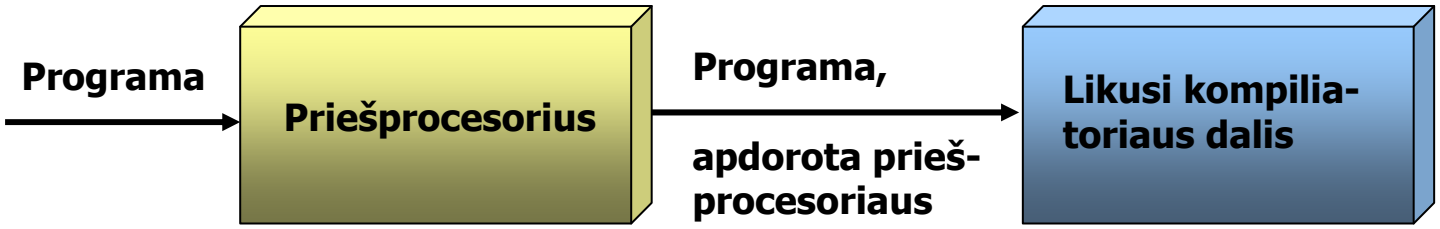
C programavimo kalba

11 paskaita

(Priešprocesorius, direktyvos)

Priešprocesorius

Priešprocesorius - tai integruota C/C++ kompiliatoriaus dalis, kuri atlieka paruošiamuosius veiksmus, prieš pradedant kompiliuoti programą.



Priešprocesorius valdomas direktyvomis, t.y. komandomis, kurios nurodo, kokius veiksmai turi būti atlikti prieš pradedamos programos kompiliavimą.

Direktyvos – tai tam tikras raktinis žodis, kuris pradedamas **#** simboliu.

Direktyvos apdorojamos pirminiame programos kompiliavimo etape, o jų naudojimas apibrėžtas C standarte.

Direktyvos

Direktyvų sąrašas:

#define	#elif	#else	#endif
#error	#if	#ifdef	#ifndef
#include	#line	#pragma	#undef

Direktyva #include skirta nuskaityti ir sukompiliuoti nurodytą failą.

Sintaksė:

```
#include <header_name>      // paieška vykdoma kataloge  ....\include
#include "header_name"      // paieška vykdoma .\   po to  ....\include
#include macro              // makro komandos prijungimas
```

#include

Failas, nurodytas su direktyva #include įtraukiamas kaip papildomas programos tekstas.

Failai, apskliausti < > vadinami antraštės failai (*header files*). Dažniausiai tai specialios paskirties failai, skirti: I/O, matematinėms funkcijoms, laikui ir datai, atminties valdymui, sisteminėms komandoms ir t.t.
Jie turi plėtinius **.h** (stdio.h, math.h, string.h, time.h, stdllib.h, malloc.h,).

Tokie failai saugomi sisteminiame kataloge (dažniausiai ....\include).

Pavyzdys:

```
#include <stdio.h>
#include <stdlib.h>
#include "kopija.txt"          // prijungiamas failas kopija.txt
#include MACRO(x)              // prijungiama makrokomanda
```

#define #undef

Direktyva #define naudojama siekiant apibrėžti identifikatorių arba makrokomandą.

Sintaksė:

#define *pavadinimas* [simbolių_seka]

Įtraukus tokią direktyvą, visur programoje sutiktas *pavadinimas* bus pakeistas simbolių_seka. Pavyzdžiui:

```
#define MAX_WIDTH 100
    char str1[MAX_WIDTH];
#undef MAX_WIDTH                    //  naikinamas identifikatorius
#define MAX_WIDTH 200
int array[MAX_WIDTH];
```

#define gali būti naudojamas makro funkcijos generavimui pvz:

```
#define GETMAX(a, b) a>b?a:b
  int x=5, y;
   y = GETMAX(x, 2);
```

Makrokomanda (Macros)

Macros – tai programinio kodo fragmentas, kuris apibrėžiamas ir veikia kaip funkcija, tačiau egzistuoja tokie skirtumai:

- Macros naudojamas priešprocesoriuje, prieš pradedant kompiliavimą;
- Macros dirba greičiau nei funkcija (*nereikalingas kvietimas*), nors ir pailgina programą;
- Macros'e netikrinami kintamųjų tipai;
- Kitaip nei funkcijos atveju, neįmanoma nustatyti rodyklės į macros, nes tai programos dalis.

```
#include <stdio.h>
#define MULTIPY(x,y) ((x)*(y))
  void main()
{    int a=2, b=3;
      printf("Sandauga = %d\n ", MULTIPLY(a, b));
}
```

