C programavimo kalba

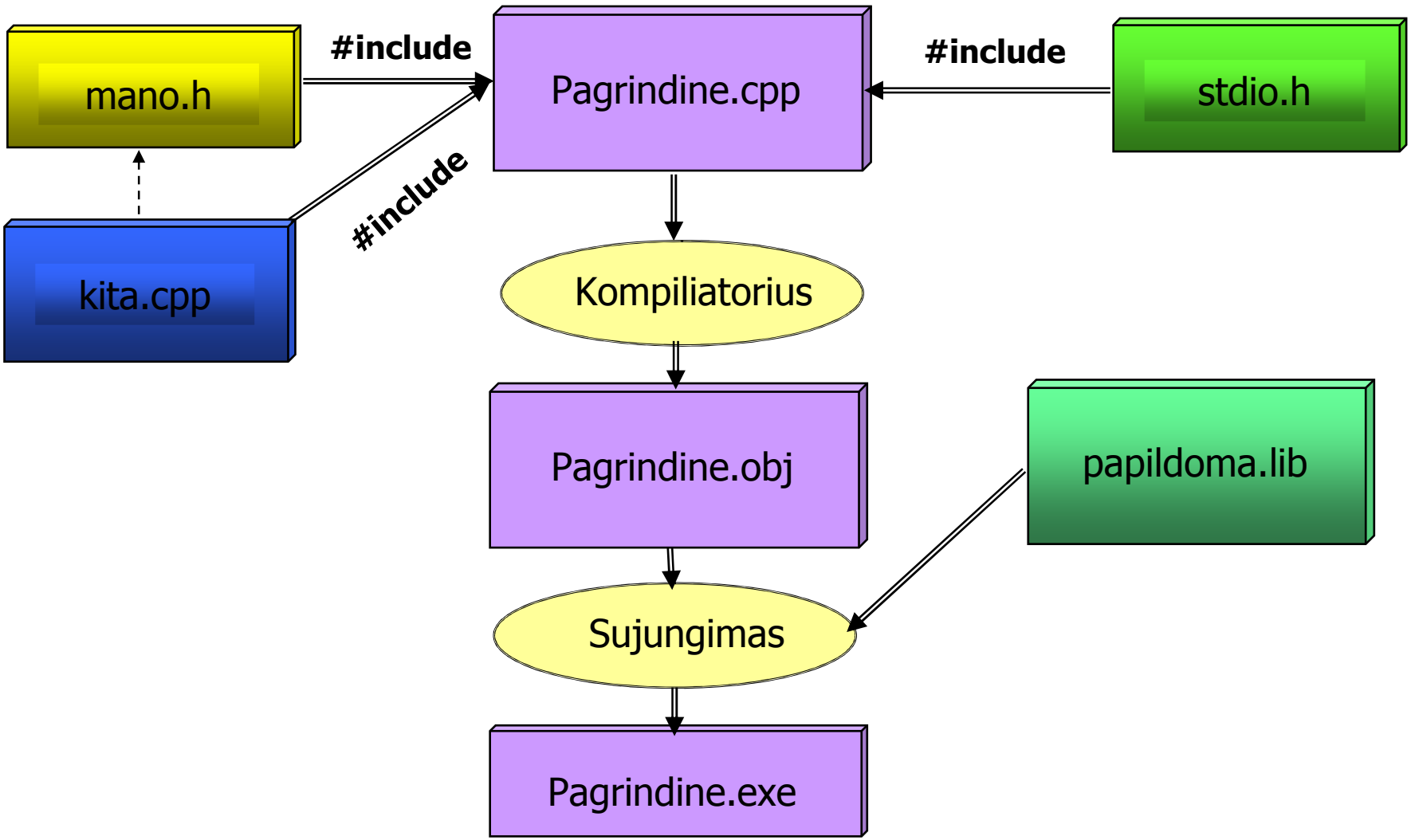
12 paskaita

(Daugiafailinės programos, laiko ir datos funkcijos)

Kodėl programą sudaro daug failų?

- Sukurtos tipinės funkcijų galėtų būti panaudojamos dar kartą;
- Sudaroma aiškesnė programos struktūra;
- Sudaroma galimybė funkcijas apjungti į bibliotekas (.LIB failus) ;
- Leidžia programos kūrime (projekte) dalyvauti keliems programuotojams;

Daugiafailinės programos struktūra



Programa keliuose failuose

Jei programą sudaro keletas failų, jų sujungimas atliekamas pagal tokias taisykles:

- Failai prisijungiami naudojant direktyvą **#include** ;
- Prijungiamo failo būvimo vieta apibrėžiama pagal **#include** reikalavimus;
- Prijungus failą sudaromas vienas globalus programos tekstas, todėl reikia galvoti apie tai ar logiškai galimas ir teisingas funkcijų sujungimas.

