Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Споредбени и логички изрази

Структурно програмирање

ФИНКИ 2023

Содржина

- Логички тип податоци
- Споредбени изрази и споредбени оператори
- Логички изрази и логички оператори
- Контролни структури за избор
- Наредби за избор: if и if – else
- Условен оператор ?:

Логички тип податоци

- Податоците чија вредност може да биде или **true** или **false** се од **логички тип**
- Во C++, променливи од логички тип се декларираат со помош на клучниот збор **bool**
- Пример:

```
bool dane;
bool semafor = true;
```

Споредбени (релациски) изрази

- Споредбените изрази овозможуваат да се провери релацијата помеѓу аритметичките величини
- Секој споредбен израз е операција над два операнди
- Резултатот е една од логичките вредности: **точно** (**true**) или **неточно** (**false**)

Споредбени (релациски) оператори

Оператори	Синтакса	Пример	Значење
Релациски оператори			
>	>	x > y	x е поголемо од y
<	<	x < y	x е помало од y
	>=	x >= y	x е поголемо или еднакво на y
	<=	x <= y	x е помало или еднакво на y
Оператори за еднаквост			
=/=	==	x == y	x е еднакво на y
	!=	x != y	x не е еднакво на y

Примери за релациски изрази

- Пример 1: **5 > 7**
  - Операнди се **5** и **7**, операторот е **>**
  - Резултат е логичката вредност **неточно**
- Пример 2: **2 + 3 <= 10**
  - Операнди се **2+3** и **10**, операторот е **<=**
  - Резултат е логичката вредност **точно**
- Пример 3: **1 + 8 != 7 + 2**
  - Операнди се изразите **1+8** и **7+2**, операторот е **!=**
  - Резултат е логичката вредност **неточно**

Правила за логичките типови

- Во C++, секоја ненулта вредност има значење „вистина“ (true)
- Вредноста **`0` има значење „невистина“** (`0.0`, `-0`, `+0`)
- При пресметување на сите релациски изрази, резултатот ќе биде 1 ако условот е исполнет, односно `0` ако условот не е исполнет
  - Според тоа, вредноста на **5 > 7** е `0`, додека вредноста на **2 + 3 <= 10** е 1

Примери за релациски изрази (2)

- Примери
  - (7 == 5) враќа невистина (false)
  - (5 > 4) враќа вистина (true)
  - (3 != 2) враќа вистина (true)
  - (6 >= 6) враќа вистина (true)
  - (5 < 5) враќа невистина (false)

Задача:

Нека е a=2, b=3 c=6

(a == 5) враќа:
невистина (false)

(a*b >= c) враќа:
вистина (true)

(b+4 > a*c) враќа:
невистина (false)

((b=2) == a) враќа:
вистина (true)

Логички изрази и оператори

- Во програмите, често се јавува потреба да се комбинираат неколку споредбени изрази
- На пример, треба да провериме дали еден број припаѓа во интервалот **[0, 90)**
  - Бројот треба да е **поголем или еднаков на 0 И помал од 90**
- Во вакви случаи, се прави комбинација на два или повеќе споредбени изрази во логички израз, користејќи ги логичките оператори
  - Решение на примерот: **x >= 0 && x < 90**

Логички оператори

- логичко И - **&&**
  - вредноста на изразот ќе биде вистина (различна од `0`) ако и само ако двата операнди се вистина
- логичко ИЛИ - **||**
  - вредноста на изразот ќе биде вистина (различна од `0`) ако барем еден од двата операнди е вистина
- логичко НЕ - **!**
  - вредноста на изразот ќе биде вистина (различна од `0`) ако операндот има вредност невистина
- логичко ИСКЛУЧИТЕЛНО ИЛИ - **^**
  - вредноста на изразот ќе биде вистина (различна од `0`) ако и само ако едниот операнд е вистина а другиот невистина

Израз	Резултат
`true && false`	`false`
`true || false`	`true`
`!false`	`true`

Логички оператори (2)

- **Приоритетот на операторот** ! е
  - повисок од множење и делење,
  - ист со операторите за инкрементирање и декрементирање
  - понизок од заградите.
- Операторот && има
  - повисок приоритет од операторот ||
  - и двата имаат понизок приоритет од релациските оператори.
- Така изразот

  a > b && b > c || b > d

  ќе биде пресметан како

  ((a > b) && (b > c)) || (b > d)
- Пример со проблеми:
  - 1 < i < 10 секогаш ќе биде вистина. **Зошто?**

Пресметување на логички изрази

- Сите логички изрази се пресметуваат одлево-надесно
- Пресметувањето се врши се додека „не сме сигурни” за вредноста на изразот
- Пример:
  - за i=11 при пресметување на изразот (i<10) && (i>5) ќе се пресмета само вредноста на изразот (i<10) и бидејќи истата е 0, пресметувањето на целиот израз ќе прекине!
- Пример:
  - каква ќе биде вредноста на изразот !i == 5?

