Нов Български Университет

GENB006 Увод в алгоритмите

Занятие 2

Филип Андонов, fandonov@nbu.bg

департамент Информатика

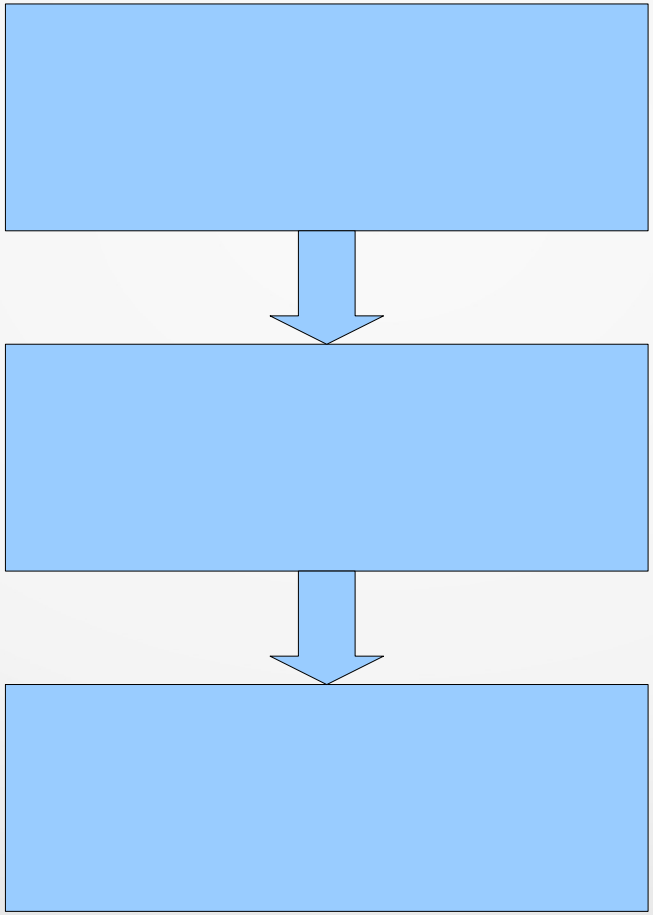
Структури за управление

Структурите за управление са средство за изграждане на алгоритми и съществуват във всички алгоритмични езици с някои различия в синтаксиса. Основните структури са:

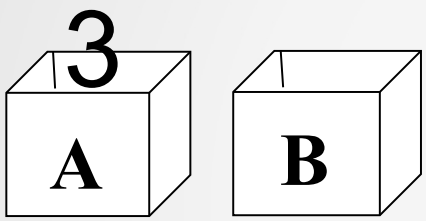
- *избор на алтернатива (двуклон);*
- *цикъл с предусловие;*
- *цикъл със след-условие;*
- *избор на вариант (многоклон)*.

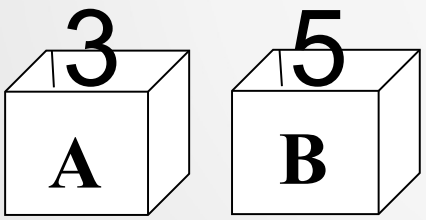
Структури за управление

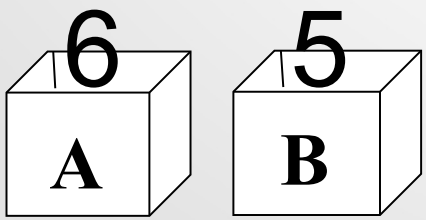
Най-елементарният алгоритъм е линейната структура, чиято блок-схема изглежда така:



Структури за управление







A=3

B=5

A=B+1

Безусловен преход

Извърши действие X (промяна на стойности в средата). *(След това)*

Извърши действие Y (промяна на стойности в средата). *(След това)*

Извърши действие Z (промяна на стойности в средата).