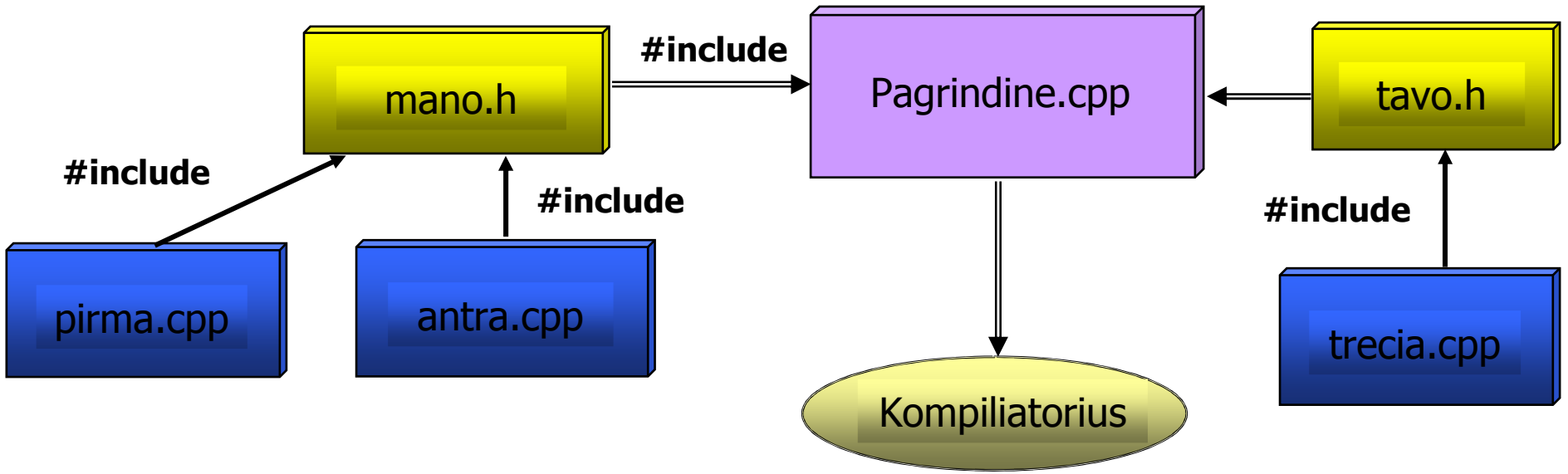
Programos kodo skirstymo į atskirus failus eiga:

1. Programa rašoma, skaidant jas į logiškas procedūras-funkcijas;
2. Procedūros-funkcijos išsaugomos atskiruose failuose, kurių pavadinimai paprastai sutampa (*bet nebūtinai*) su f-jos pavadinimu. Failo pavadinimo plėtinys .cpp arba .h (dažniausiai, nors gali būti ir kitoks).
3. Pagrindinėje programoje failai prisijungiami naudojant direktyvą **#include** "failo_pavadinimas"

Programa keliuose failuose

Galimos failų prijungimo modifikacijos:

- Sukuriamas antraštės failas (*pvz. header.h*) , jame prijungiami visi failai, o pagrindinis failas prisijungia tik *header.h* failą.



Pavyzdys

```
// pagrindine.cpp

#include <stdio.h>
#include <stdlib.h>
#include "mano.h"

#define VAR 10

void main()
{ int k, arr[VAR];
   k = ivedimas();
   srand (k);
   for(int j =0; j < VAR; j++)
   {  arr[j] = rand();
      printf("arr[%d] = %d\n", j, arr[j] );
   }
 pabaiga();
}
```

```
// Failas mano.h

#include "pirmas.cpp"
int ivedimas()
{  int i;
    puts("Ivesk skaiciu");
    scanf("%d",&i);
    return i;
 }
```