Пресметување на логички изрази (2)

- Важат Де Моргановите закони
  - Дистрибуција на негацијата и логичкото И
    - !(m || n) е исто со !m && !n
    - !((a < b) || (b >= c)) е исто со (a >= b) && (b < c)
  - Дистрибуција на негацијата и логичкото ИЛИ
    - !(x && y) е исто со !x || !y
    - !((a <= b) && (b > c)) е исто со (a > b) || (b <= c)

Приоритет и асоцијативност на операторите

- Приоритет
  - Сите унарни оператори имаат повисок приоритет од бинарните
  - Употребата на загради го менува приоритетот
  - Во C++ се дефинирани 17 нивоа
- Асоцијативност
  - За два оператора со ист приоритет, операцијата што треба да се изврши се избира на основа на правилата за асоцијативност на операторите
  - Дефинирани се „одлево надесно” и „оддесно налево”

Ниво	Оператор						Вид	Редослед
Високо								
	()	[]	->	.			Бинарен	Одлево надесно
	+ -	++ --	! ~	* &	sizeof		Унарен	Оддесно налево
	->*	.*					Бинарен	Одлево надесно
	*	/	%				Бинарен	Одлево надесно
	+	-					Бинарен	Одлево надесно
	<<	>>					Бинарен	Одлево надесно
	<	<=	>	>=			Бинарен	Одлево надесно
	==	!=					Бинарен	Одлево надесно
	&						Бинарен	Одлево надесно
	^						Бинарен	Одлево надесно
	|						Бинарен	Одлево надесно
	&&						Бинарен	Одлево надесно
	||						Бинарен	Одлево надесно
	? :						Тернарен	Одлево надесно
	=	+= -=	*= /=	^= %=	&= |=	<<= >>=	Бинарен	Оддесно налево
Ниско	,						Бинарен	Одлево надесно

Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Контролни структури за избор