Структури за управление

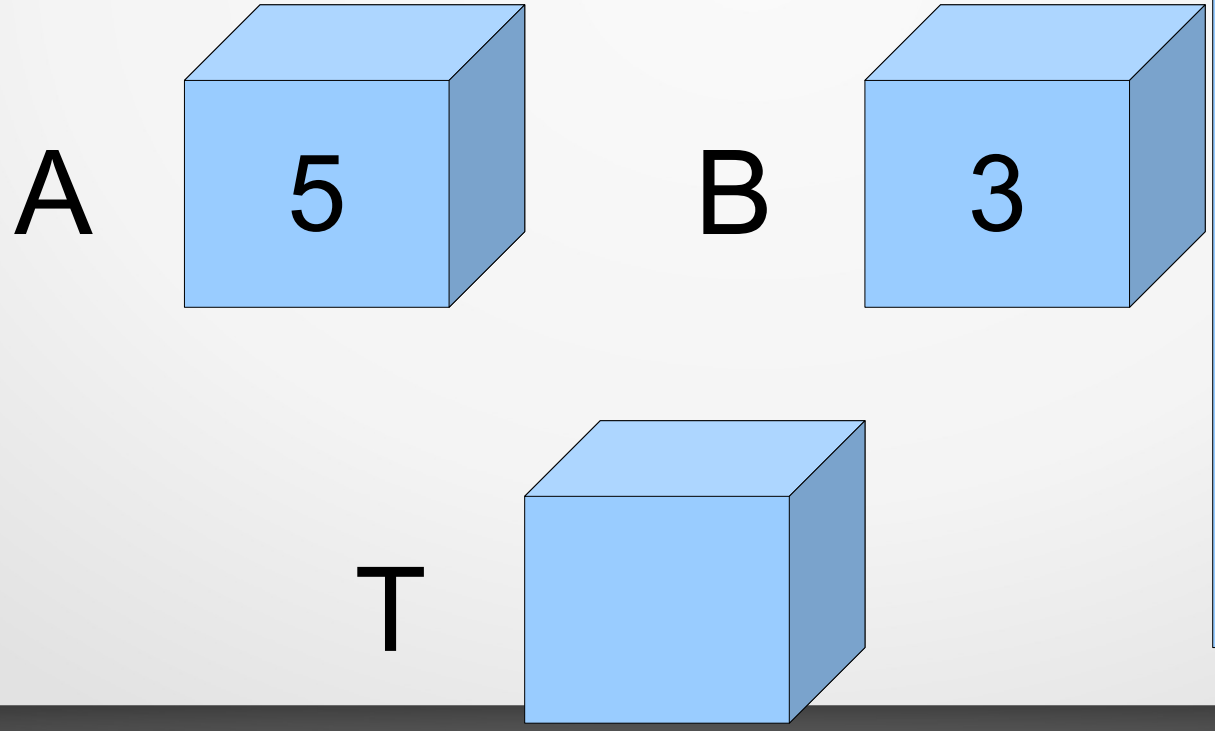
Такива програми на практика се изпълняват по един и същ начин всеки път и този начин е известен по време на създаването им (design time). Това поведение на програмите се нарича безусловен преход.

За да могат програмите да се адаптират към средата по време на изпълнение (runtime), те трябва да имат средства за условно разклонение.

Структури за управление

Пример за често използван блок

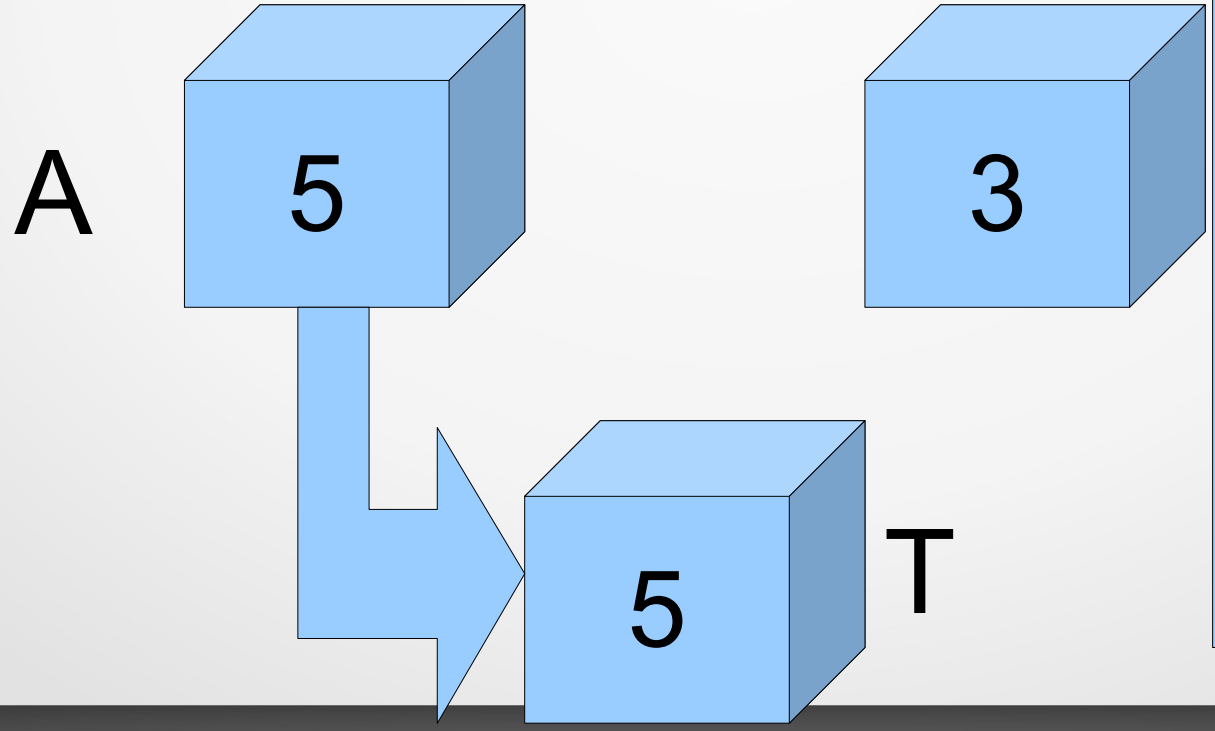
Блок за размяна на стойностите на две променливи