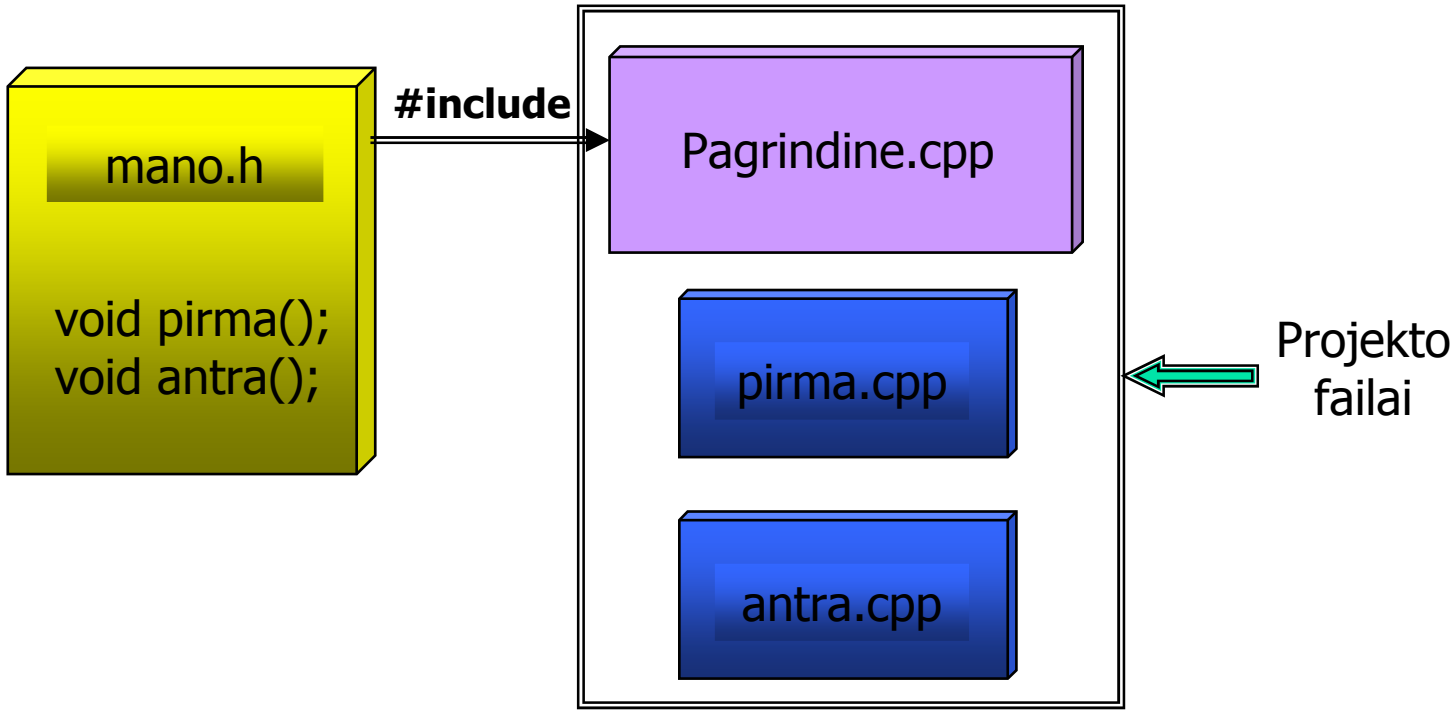
```
// Failas pirmas.cpp

void pabaiga()
{ printf ("Programa baige
            darba sekmingai\n");
  }
```

Programa keliuose failuose

Galimos failų prijungimo modifikacijos (Microsoft Visual C++ atvejis):

- Galima reikiamus .cpp failus įtraukti į projektą .dsp, o su direktyva **#include** prisijungti tik antraštės failą *header.h*, kuriame deklaruojamos visos funkcijos, esančios failuose .cpp



Pavyzdys

```
// pagrindine.cpp

#include <stdio.h>
#include <stdlib.h>
#include "mano.h"

#define VAR 10

void main()
{ int k, arr[VAR];
   k = ivedimas();
   srand (k);
   for(int j =0; j < VAR; j++)
   {  arr[j] = rand();
      printf("arr[%d] = %d\n", j, arr[j] );
   }
 pabaiga();
}
```

```
// Failas mano.h

int ivedimas();
void pabaiga();
```

```
//Failas pirmas.cpp
int ivedimas()
{  int i;
    puts("Ivesk skaiciu");
    scanf("%d",&i);
    return i;
 }
```

```
// Failas antras.cpp

void pabaiga()
{ printf ("Programa baige
            darba sekmingai\n");
 }
```

Laikas ir data

time.h – failas, kuris turi būti prijungtas naudojant datos ir laiko funkcijas.