Структурно програмирање

ФИНКИ 2023

Управувачки (контролни) структури

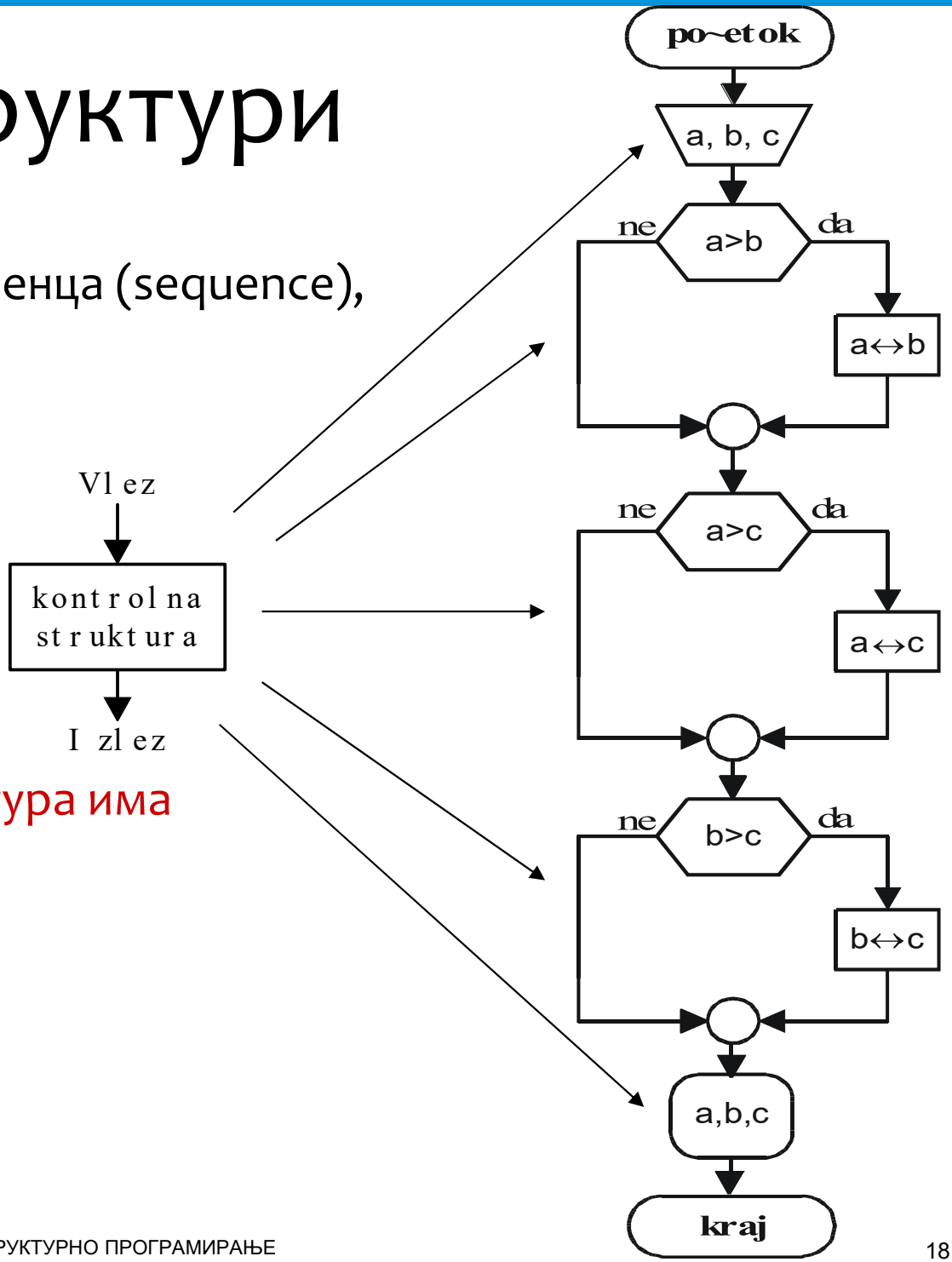
- Секвенцијално (последователно) извршување
  - Чекорите (инструкциите, наредбите) се извршуваат една по друга во испишаниот редослед
- Пренос на контрола
  - Кога следната наредба што се извршува НЕ е следна во редоследот
- Bohm и Jacopini (1966)
  - Математички докажано дека сите програми може да се напишат со помош на 3 контролни структури
    - Редоследна структура: програмите по дефиниција се извршуваат секвенцијално
    - Изборна структура: ако-тогаш (if), ако-тогаш-инаку (if/else), и случај (switch)
    - Структури за повторување (циклуси): while, do/while и for

Контролни структури

1.Редоследна структура или секвенца (sequence),

2. Структура избор
селекција (selection),

3. Структура повторување
итерација (iteration)



Секоја контролна структура има
ЕДНА влезна точка и
ЕДНА излезна точка

Се нарекуваат и:
- линиска структура
- разгранета структура
- циклична структура

За контролните структури

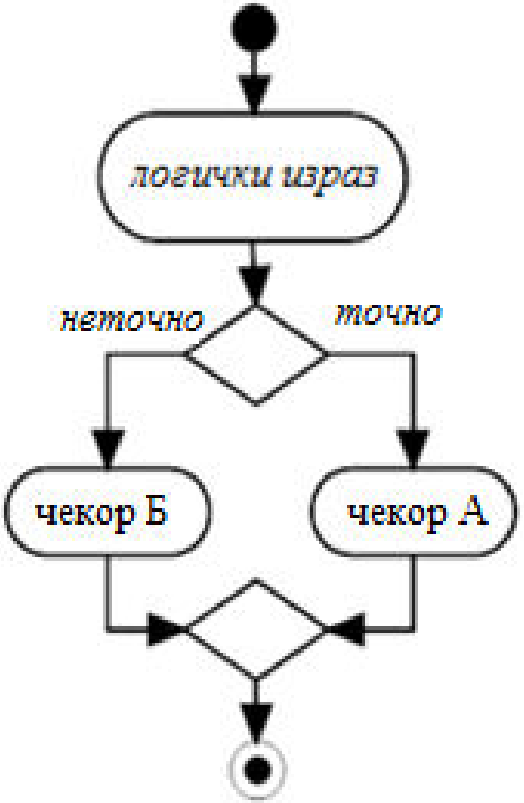
- Единствен влез/единствен излез кај контролните структури
  - Излезната точка од една контролна структура е поврзана со влезната точка од следната контролна структура
  - Програмите лесно се градат на овој начин

