Pokazivači

Slide 1

- Svakoj varijabli u programu pridružena je memorijska lokacija čija veličina ovisi o tipu varijable.
- Svakoj je memorijskoj lokaciji pridružena jedinstvena adresa.
- Varijabli se može pristupiti
  - putem imena varijable;
  - putem adrese varijable;

Slide 2

Deklaracija pokazivača:

- Pokazivač na `tip` je varijabla koja sadrži adresu varijable tipa `tip`;
- Deklaracija pokazivača:

```
mem_klasa tip * p_var;
```

Primjer:

```
static int * pi;
double  *px;
char*    pc;
int a, *b;
float*  pf, f;
```

Slide 3

Adresni operator (&), operator dereferenciranja (*):

```
int  i=5;
int  *pi=&i;
..........
i=2*(*pi+6);
printf("i= %d, adresa od i= %p\n",i,pi);
```

- `&ime` je adresa varijable `ime`;
- `*p_ime` je vrijednost spremljena u memorijsku lokaciju na koju `p_ime` pokazuje.

Slide 4

Slide 5

Pokazivači i funkcije:

- Pokazivači mogu biti argumenti funkcije. U tom slučaju funkcija može promijeniti vrijednost varijable na koju pokazivač pokazuje.

```
void zamjena(int *x, int *y)
{
     int temp=*x;
     *x=*y;
     *y=temp;
}
```

Slide 6

Programiranje (C) Ak. god. 2003/04

Polje kao argument funkcije:

- Funkcija koja kao argument ima pokazivač na `tip` može uzeti polje tog tipa.

Ispravni su sljedeći pozivi funkcije `f`:

```
char z[100];
void f(char *);
.....
f(z);
f(&z[0]);
f(&z[50]);
```

Slide 7

Operacije nad pokazivačima:

- Aritmetičke operacije dozvoljene nad pokazivačima konzistentne su sa svrhom pokazivača da pokazuju na varijablu određenog tipa.

Svakom pokazivaču moguće je dodati i oduzeti cijeli broj. Ako je `px` pokazivač i `n` varijabla tipa `int`, onda su dozvoljene operacije

```
++px   --px   px+n   px-n
```

Pokazivač `px+n` pokazuje na `n`-ti objekt nakon onog na kog pokazuje `px`.

Primjer:

Slide 8

M. Jurak pred8.tex

```
#include <stdio.h>

int main(void)
{
  float  x[]={1.0,2.0,3.0},*px;
  px=&x[0];
  printf("Vrijednosti: x[0]=%g, x[1]=%g, x[2]=%g\n",
                                  x[0],x[1],x[2]);
  printf("Adrese     : x[0]=%x, x[1]=%x, x[2]=%x\n",
                                   px,px+1,px+2);
  return 0;
}
```

Slide 9

Uspoređivanje pokazivača:

Pokazivače istog tipa možemo međusobno uspoređivati pomoću relacijskih operatora. Takva operacija ima smisla ako pokazivači pokazuju na isto polje:

```
px < py    px > py    px == py   px != py
```

Sljedeće dvije petlje su ekvivalentne:

```
int i,*pi,x[10];
.........
for(i=0;i<10;i++) x[i]=i;

for(pi=&x[0];pi<=&p[9];++pi,++i) *pi=i;
```

Slide 10

Programiranje (C) Ak. god. 2003/04

Pokazivači i cijeli brojevi;

- Pokazivaču nije moguće pridružiti vrijednost cjelobrojnog tipa, osim nule.
- Nula nije legalna adresa; ona označava da pokazivač nije inicijaliziran.

```
double *p=0;
```

ili

```
#define NULL 0
........
double *p=NULL;
```

(simbolička konstanta `NULL` definirana je u `<stdio.h>`).

