Preprocesor

Slide 1

- Prije no što prevodilac krene prevoditi izvorni kôd poziva se C preprocesor koji izvršava preprocesorske naredbe.
- Svaka linija izvornog koda koja počinje znakom `#` predstavlja naredbu preprocesoru. Znaku `#` mogu prethoditi bjeline.
- Preprocesorska naredba završava krajem linije (a ne znakom točka-zarez).
- Opći oblik preprocesorskih naredbi je

  ```
  #naredba parametri
  ```

  i one nisu sastavni dio jezika C te ne podliježu sintaksi jezika.
- Neke od preprocesorskih naredbi su

  `#include #define #undef #if #ifdef #ifndef #elif #else`

Slide 2

Programiranje (C) Ak. god. 2003/04

Naredba `#include`

- Naredba `#include` može se pojaviti u dva oblika:

  ```
  #include "ime_datoteke"

  #include <ime_datoteke>
  ```

- U oba slučaja preprocesor će obrisati liniju s `#include` naredbom i uključiti sadržaj datoteke `ime_datoteke` u izvorni kôd, na mjestu `#include` naredbe. Ako je `ime_datoteke` navedeno unutar navodnika, onda preprocesor datoteku traži u direktoriju u kojem se nalazi izvorni program. Ime datoteke navedeno između oštrih zagrada signalizira da se radi o sistemskoj datoteci (kao npr. `stdio.h`), pa će preprocesor datoteku tražiti na mjestu određenom operacijskim sustavom.

Slide 3

Naredba `#define`

```
#define ime tekst_zamjene
```

Preprocesor će od mjesta na kome je `#define` naredba upisana do kraja datoteke svako pojavljivanje imena `ime` zamijeniti s tekstom `tekst_zamjene`. Do zamjene neće doći unutar znakovnih nizova, tj. unutar dvostrukih navodnika. Tako će na primjer dio izvornog kôda

```
#define PI 3.14
......
x=2*r*PI;
printf("PI");
```

prije prevođenja biti zamijenjen s

```
......
x=2*r*3.14;
printf("PI");  /* nema zamjene */
```

Slide 4

M. Jurak pred13.tex

#undef

Definicija nekog imena može se poništiti pomoću `#undef` naredbe.

Primjer:

```
#include <math.h>
/* math.h definira M_PI kao 3.14..... */

#undef M_PI
#define M_PI atan(1.0)
...........
```

Slide 5

Uvjetno uključivanje

Pomoću preprocesorskih naredbi `#if`, `#else`, `#elif` možemo uvjetno uključivati ili isključivati pojedine djelove izvornog koda.

Naredba `#if`:

```
#if uvjet
   blok naredbi
#endif
```

- Ako je `uvjet` ispunjen blok naredbi između `#if uvjet` i `#endif` bit će uključen u izvorni kôd. Ako `uvjet` nije ispunjen blok naredbi između `#if uvjet` i `#endif` biti neće uključen u izvorni kôd.
- Uvjet koji se pojavljuje u `#if` naredbi je konstantan cjelobrojni izraz. Nula se interpretira kao laž, a svaka vrijednost različita od nule kao istina.

Slide 6

Programiranje (C) Ak. god. 2003/04

Primjer:

Najčešće se uključivanje/isključivanje dijela programa čini u ovisnosti o tome da li je neka varijabla definirana ili nije. Tu nam pomaže izraz

```
defined(ime)
```

koji daje 1 ako je `ime` definirano, a 0 ako nije.

```
#if !defined(__datoteka.h__)
#define __datoteka.h__

/* ovdje dolazi datoteka.h  */

#endif
```

To je standardna tehnika kojom se izbjegava višestruko uključivanje `.h` datoteka.

Slide 7

#ifdef, #ifndef

Budući da se konstrukcije

`#if defined` i `#if !defined`

često pojavljuju postoje kraći izrazi s istim značenjem:

`#ifdef` i `#ifndef`

Primjer:

```
#ifndef __datoteka.h__
#define __datoteka.h__

/* ovdje dolazi datoteka.h  */

#endif
```

Zagrade oko varijabli nisu obavezne.

Slide 8

M. Jurak pred13.tex

#else, #elif

- Naredba `#else` ima isto značenje kao u C-u;
- Naredba `#elif` ima značenje `else if`.