Makrokomanda (Macros) tęsinys

Naudojant macros, reikia atkreipti dėmesį į sintaksę:

```
#define SQUARE(x) x*x       // blogai, reikėjo ((x)*(x))
int y = SQUARE(2)           // atsakymas y=2*2 = 4
int z = SQUARE(2 + 1);      // atsakymas z=2+1*2+1=5
```

```
#define BAD (x)  printf("%d\n", x)    // blogai, nes yra tarpas – turėjo būti
                                          BAD(x)  printf("%d\n", x)
```

Macros'e gali būti ir kitos macros funkcijos.

```
#define PI 3.14159
#define SQUARE(x) ((x)*(x))
#define AREA(x) (PI * SQUARE)
```

ANSI apibrėžti Macros (dalis)

Macros	Aprašymas
__DATE__	The **compilation date** of the current source file. The date is a string literal of the form *Mmm.dd.yyyy*. The month name *Mmm* is the same as for dates generated by the library function **asctime** declared in TIME.H.
__FILE__	The **name of the current source file**. **__FILE__** expands to a string surrounded by double quotation marks.
__LINE__	The **line number in the current source file**. The line number is a decimal integer constant. It can be altered with a **#line** directive.
__STDC__	**Indicates full conformance with the ANSI C standard**. Defined as the integer constant 1 only if the /Za compiler option is given and you are not compiling C++ code; otherwise is undefined.
__TIME__	The **most recent compilation time of the current source file**. The time is a string literal of the form *hh:mm:ss*.
__TIMESTAMP__	The **date and time of the last modification of the current source file**, expressed as a string literal in the form *Ddd Mmm Date hh:mm:ss yyyy*, where *Ddd* is the abbreviated day of the week and *Date* is an integer from 1 to 31.

Pavyzdys

```
#include <stdio.h>
char *Date = __DATE__;
char *Time = __TIME__;

int main()
{  FILE* file;
   printf ("Sukurimo data %s ir laikas %s\n", Date, Time);
   file = fopen("duomenys.txt", "r+");

   if (file ==NULL)
   {   printf( "Klaida atidarant faila %s eiluteje %d\n", __FILE__,  __LINE__);
       return 1; }

  return 0;
}
```

#ifdef #ifndef #endif

Direktyva #ifdef leidžia vykdyti programos dalį tik tuo atveju, jei apibprėžta konstanta *name*, nurodyta šalia direktyvos.

#ifdef *name*
// programos dalis
#endif

#**ifdef** MAX_WIDTH
  char str[MAX_WIDTH];
#**endif**

Direktyva #ifndef priešinga #ifdef ir leidžia vykdyti programos dalį tik tuo atveju, kai konstanta *name* neapibrėžta.

#**ifndef** MAX_WIDTH
#**define** MAX_WIDTH 100
#**endif**
char str[MAX_WIDTH];

#if #elif #else #endif

#if direktyva kartus su **#elif**, **#else ir #endif** (*elif = else if*) direktyvomis, kontroliuoja tam tikrų programos dalių kompiliavimą.

Jei išraiška, kuri parašyta po **#if** turi nenulinę reikšmę, eilučių grupė, einanti tuoj pat po **#if** direktyvos siunčiama kompiliuoti, priešingu atveju–po #**else**.

```
#if MAX_WIDTH>200
#undef MAX_WIDTH
#define MAX_WIDTH 200

#elif MAX_WIDTH<50
#undef MAX_WIDTH
#define MAX_WIDTH 50

#else
#undef MAX_WIDTH
#define MAX_WIDTH 100
#endif
char str[MAX_WIDTH];
```

Kitas pavyzdys

```
#ifndef FLAT
#include "failas1.txt"
#else
#include "failas2.txt"
#endif
```

Tęsinys

Naudojant sąlygos direktyvas (#if #ifdef ….) reikia atsiminti:

- kieviena direktyva #if turi būti surišta su #endif;
- direktyvos #elif bei #else nebūtinos;
- sąlyga turi būti sveika konstanta;
- sąlygose gali būti naudojama lyginimo operatoriai == > >= < <=

Sąlygos direktyvose galima naudoti #define bei loginį neigimą (!).

```
#if !defined (OS2)
#define MAX 10
#elif !defined (WIN32)
#define MAX 8
#else
#define MAX 6
#endif
```

Pavyzdys

```
#include "ivedimas.h"
#include <stdlib.h>

#ifndef VAR
#include <stdio.h>
#define VAR 10
#define ivedimas() 4
#endif

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] );
   } }
```

```
#define VAR 10
#include <stdio.h>
int ivedimas()
{  int i; puts("Ivesk skaiciu");
    scanf("%d",&i);
    return i;
}
```

Failas ivedimas.h

#line #error

Direktyva #**line** leidžia pakeisti eilučių numeraciją, kuri naudojama išvedant eilutės numerį įvykus klaidai. Su šia direktyva galima keisti ir failo pavadinimą, kuris bus išvestas įvykus klaidai.