Slide 11

```
double *px;
..........
if(px != 0) ...  /* O. K. da li je pokazivac
                            inicijaliziran? */
if(px == 0x3451) ... /* GRESKA usporedjivanje
                           s cijelim brojem */
```

Važnost prioriteta i asocijativnosti:

```
 *px += 1;   /*  povecanje vrijednosti
       zbog viseg prioriteta derefrenciranja */
++*px;       /*  povecanje vrijednosti
       zbog asocijativnosti  D -> L unarnih op. */
*px++;      /* inkrementira pokazivac nakon sto
              vrati vrijednost */
```

Slide 12

M. Jurak pred8.tex

Razlika pokazivača:

- Jedan pokazivač može se oduzeti od drugoga ukoliko oni pokazuju na isto polje.
- Ako su `px` i `py` dva pokazivača, tada je `py-px+1` broj elemenata između `px` i `py`, uključujući krajeve.
- Razlika pokazivača je izraz cjelobrojnog tipa, preciznije tipa `ptrdiff_t` definiranog u `<stddef.h>`.

```
int strlen(char *s)
{ /* Implementacijs funkcije strlen() iz string.h */
    char *p=s;

    while(*p != '\0') p++;
    return p-s;
}
```

Slide 13

Primjer: funkcija strcpy

1) Verzija s indeksima:

```
void strcpy(char *s, const char *t)
{
    int i=0;
    while((s[i]=t[i])!='\0') i++;
}
```

2) Verzija s pokazivačima:

```
void strcpy(char *s, char *t)
{
    while((*s=*t)!='\0') {
        s++; t++;
    }
}
```

Slide 14

3) Kraća verzija:

```
void strcpy(char *s, char *t)
{
    while((*s++=*t++)!='\0') ;
}
```

4) Još kraća verzija:

```
void strcpy(char *s, char *t)
{
    while(*s++=*t++) ;
}
```

Slide 15

Generički pokazivač:

Pokazivači na različite tipove podataka ne mogu se pridruživati bez eksplicitne konverzije:

```
char *pc;
int *pi;
.....
pi=pc;          /* GRESKA */
pi=(int *) pc;  /* ISPRAVNO */
```

Pokazivač može biti deklariran kao pokazivač na `void` i tada govorimo o generičkom pokazivaču.

```
void *p;
```

Slide 16

Pokazivač na bilo koji tip može se konvertirati u pokazivač na `void` i obratno.

```
double *pd0,*pd1;
void *p;
.....
p=pd0;   /* ISPRAVNO */
pd1=p;   /* ISPRAVNO */
```

Osnovna uloga generičkog pokazivača je da omogući funkciji da uzme pokazivač na bilo koji tip podatka.

```
double *pd0;
void f(void *);
.....
f(pd0);   /* O.K. */
```

Generički pokazivač se ne može dereferencirati, povećavati i smanjivati.

Slide 17

Pokazivači i jednodimenzionalna polja

- Ime jednodimenzionalnog polja je konstantan pokazivač na prvi elemet polja.

```
char *px,x[128];
px=&x[0];
 /*  ekvivalentno je s */
px=x;
*(px+3)='d';  /* ekvivalentno s x[3]='d'  */
px++;         /* O.K.  */
x++;       /* Greska, x je konstanta pokazivac */
*(x+1)='b';  /* O.K. isto sto i x[1]='b' */
```

Slide 18

Polja pokazivača

```
int  *ppi[10];  /* polje od 10 pokazivaca na int */

int  (*_pi)[10]; /* pokazivac na polje od 10 int-ova */
```

Pokazivač na `char` može biti inicijaliziran stringom. Npr.

```
static char *gradovi[]={ "Osijek",
                         "Split",
                         "Virovitica",
                         "Zagreb"};
```

Slide 19

Pokazivači i višedimenzionalna polja:

Indeksiranje jednodimenzionalnog polja:

```
double x[10];
```

`x[i]` je ekvivalentno s `*(x+i)`

Indeksiranje višedimenzionalnog polja:

```
double x[10][20];
```

`x[i][j]` $\Leftrightarrow$ `*(x[i]+j)` $\Leftrightarrow$ `*(*(x+i)+j)`.

Itd. ....