Структури за избор

- Структура за избор, разгранета структура, селекција
- Овозможува да се избере извршување на една наредба (или блок-наредби) од една, две или повеќе наредби, во зависност од некој услов:
  - **ако-тогаш** (if),
  - **ако-тогаш-инаку** (if/else), и
  - **случај** (switch)

Структура за избор од две можности

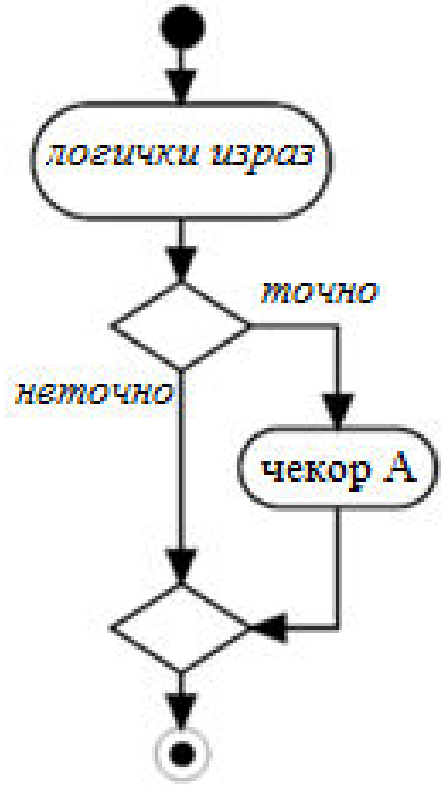
- Со контролната структура за избор од две можности се врши избор на едната од двете можности на продолжување на извршувањето, во зависност од вредноста на некој логички израз
- Структурата се нарекува **ако-тогаш-инаку**



ако *логички израз*
тогаш
чекор А;
инаку
чекор Б;
крај_ако {*логички израз*}

Структура за избор од една можност

- Контролната структура за избор од две можности може да се користи и кога една од можностите не содржи извршни чекори – структура за избор од една можност
- Оваа структура се нарекува **ако-тогаш**



ако *логички израз*
тогаш
чекор А;
крај_ако {*логички израз*}

if - else

Општиот облик на if наредбата е следниот:

```
if (uslov)
    naredba_za_vistinit_uslov;
else
    naredba_za_nevistinit_uslov;
```

Ако има блокови наредби тогаш се означува почетокот и крајот на блокот:

```
if (uslov)
    {
        blok_naredbi_za_vistinit_uslov;
    }
else
    {
        blok_naredbi_za_nevistinit_uslov;
    }
```

if

Делот else не мора да постои.

```
if (uslov)
    naredba_za_vistinit_uslov;
```

и

```
if (uslov)
    {
        blok_naredbi_za_vistinit_uslov;
    }
```

- Условот во заградата може да биде каков било аритметичко-логички израз. Притоа, ако тој има вредност различна од 0, се третира како true, а инаку како false.

Пример: Употреба на if

```
#include <iostream>
using namespace std;
int main() {
   int i;
   cout << "Vnesete cel broj ";
   cin >> i;
   if (i > 0)
      cout << "Vnesen e pozitiven broj." << endl;
   if (i < 0)
      cout << "Vnesen e negativen broj." << endl;
   if (i == 0)
      cout << "Vnesena e nula." << endl;
   return 0;
}
```

Пример: if со сложен услов

- Пример:

```
#include <iostream>
using namespace std;
int main()
{
	char c;
	cout << "Vnesi bukva: ";
	cin >> c;
	if( c == 'a' || c == 'e' || c == 'i' ||
		c == 'o' || c == 'u' )
		cout << "Vnesena e samoglaska" ;
	cout << endl;
	return 0;
}
```

Пример: if-else со сложен услов

- Пример:

```
#include <iostream>
using namespace std;
int main()
{
	char c;
	cout << "Vnesi bukva: ";
	cin >> c;
	if( c == 'a' || c == 'e' || c == 'i' ||
		c == 'o' || c == 'u' )
		cout << "Vnesena e samoglaska" << endl;
	else
		cout << "Vnesena e soglaska" << endl;
	return 0;
}
```