Структури за управление

Пример за често използван блок

Блок за размяна на стойностите на две променливи



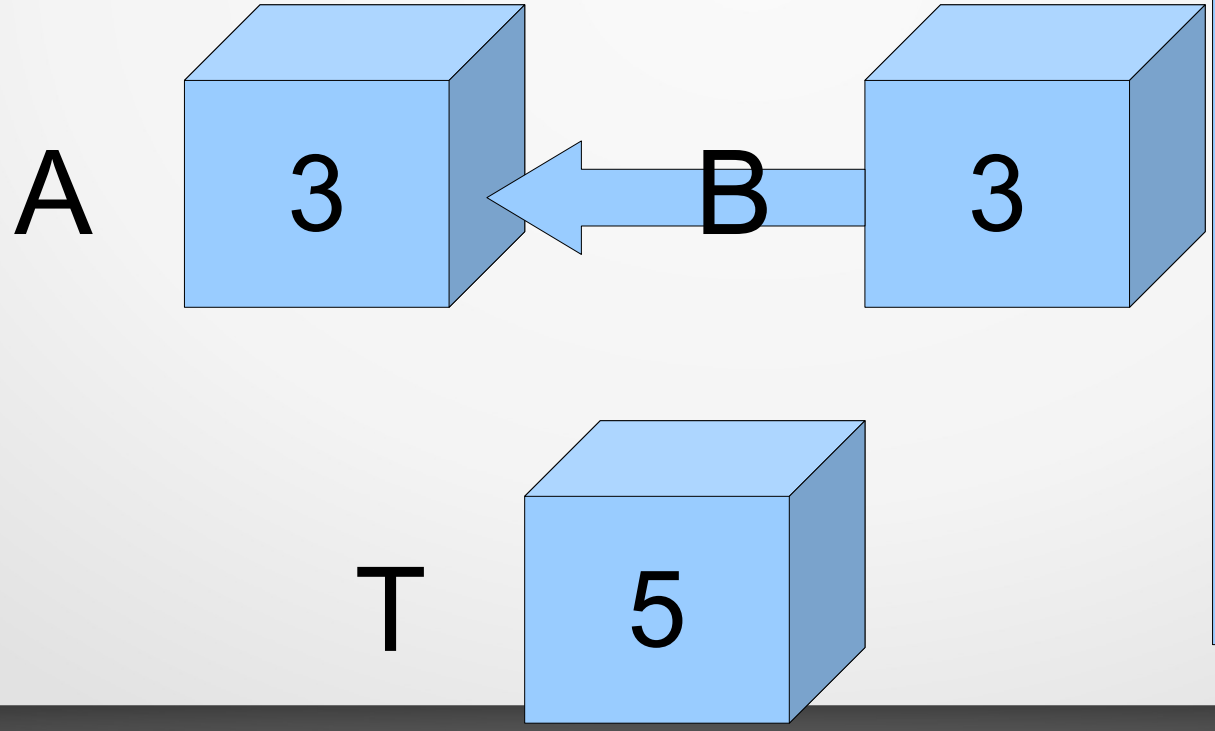
Размяна на стойностите
На две променливи