Sintaksė:

#**line** numeris "failo_pav"

Pavyzdys #**line** 1 "blogas kintamasis"
int a?; // klaida, ekrane bus rodoma blogas kintamasis line 1

Direktyva **#error** sustabdo kompiliavimo procesą ir išveda pranešimą, jei randama klaida.

```
#ifndef MYNAME
#error Turi buti apibrezta konstanta MYNAME
#endif
```

Veiksmai, naudojami direktyvose

Priešprocesoriaus direktyvose galima naudoti dvi operacijas t.y. eilutės įstatymą į tekstą (#) ir eilučių sujungimą (##).

Jei prieš macros parametrą panaudojamas simbolis #, tuomet kompiliatorius naudoja argumento vardą (tekstą).

Sujungimo operacija ## sujungia dvi eilutes į vieną, sujugimo metu naikinami tarpai tarp eilučių.

```
#include <stdio.h>
#define SHOW (val) printf (#val "=%d\n", (int)(val))

void main()
{int count = 10;
  SHOW (count);
}
```

Ekrane count=10

Rūšiavimo algoritmai

Rūšiavimo algoritmu vadiname algoritmą, kuris sustato elementus pagal tam tikrą tvarką. Dažniausiai tokia tvarka yra didėjančių skaičių tvarka arba leksikografine (abėcėlės).

Rūšiavimo algoritmai skirstomi pagal:

- skaičiavimo sudėtingumą;
- atminties panaudojimą;
- stabilumą;
- rūšiavimo būdą (įterpimas, sukeitimas, išrinkimas ir t.t.).

Rūšiavimo algoritmai

Burbulo rūšiavimo algoritmas pradedamas nuo duomenų rinkinio pradžios. Lygina du pirmus elemetus. Jei pirmas didesnis už antrą, sukeičia juos vietomis. Taip lyginamos visos skaičių poros iki galo. Po to viskas kartojama nuo pradžių tol, kol neliks nebebus atlikta nei vieno pakeitimo.

Išrinkimo rūšiavimo algoritmas randa duomenų rinkinyje didžiausią elementą ir jį patalpina į galą. Sekančiame žingsnyje ieškomas didžiausias elementas sumažintame rinkinyje, t.y. be prieš tai rasto elemento. Ciklas tęsiamas iki lieka vienas elementas.

Burbulo rūšiavimo algoritmas

```
#include <stdlib.h>
#include <stdio.h>
void Bubble_Sort(int This[], int ub)
{ int indx;   int indx2;  int temp;  int temp2;  int flipped;
  if (ub <= 1)        // ub – rūšiuojamų elementų skaičius
    return;
  indx = 1;
  do
  {    flipped = 0;
    for (indx2 = ub - 1; indx2 >= indx; indx2--)
    { temp = This[indx2];
      temp2 = This[indx2 - 1];
      if (temp2 > temp)
      {
        This[indx2 - 1] = temp;
        This[indx2] = temp2;
        flipped = 1;
      } }
  } while ((++indx < ub) && flipped);}
```

Burbulo rūšiavimo algoritmas

```
#define ARRAY_SIZE 14
int my_array[ARRAY_SIZE];
int main()
{
  int indx;
  for (indx=0; indx < ARRAY_SIZE; indx++)
  {   my_array[indx] = rand();
  }

 Bubble_Sort(my_array, ARRAY_SIZE);
   for (indx=1; indx < ARRAY_SIZE; ++indx)
   {   if (my_array[indx - 1] > my_array[indx])
         {   printf("Blogas rūšiavimas\n");
           return(1);
            }
   }
   return(0);
}
```

Išrinkimo rūšiavimo algoritmas

```
void Selection_Sort(int This[], int the_len)
{  int indx;  int indx2;  int large_pos;   int temp; int large;
    if (the_len <= 1)   // len – elementų skaičius
    return;
  for (indx = the_len - 1; indx > 0; indx--)
  {    /* Randamas didžiausias elementas ir padedamas į galą */
    large = This[0];
    large_pos = 0;
    for (indx2 = 1; indx2 <= indx; indx2++)
    {   temp = This[indx2];
          if (temp >large)
          {  large = temp;
             large_pos = indx2;
           }
    }
    This[large_pos] = This[indx];
    This[indx] = large;
  }
}   // main() funkcija tokia pat, kaip ankstesnėje skaidrėje
```