Deklaracija višedimenzionalnog polja:

```
double x[10][20];
double y[][20];
double (*z)[20];  /* deklaracija <-> definicija !! */
```

Slide 20

Pokazivači i `const`

```
double x[]={0.1,0.2,0.3};
const double y[]={0.1,0.2,0.3};

const double *p1;   // pokazivac na konstantan double
double * const p2=x;  // konstantan pokazivac na double
const double * const p3=y; // konstantan pokazivac na
                           // konstantan double
p1=x;      /* DOZVOLJENO, x ne mogu mijenjati kroz p1 */
p1[1]=4.0; /* NIJE DOZVOLJENO */
p2=&x[2];  /* NIJE DOZVOLJENO  */
p3=&y[2];  /* NIJE DOZVOLJENO  */
*p3=4.0;   /* NIJE DOZVOLJENO */
```

Slide 21

Dinamička alokacija memorije:

Funkcijom `malloc` možemo alocirati memoriju dinamički.

```
void *malloc(size_t n);
```

- `malloc` alocira blok memorije od `n` bajtova;
- Vraća pokazivač na rezervirani blok memorije ili `NULL` ako zahtijev za memorijom nije mogao biti ispunjen.

Primjer:

```
double *p;
 ......
p=(double *) malloc(128*sizeof(double));
if(p==NULL) {
  printf("Greska: alokacija memorije nije uspjela!\n");
  exit(-1);
}
```

Slide 22

Programiranje (C) Ak. god. 2003/04

Delokacija memorije:

Memoriju alociranu pomoću `malloc` treba osloboditi čim više nije potrebna. Taj posao obavlja funkcija `free` koja uzima pokazivač na početak alocirane memorije:

```
double *p;
 ......
p=(double *) malloc(128*sizeof(double));
if(p==NULL) {
  printf("Greska: alokacija memorije nije uspjela!\n");
  exit(-1);
}
..........
/*  memorija vise nije potrebna  */
free(p);
```

Slide 23

Pokazivač na funkciju:

Pokazivač na funkciju deklarira se kao

```
tip_pod (*ime)(tip_1 arg_1,tip_2 arg_2, ...,tip_n arg_n);
```

`ime` je tada pokazivač na funkciju koja uzima n argumenata tipa `tip_1` do `tip_n` i vraća vrijednost tipa `tip_pod`.

Npr:

```
int (*pf)(char c, double a);
```

Primjer:

Slide 24

M. Jurak pred8.tex

```
#include <stdio.h>
#include <math.h>
double integracija(double, double, double (*)(double));

int main(void)
{
    printf("Sinus: %f\n",integracija(0,1,sin));
    printf("Kosinus: %f\n",integracija(0,1,cos));
    return 0;
}

double integracija(double a, double b, double (*f)(double))
{
  return 0.5*(b-a)*((*f)(a)+(*f)(b));
}
```

Slide 25

Argumenti komandne linije:

Programi pod UNIX-om često uzimaju parametre koje nazivamo argumenti komandne linije. Na primjer:

```
cp ime1 ime2
```

Program koji želi iskoristiti argumente komandne linije mora funkciju `main` deklarirati s dva argumenta:

```
int main(int argc, char *argv[])
{ .... }
```

- `argc`: `argc`-1 je broj argumenta komandne linije koji su utipkani pri startanju programa. Ako nema argumenata komandne linije, onda je `argc=1`.
- `argv` je polje pokazivača na parametre komandne linije. `argv[0]` uvijek pokazuje na string koji sadrži ime programa koji se izvršava; `argv[argc]=NULL`.

Slide 26


```
#include <stdio.h> /* program args */
int main(int argc, char *argv[])
{
  int i;
  for(i=0;i<argc; i++)
     printf("%s%s", argv[i],(i<argc-1) ? "," : ".");
  printf("\n");
  return 0;
}
```

Pozovemo li program `args` naredbom

```
args ovo su neki parametri
```

on će ispisati

```
args,ovo,su,neki,parametri.
```

Slide 27

Složene deklaracije:

Što je `p`?

```
int (*p)[10];
int *f(void);
int p(char *a);
int *p(char *a);
int (*p)(char *a);
int (*p(char *a))[10];
int p(char (*a)[]);
int (*p)(char (*a)[]);
int *(*p)(char (*a)[]);
int *(*p[10])(char *a);
```

Slide 28