T=A

Структури за управление

Пример за често използван блок

Блок за размяна на стойностите на две променливи



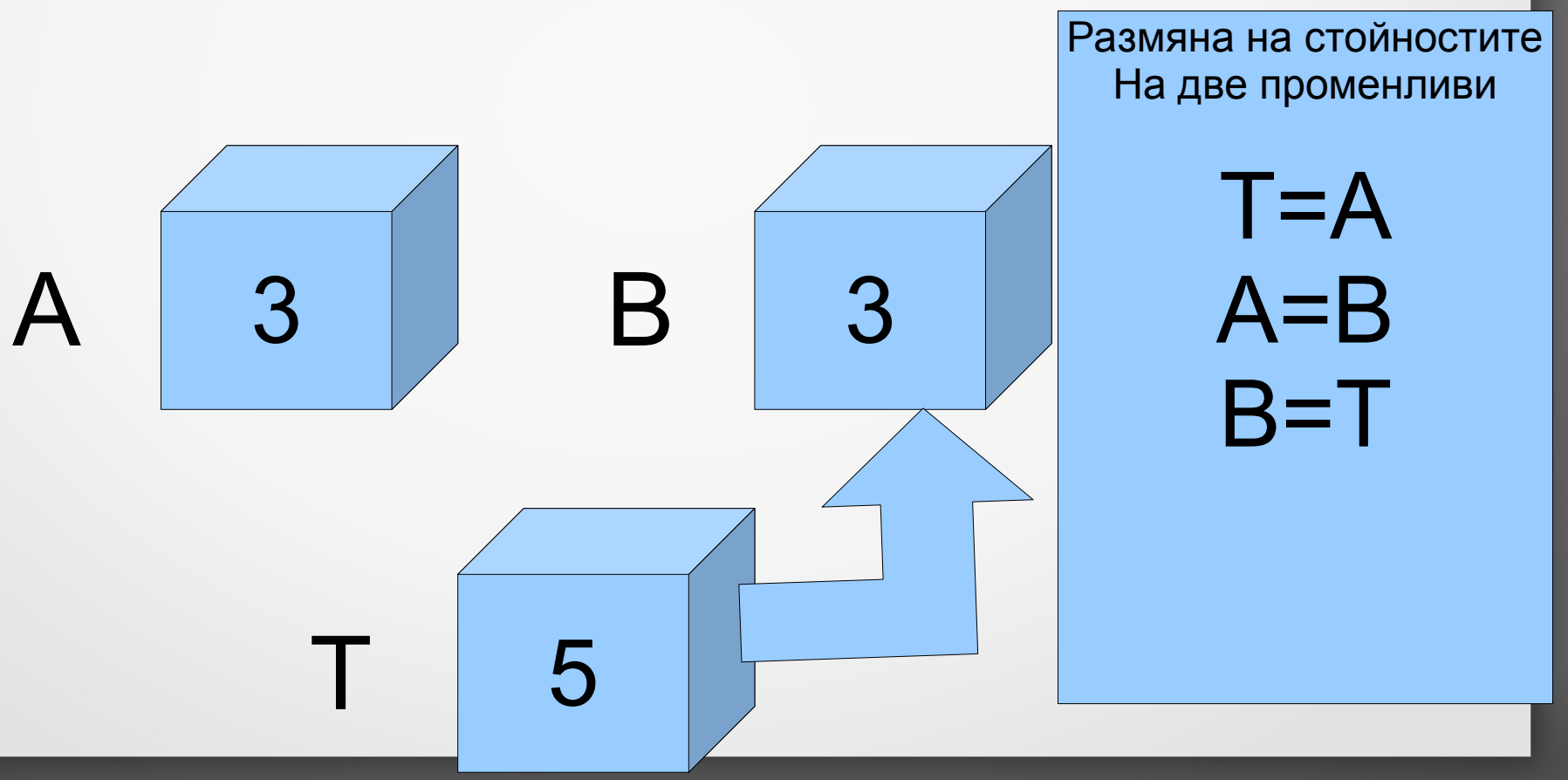
Размяна на стойностите
На две променливи

