijska lokacija za varijablu.
- Kod deklaracija, samo se deklarira ime i tip varijable dok se podrazumijeva da je varijabla definirana (odnosno da joj je pridružena memorijska lokacija) negdje drugdje. Definicija varijable je uvijek i njena deklaracija.
- Globalna varijabla može imati više deklaracija, ali samo jednu definiciju.

Slide 22

Pravila:

- Prilikom definicije globalna varijabla može biti inicijalizirana konstantnim izrazom.
- Globalna varijabla koja nije eksplicitno inicijalizirana bit će inicijalizirana nulom.
- Pri deklaraciji globalne varijable mora se koristiti ključna riječ `extern`, a inicijalizacija nije moguća.
- Pri definiciji globalnog polja mora biti definirana njegova dimenzija, dok kod deklaracije dimenzija ne mora biti prisutna.

Slide 23

- Oznaka memorijske klase `static` može se primijeniti i na globalne varijable i tada ima isto djelovanje kao i na funkcije. Ona sužava područje djelovanja varijable na datoteku u kojoj je definirana. Ime takve varijable nije vanjski simbol.
- Upozorenje. Oznaka memorijske klase `static` ispred globalne i lokalne varijable ima različito značenje:

```
static int z=3;  /* z nije vidljiv izvan datoteke   */
void f(int i) {
   static double x;  /* x je staticka varijabla */
   ...........
}
```

Slide 24

Datoteke zaglavlja:

Ako se program sastoji od više datoteka, onda se deklaracije vanjskih simbola (varijabli i funkcija) smještaju u datoteku zaglavlja koja se uključuje u sve `.c` datoteke u kojima su te deklaracije potrebne.

Na taj se način osigurava konzistentnost svih deklaracija.

Primjer:

Slide 25

```
extern void f(int);  /******** Datoteka dekl.h  ********/
extern int g(int);
extern int z;

#include <stdio.h> /*********  Datoteka 1 ***********/
#include "dekl.h"
void f(int i) { printf("i=%d\n",g(i)); }
int g(int i) { return 2*i-1;  }

#include "dekl.h" /********** Datoteka 2 *************/
#include <stdio.h>
int main(void) {
   f(3);
   printf("g(2)=%d\n",g(2));
}
```

Slide 26