Laiko skaičiavimui skaičiavimui naudojami tokie duomenų tipai:

- clock_t; time_t; tm

Laiko duomenų tipai clock_t, time_t skirti atvaizduoti sisteminį laiką ir datą, kaip sveiką skaičių, todėl šie duomenų tipai yra long int.

tm – tai struktūra, kurioje gali būti saugoma pilna informacija apie laiką ir datą.

time.h faile apibrėžta makrokomanda CLOCKS_PER_SEC, kuri parodo sisteminio laikrodžio tiksėjimų skaičių per sekundę.

Ji naudojama su funkcija clock(void), kai reikia sužinoti laiką sekundėmis.

Struktūra tm

```
struct  tm {
   int tm_hour;    /* valandos (0 - 23) */
   int tm_isdst;   /* vasaros laikas enabled/disabled */
   int tm_mday;    /* mėnesio diena (1 - 31) */
   int tm_min;     /* minutės (0 - 59) */
   int tm_mon;     /* mėnuo (0 - 11 : 0 = January) */
   int tm_sec;     /* sekundės (0 - 59) */
   int tm_wday;    /* Savaitės diena (0 - 6 : 0 = Sunday) */
   int tm_yday;    /* metų diena (0 - 365) */
   int tm_year;    /* metai po 1900 */
}
```

Struktūros kintamiesiems reikšmės suteikiamos laiko ir datos funkcijų pagalba.

Metų atskaitymas pradedamas nuo **1900**. Pvz. 2000m. tm_year=100;

Jei vasaros laiko kintamojo reikšmė teigiamas skaičius, vadinasi jis įjungtas, jei neigiamas arba 0 – išjungtas.

Laiko ir datos funkcijos

asctime	Keičia tm struktūrą į eilutę
clock	Grąžina laikrodžio tiksėjimų skaičių nuo proceso paleidimo pradžios
ctime	Keičia time_t reikšmę į eilutę
difftime	Grąžina skirtumą tarp dviejų laikų
gmtime	Keičia time_t reikšmę į tm struktūrą (laikas UTC)
localtime	Keičia time_t reikšmę į tm struktūrą (laikas lokalus)
mktime	Keičia tm struktūrą į time_t reikšmę
time	Grąžina dabartinį laiką

Laiko funkcijos

clock_t clock (void);

Grąžina laikrodžio tiksėjimų skaičių nuo proceso paleidimo pradžios**.** Makrokomanda CLOCKS_PER_SEC nustato santykį tarp tiksėjimo ir sekundės (clock ticks per second).

clock_t tipas dažniausiai apibrėžiamas kaip **long int** .

```
#include <stdio.h>
#include <time.h>
void wait ( int seconds )
{ clock_t endwait;
  endwait = clock () + seconds *
                CLOCKS_PER_SEC ;
  while (clock() < endwait)
  { }
}
```

```
void main ()
{  int n;
   printf ("Pradedam skaiciuoti...\n");
   for (n = 10; n > 0; n--)
   {    printf ("%d\n",n);
        wait (2);
   }
  printf ("Startas!!!\n");
}
```

Laiko funkcijos

time_t time (time_t * *timer* **);**

Grąžina esamą (dabartinį) laiką.
Grąžina sekundžių skaičių, praėjusį nuo 00:00, Jan 1, 1970 UTC iki dabar.

Argumentai.
timer

Vieta (rodyklė), kuri gali saugoti grąžintą reikšmę. Jei argumentas NULL, reikšmė nesaugoma, bet vistiek grąžinama. time_t tipas paprastai long int.

```
#include <stdio.h>
#include <time.h>
 void main ()
{
  time_t seconds;
  seconds = time (NULL);
  printf ("%ld dienu prabego nuo 1970.01.01\n", seconds/(3600*24) );
}
```

Laiko funkcijos

double difftime (time_t *timer2*, time_t *timer1*);

Grąžina skirtumą tarp dviejų laikų.

Suskaičiuoja laiko skirtumą sekundėmis tarp laiko *timer1* ir *timer2* .

Argumentai.

timer2 - vėlesnis laikas; *timer1* – ankstesnis laikas

```
#include <stdio.h>
#include <time.h>
void main ()
{ time_t  start, end;    char Input [256];
  double skirtumas;
  time (&start);
  printf ("Iveskite savo varda: ");   gets (Input);
  time (&end);
  skirtumas = difftime (end, start);
  printf ("Labas %s.\n", Input);
  printf ("Uztrukai %.3lf sekundziu ivedinedamas savo varda.\n", skirtumas );  }
```

Laiko funkcijos

struct tm * localtime (const time_t * *timer*);

Keičia time_t reikšmę į tm struktūrą (lokalios laiko juostos laikas).

timer duomenys tranformuojami į tm struktūrą ir kintamieji užpildomi reikšmėmis pagal lokalią laiko juostą.