T=A
A=B

Структури за управление

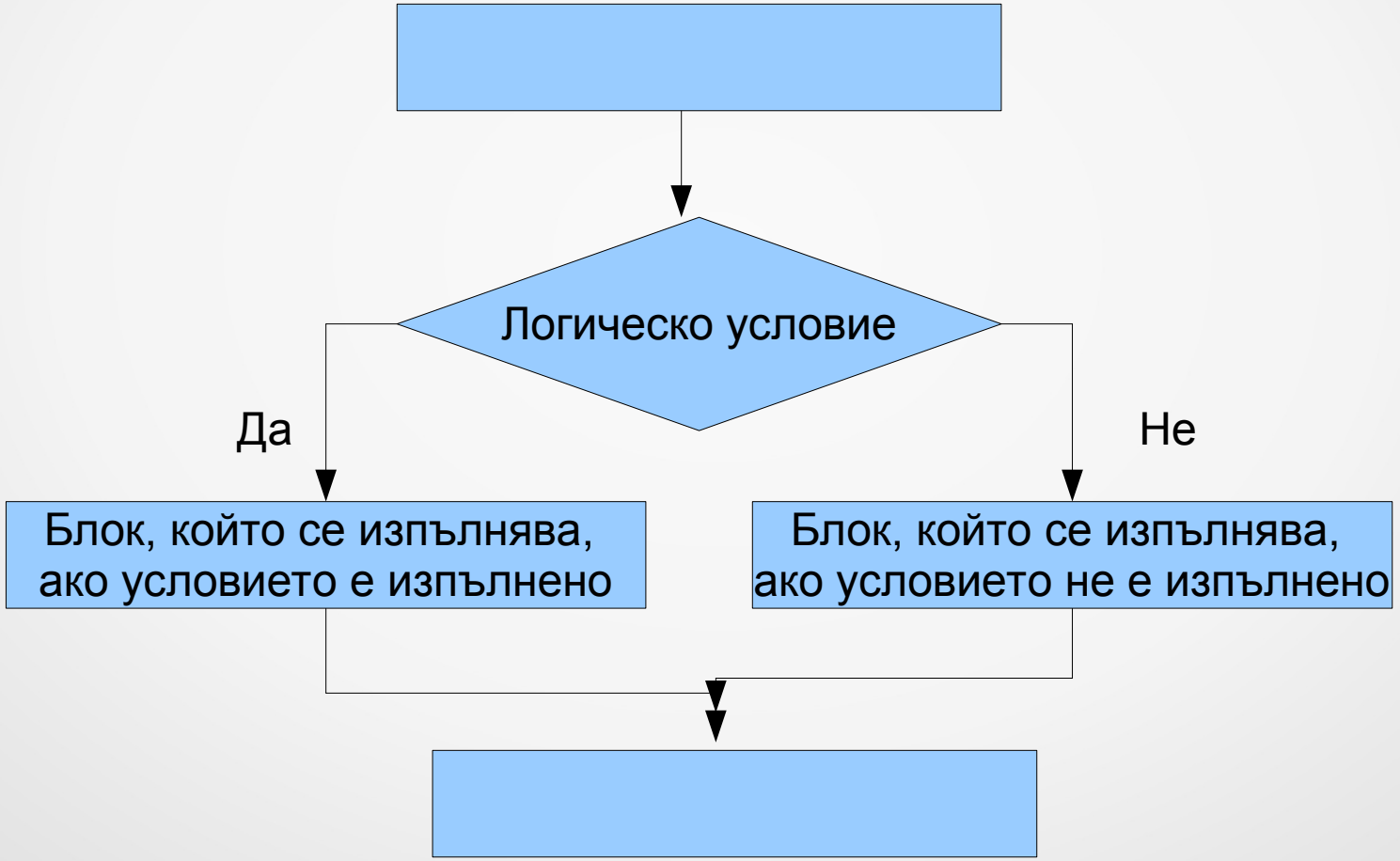
Пример за често използван блок

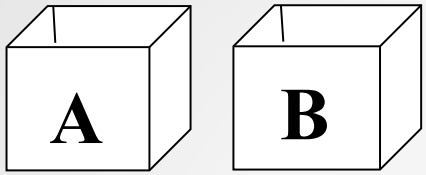
Блок за размяна на стойностите на две променливи



