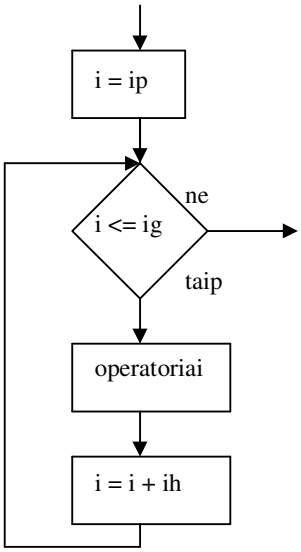
6-7-8 PASKAITOS

Turinys: Ciklai. Sąlygos. Jungiklis. Sąlyginė operacija. *break, continue* ir *goto* operatoriai.

CIKLAI

1. *for* ciklas

for(i = ip; i <=ig; i+= ih) { operatoriai}



1 pavyzdys: skaičių nuo 1 iki 10 kubų skaičiavimas.

```
#include <iostream>
#include <iomanip>  // manipuliatoriui setw
using namespace std;
int main( ){
        int i;
        for( i = 1; i <= 10; i++ ){
                cout<<setw( 5 )<<i;  // setw nustato isvesties lauka 5 simboliams
                int cube = i * i * i;
                cout<<setw( 8 )<<cube<<endl;
        }
        return 0;
}
```

2 pavyzdys: faktorialo skaičiavimas.

```
#include <iostream>
using namespace std;
int main( ){
        unsigned int number;
        unsigned long fact = 1;
```

```
        cout<<"Iveskite skaiciu";
        cin>>number;
        for(int  i = number; i > 0; i- - )
                fact*= i;        //for skliaustai vienam operatoriui apimti - nebutini
        cout<<"Skaiciaus "<<number<<" faktorialas yra "<<fact<<endl;
        return 0;
}
```

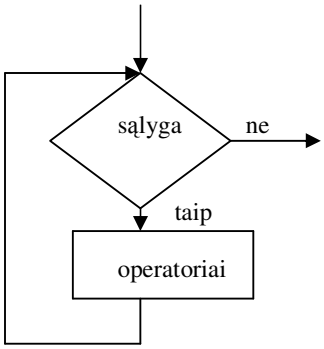
Ciklas gali būti išvis nepradėtas vykdyti.
Kelios *for* galimybės:
1. Galima cikle inicializavimo ir modifikavimo srityse talpinti po kelis operatorius: pavyzdžiui

```
for( k = 1, m = 10; k <= 100; k++, m--) { . . . }
```

2. Galimas begalinis ciklas `for( ; ; ){ . . . }`
3. Ciklas sutapdintas su veiksmais: `for( k = 1, k < max; suma+= k++ );`

2. *while* ciklas

```
while( sąlyga ) { operatoriai }
```



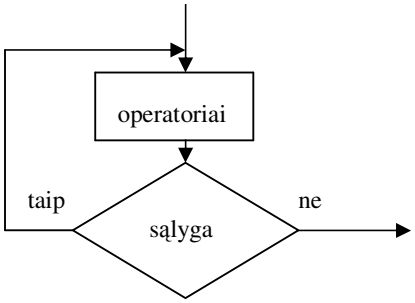
Pavyzdys: skaičiuoti kubus (kaip 1-ame pavyzdyje), kol šie neviršys 1000.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( ){
        int number = 1;
        int cube = 1; // butina inicializuoti
        while(cube < 1000 ){
                cout<<setw( 5 )<<number<<
                        setw( 10 )<<cube<<endl;
                number++;
                cube = number* number* number;
        }
        return 0;
}
```

Ciklas gali būti nepradėtas vykdyti.

3. *do* ciklas

```
do{ operatoriai }
while( sąlyga );
```



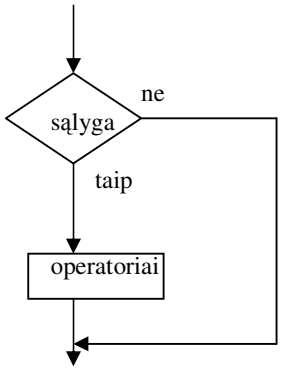
Patogus tada, kai reikalingas dialogas su vartotoju. Pavyzdys: įvesti sveikąjį skaičių, sveikąjį daliklį ir išvesti dalybos rezultatą bei likutį (programa vėliau bus patobulinta – šioje galima dalyba iš 0); programa turi užklausti vartotojo, ar tęsti skaičiavimus.