Argumentai.

timer – rodyklė į time_t kintamąjį, paprastai f-jos time() grąžinama reikšmė.

Grąžinama reikšmė.

Rodyklė į tm struktūrą.

---

char * asctime (const struct tm * *tmptr*);

Keičia tm struktūrą į eilutę.

Keičia duomenis iš struktūros tm į eilutę, kurioje data ir laikas pateikti suprantamai (pvz. Sat May 20 15:21:51 2008).

Eilutės formatas: Www Mmm dd hh:mm:ss yyyy
Bendras eilutės ilgis 26 simboliai.

Pavyzdys

```
/* localtime() ir asctime()  pavyzdys */
#include <stdio.h>
#include <time.h>

int main ()
{
  time_t rawtime;
  struct tm * timeinfo;

  time ( &rawtime );
  timeinfo = localtime ( &rawtime );
  printf ( "Dabartinė data ir laikas:  %s", asctime (timeinfo) );

  return 0;
}
```

Ekrane → Dabartinė data ir laikas: Wed Nov 30 16:37:00 2008

Laiko funkcijos

struct tm * gmtime (const time_t * *timer*);

Keičia time_t reikšmę į tm struktūrą (UTC laiko juostos laikas).
timer duomenys tranformuojami į tm struktūrą ir kintamieji užpildomi reikšmėmis pagal UTC laiko juostą.

Argumentai.
timer – rodyklė į time_t kintamąjį, paprastai f-jos time grąžinama reikšmė.

Grąžinama reikšmė.
Rodyklė į tm struktūrą.

```
#include <stdio.h>
#include <time.h>
void main ()
{ time_t rawtime;   tm * ptm;
  time ( &rawtime );
  ptm = gmtime ( &rawtime );
  printf ("Laikas Los Angeles: %2d:%02d\n", ptm->tm_hour-8, ptm->tm_min);
 printf ("Laikas Hong Konge:   %2d:%02d\n", ptm->tm_hour+8, ptm->tm_min);
}
```

Laiko funkcijos

time_t mktime (struct tm * *ptm*);

Keičia tm struktūrą į time_t reikšmę.
Tikrina tm struktūros kintamojo į kurį rodo rodyklė *ptm* laukus ir suskaičiuoja bei užpildo trūkstamų laukų reikšmes arba koreguoja esamas, jei reikšmė nepatenka į rėžius. Po korekcijų grąžinama time_t reikšmė (t.y. sekundės, praėjusios nuo 1970.01.01).

Argumentas.
ptm – rodyklė į tm struktūros kintamąjį, kurio duomenis reikia papildyti suskaičiuotais.

Pavyzdys

```
* mktime() pavyzdys: savaitės dienos skaičiavimas */
#include <stdio.h>
#include <time.h>
void main ()
{  time_t  rawtime;     struct tm * timeinfo;
   int year, month ,day;
   char * weekday[] = { "Sunday", "Monday", "Tuesday", "Wednesday",
                        "Thursday", "Friday", "Saturday"};

  printf (“Ivesk metus: "); scanf ("%d", &year);
  printf ("Ivesk menesi: "); scanf ("%d", &month);
  printf (“Ivesk diena: "); scanf ("%d", &day);

  time ( &rawtime );
  timeinfo = localtime ( &rawtime );
  timeinfo->tm_year = year - 1900;
  timeinfo->tm_mon = month - 1;      timeinfo->tm_mday = day;

  mktime ( timeinfo );
  printf ("That day is a %s.\n", weekday[timeinfo->tm_wday]);    }
```

Laiko funkcijos

char * ctime (const time_t * *timer*);

Keičia time_t reikšmę į eilutę.

Keičia *timer* reikšmę į eilutę, kurioje data ir laikas pateikti lokalioje laiko juostoje.

Eilutės formatas: Www Mmm dd hh:mm:ss yyyy.

Bendras eilutės ilgis 26 simboliai.

Argumentas

tmptr – rodyklė į time_t reikšmę, kuri paprastai gaunama iš time() funkcijos.

```
#include <stdio.h>
#include <time.h>
void main ()
{  time_t rawtime;
    time ( &rawtime );
    printf ( "Dabartinė data ir laikas: %s", ctime (&rawtime) );
}
```

Ekrane: Dabartinė data ir laikas: Sat May 20 16:05:33 2008