Primjer:

```
#if SYSTEM == SYSV
   #define DATOTEKA "sysv.h"
#elif SYSTEM == BSD
   #define DATOTEKA "bsd.h"
#elif SYSTEM == MSDOS
   #define DATOTEKA "msdos.h"
#else
   #define DATOTEKA "default.h"
#endif
```

Slide 9

Primjer: završna i razvojna verzija programa.

```
   ......
   scanf("%d",&x);
 #ifdef DEBUG
   printf("Debug:: x=%d\n",x);  /*  testiranje */
 #endif
```

- Prevoditelji pod UNIX-om obično imaju `-Dsimbol` opciju koja dozvoljava da se `simbol` definira na komandnoj liniji.

Završna verzija:

```
cc -o prog prog.c
```

Razvojna verzija:

```
cc -DDEBUG -o prog prog.c
```

Slide 10

Programiranje (C) Ak. god. 2003/04

Parametrizirane makro naredbe (1)

U parametriziranoj makro naredbi simboličko ime i tekst koji zamijenjuje simboličko ime sadrže argumente koji se definiraju prilikom poziva makro naredbe.

Primjer:

```
#define  max(A,B) ((A)>(B) ? (A) : (B))
```

Ako se u kôdu pojavi naredba

```
x=max(a1,a2);
```

preprocesor će ju zamijeniti s

```
x=((a1)>(a2) ? (a1) : (a2))
```

Slide 11

Parametrizirane makro naredbe (2)

- Argumente makro naredbe treba stavljati u zagrade!

Naredbu

```
x=max(a1+a2,a1-a2);
```

preprocesor će zamijeniti s

```
x=((a1+a2)>(a1-a2) ? (a1+a2) : (a1-a2))
```

- Makro naredba je efikasnija od funkcije jer u njoj nema prenošenja argumenata.

Slide 12

M. Jurak pred13.tex

Razlika između makro naredbe i funkcije

- Ako bismo makro naredbu `max` pozvali na sljedeći način

```
x=max(i++,j++);
```

varijable `i`, `j` ne bi bile inkrementirane samo jednom (kao pri funkcijskom pozivu) već bi veća varijabla bila inkrementirana dva puta.

- Kod makro naredbe nema konrtrole tipa argumenata.
- Neke su "funkcije" deklarirane u `<stdio.h>` ustvari makro naredbe, na primjer `getchar` i `putchar`. Isto tako, funkcije u `<ctype.h>` uglavnom su izvedene kao makro naredbe.

Slide 13

- U `#define` naredbi tekst zamjene se prostire od imena koje definiramo do kraja linije.
- Ako želimo da ime bude zamijenjeno s više linija teksta moramo koristiti kosu crtu (\) na kraju svakog reda osim posljednjeg.

Primjer: makro naredba za inicijalizaciju polja

```
#define INIT(polje, dim)  for(i=0; i < dim; ++i) \
                                 polje[i]=0.0;
```

Slide 14

Assert

- `assert` je makro naredba definirana u `<assert.h>`.
- Makro `assert` koristi se kao funkcija

```
void assert(int izraz)
```

gdje je `izraz` cjelobrojni izraz.

- Ako je `izraz` jednak nuli u trenutku kad se izvršava

```
assert(izraz)
```

`assert` će ispisati poruku

```
Assertion failed: izraz, file ime_datoteke, line br_linije
```

Nakon toga `assert` zaustavlja izvršavanje programa.

Slide 15

Primjer: ako funkcija `f` očekuje pozitivan argument možemo pisati

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

int f(int x) {
   assert(x>0);
   return x;
 }
int main(void) {
    int x=-1;
    printf("x=%d\n",f(x));
    return 0;
}
```

U slučaju negativnog argumenta:

```
Assertion failed: x>0,  file C:\prjC\test\a1.cpp, line 6
```

Slide 16

Isključivanje assert naredbi:

Želimo li isključiti `assert` naredbe iz programa dovoljno je prije uključivanja datoteke zaglavlja `<assert.h>` definirati `NDEBUG` kao u ovom primjeru:

```
#include <stdio.h>
#define NDEBUG
#include <assert.h>
..............
```

Slide 17