Пример: Дали даден агол е остар

```
#include <iostream>
using namespace std;

int main() {
   int alfa;
   cout << "Vnesete agol vo stepeni: ";
   cin >> alfa;
   if ((alfa > 0) && (alfa < 90))
         cout << "Agolot e ostar " << endl;
   else
          cout << "Agolot ne e ostar " << endl;
   return 0;
}
```

Вгнездување на if-наредби

```
if (sredstva > cena)
      kupi;
   else
      if (imas_prijatel)
            pozajmi_pari;
         else
            najdi_rabota;
```

Вгнездување на наредбите за избор

- Наредбите за избор if и if-else може да се вгнездуваат една во друга
- Треба да се води сметка дека C++ преведувачот секое else го поврзува со претходното if
- На пример, во следниот програмски сегмент, во случај ако не е исполнет условот a > b нема да се отпечати ништо, бидејќи else е врзан со второто if:

```
if (a > b)
     if (a > c)
      cout << "Najgolem broj e " << a;
     else
      cout << "Najgolem broj ne e " << a;
```

Ако-или-ако-инаку

- Ако вгнездувањето се врши во else, се добива контролната структура **ако-или-ако-инаку**
- Пример:

```
if (услов1)
        наредба 1;
else if (услов2)
        наредба 2;
    else if  (услов3)
        наредба 3;
    else
        наредба 4;
```

Пример: Употреба на if-else

```
#include <iostream>
using namespace std;
int main() {
   int i;
   cout << "Vnesete cel broj ";
   cin >> i;
   if (i > 0)
        cout << "Vnesen e pozitiven broj." << endl;
     else
       if (i < 0)
            cout << "Vnesen e negativen broj.";
          else
            cout << "Vnesena e nula." << endl;
   return 0;
}
```

Употреба на if-else – Поинаков запис

```
#include <iostream>
using namespace std;
int main() {
   int i;
   cout << "Vnesete cel broj ";
   cin >> i;
   if (i > 0)
      cout << "Vnesen e pozitiven broj." << endl;
   else if (i < 0)
       cout << "Vnesen e negativen broj." << endl;
   else
       cout << "Vnesena e nula." << endl;
   return 0;
}
```

Вгнездување на if-структура

Што ќе се изврши за x<5? А што за x=11?

```
if (x > 5)
    if(x < 11)
        polozi;
else
    padna;
```

Како треба да гласи структурата за да биде логична?

Вгнездување на if-структура

Што ќе се изврши за x<5? А што за x=11?

```
if (x > 5)
    if(x < 11)
       polozi;
else
    padna;
```

Како треба да гласи структурата за да биде логична?

```
if (x > 5) {
      if(x < 11) polozi;
      }
else   padna;
```

Пример: Најголем од три броја

```
#include <iostream>
using namespace std;
int main() {
   int a, b, c;
   cout << "Vnesete tri razlichni broja: ";
   cin >> a >> b >> c;
   if (a > b)
         if (a > c)
               cout << a << endl;
         else
               cout << c << endl;
  else
         if (b > c)
               cout << b << endl;
         else
               cout << c << endl;
   return 0;
}
```

Пишување на условот

Често се пишува

```
if(izraz)
```

наместо

```
if(izraz != 0)
```

или

`if(!izraz)` наместо `if(izraz == 0)`

Во изразите може да се најдат и наредби за доделување и/или инкрементирање/декрементирање

```
if(x = y + 1  && ++i<n)  ...
```

Условен оператор ?:

- Условниот оператор се користи за претставување на едноставни if-else наредби
- Синтакса:

услов? **израз Т** : **израз Н;**

- Прво се пресметува *услов*, чија вредност може да биде true или false. Ако е true, тогаш се пресметува **израз Т**, а ако е false се пресметува **израз Н**
- На пример, програмскиот сегмент

```
if (m > n) s = m - n;
else s = n - m;
```

може да се запише вака:

```
s = (m > n) ? (m - n) : (n - m);
```

Пример: Употреба на ?:

- Со следнава наредба ќе се отпечати „paren“ ако променливата broj има парна вредност, во спротивно ќе се отпечати „neparen“:

```
cout << "Brojot " << broj << "  e " << ((broj % 2)  ? "neparen." : "paren.");
```

- На пример, за broj = 5 ќе се отпечати:

```
Brojot 5 e neparen.
```

Да се реши!

- На влез се внесува еден број – ден од месец октомври.
- На излез да се испише кој е датумот на утрешниот ден.
- Влез: 20
- Излез: 21. oktomvri

Прашања?