```
#include <iostream>
using namespace std;
int main( ){
        long number, divisor;
        char c;
        do{
                cout<<"Iveskite skaiciu";
                cin>>number;
                cout<<"Iveskite dalikli";
                cin>>divisor;
                cout<<"Ivestas skaicius: "<<number<<", jo daliklis"<<divisor
                        <<endl;
                cout<<"Dalybos rezultatas: "<<number/divisor<<" likutis: "<<
                        number%divisor<<endl;
                cout<<"Ar testi?: y/n";
                cin>>c;
        } while( c!= 'n' );
        return 0;
}
```

SĄLYGOS

1. *if*

if(sąlyga){ operatoriai }

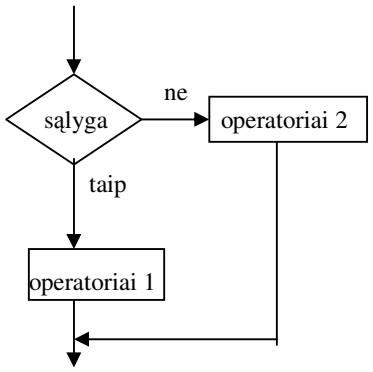


Pavyzdys: ar įvestas skaičius yra natūrinis?

```
#include <iostream>
#include <cstdlib>   // funkcijai exit
using namespace std;
int main( ){
        unsigned int number, div;
        cout<<"Iveskite skaiciu";
        cin>>number;
        cout<<"Ivesta: "<<number;
        for( div = 2; div <= number/2; div++ ){
                if( number%div == 0 ){
                        cout<<"Skaicius ne naturinis: dalijasi is "<<div<<endl;
                        exit( 0 ); // nutraukia programa; 0 – viskas tvarkoj
                }
        }
        cout<<"Skaicius naturinis\n";
        return 0;
}
```

2. ***if - else***

if(sąlyga){ operatoriai 1 }
else{ operatoriai 2}



Pavyzdys: kvadratinės lygties sprendimas.

```
#include <iostream>
#include <cmath>   // funkcijai sqrt
using namespace std;
int main( ){
        double a, b, c;
        cout<<"Iveskite lygties koeficientus";
        cin>>a>>b>>c;
        cout<<"Lygtis: \n";
        cout<<a<<"x^2 + "<<b<<"x + "<<c<<" = 0\n";
        double d = b*b – 4*a*c;
        if( d>= 0 ){
                double x1 = (-b + sqrt( d ))/( 2.*a ),
                        x2 = (-b – sqrt( d))/( 2.*a );
                cout<<"Saknys: "<<x1<<"   "<<x2<<endl;
        }else{
                cout<<"Realiu saknu nera\n",
        }
        return 0;
}
```

Pastaba: algoritmas uždaviniui buvo užrašytas pirmojoje paskaitoje; čia programoje vietoje trijų algoritmo šakų liko tik dvi: šaka "*d == 0*" faktiškai neturi prasmės, nes dviejų *double* formato skaičių lygybė taip negali būti patikrinta.

3. Įdėtinės sąlygos – *else if* konstrukcija
Pavyzdys-schema:

```
. . .
int k;
. . .
if( k > 1000 )
        { // operatoriai bus vykdomi, kai 1000 < k }
else if( k > 100 )
        { //..., kai 100 < k <= 1000 }
else if( k > 10 )
        { //..., kai 10 < k <= 100 }
else
        { //..., kai k <= 10 }
. . .
```

4. **Operatorius *switch***

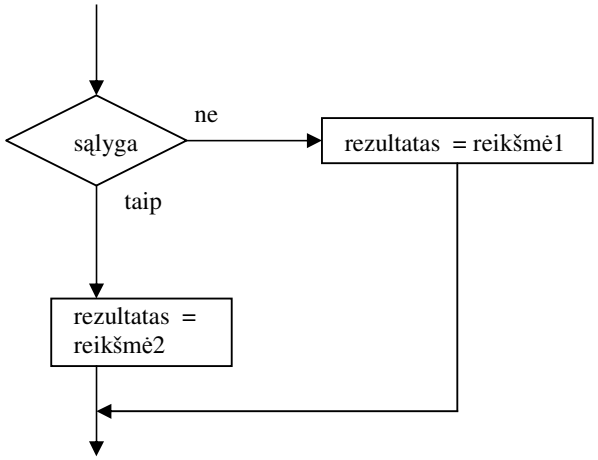
```
switch( kintamasis ) {
        case kr1:
                operatoriai 1;
                [break;]
        case kr2:
                operatoriai 2;
                [break;]
        . . .
        [default:
                operatoriai n;]
}
```

Vykdymas ir reikalavimai: kintamasis gali būti tik sveikojo ir *char* tipų; konstanta kr_i – atitinkamų tipų; *default* ir *break* – nebūtini. Nustačius, kad kintamasis yra kr_i reikšmės, bus vykdoma operatorių grupė *i*. *break* valdymą nusiunčia į pirmąjį operatorių po operatoriaus srities pabaigos. Jei kintamojo reikšmė nesutampa nė su viena *case*'uose išvardintų reikšmių, vykdoma *default* grupė.
Pavyzdys: duota sveikųjų skaičių x ir y lentelė (tokia lentelė gali būti kokio nors žaidimo pagrindas). Joje judama 4 kryptimis per vieną poziciją, įvedus atitinkamą simbolį: į šiaurę – n, pietus – s, vakarus – w ir rytus – e. Judėti pradedama nuo koordinačių pradžios. Judama tol, kol nepaspaudžiamas *Enter* (t.y. simbolis '\r').

```
#include <iostream>
#include <conio.h>  // funkcijai getche
using namespace std;
int main( ){
        int x = 0, y = 0;
        while( (char dir = getche( )) !='\r' ){ // getche nuskaito spaudziamo klaviso
                                                // koda
                switch( dir ){
                        case 'n':
                                y++;
                                break;
                        case 's':
                                y--;
                                break;
                        case 'e':
                                x++;
                                break;
                        case 'w':
                                x--;
                                break;
                        default:
                                cout<<"Netinkamas simbolis. Iveskite dar karta\n";
                }
        }
        return 0;
}
```

5. Sąlyginė operacija

rezultatas = (sąlyga) ? reikšmė1 : reikšmė2;



Pavyzdys-schema: mažesniojo iš dviejų skaičių atrinkimas.

```
. . .
int r, a, b;
. . .
r = ( a < b ) ? a : b;
. . .
```

Operatoriai *break* ir *continue* cikluose

break nutraukia atitinkamą ciklą, *continue* – praleidžia žemiau esančius atitinkamo ciklo operatorius.
Pavyzdys: perrašomas ankstesnis *do* ciklui teiktas pavyzdys, šįkart nepaliekant galimybės dalinti iš nulio.

```
#include <iostream>
using namespace std;
int main( ){
        long number, divisor;
        char c = 'y'; // inicializavimo gali prireikti continue atveju
        do{
                cout<<"Iveskite skaiciu";
                cin>>number;
                cout<<"Iveskite dalikli";
                cin>>divisor;
                cout<<"Ivestas skaicius: "<<number<<", jo daliklis "<<divisor
                        <<endl;
                if( divisor == 0 ){
                        cout<<"Bloga daliklio reiksme!\n";
                        continue; // pastaba 1
```

```
                // break; // pastaba 2
            cout<<"Dalybos rezultatas: "<<number/divisor<<" likutis: "<<
                number%divisor<<endl;
            cout<<"Ar testi?: y/n";
            cin>>c;
        } while( c!= 'n' );
        return 0;
}
```

Pastaba 1: *continue* atveju, jei daliklio reikšmė būtų nulinė, būtų praleisti visi operatoriai iki ciklo galo; ciklas būtų kartojamas, nes *c* inicializuotas reikšme *'y'*.
Pastaba 2: jei *continue* būtų užkomentuotas, o *break* – ne, programa nulinės daliklio reikšmės atveju būtų iš viso nutraukta.

Operatorius *goto*

Vengtinas.
Perduoda valdymą į žyme pažymėtą operatorių. Žymė – tekstas, rašomas prieš reikiamą operatorių, o po jo rašomas *:*. Pavyzdys-schema:

```
. . .
goto Label;
. . .
Label: operatorius;
. . .
```