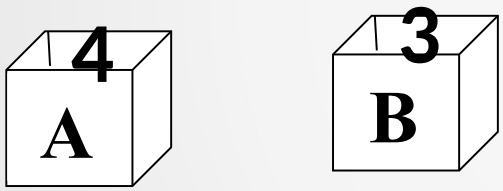
Избор на алтернатива

- Условен преход

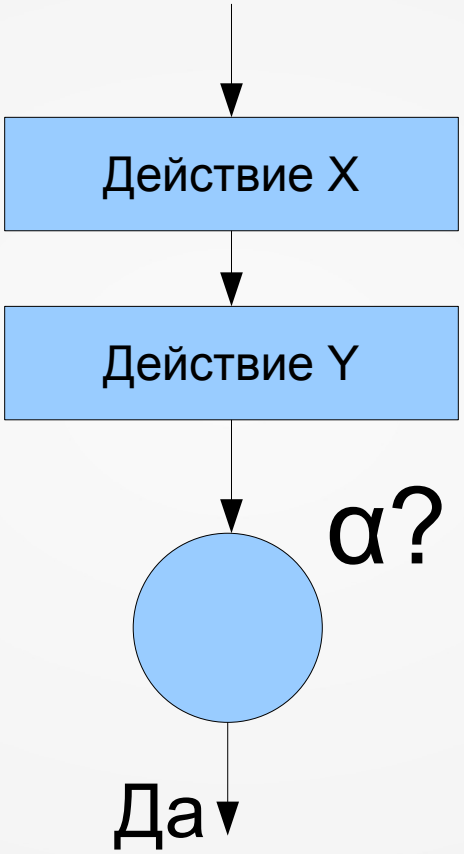




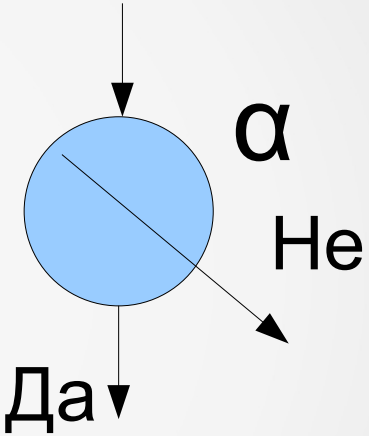
Среда

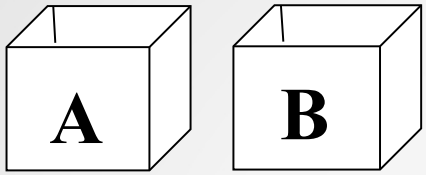


A > B ?

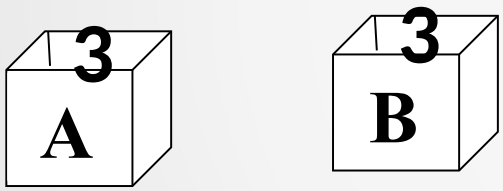


Базови елементи на управление

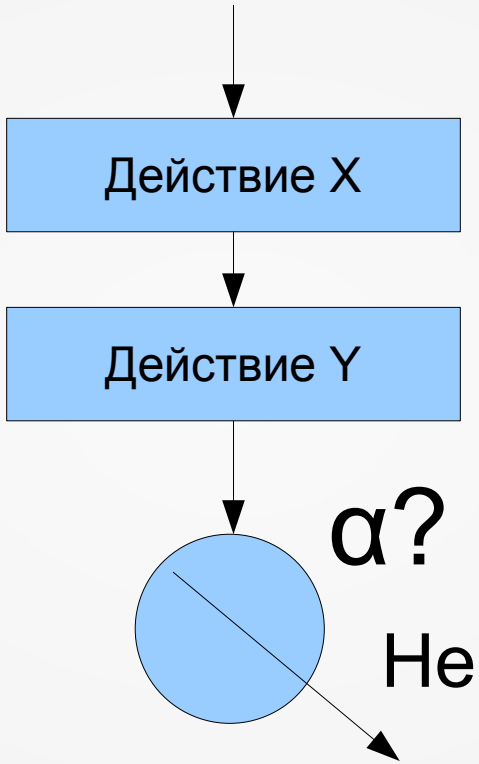




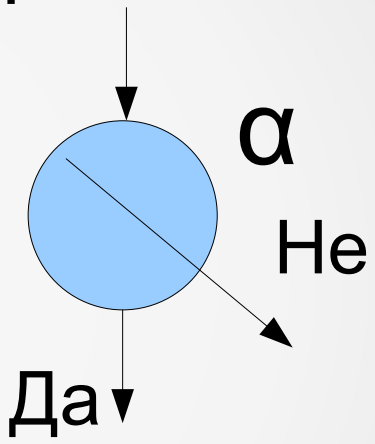
Среда

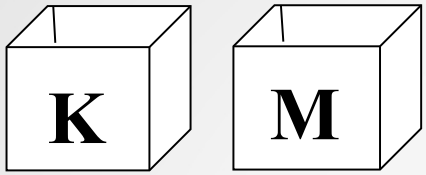


A > B ?

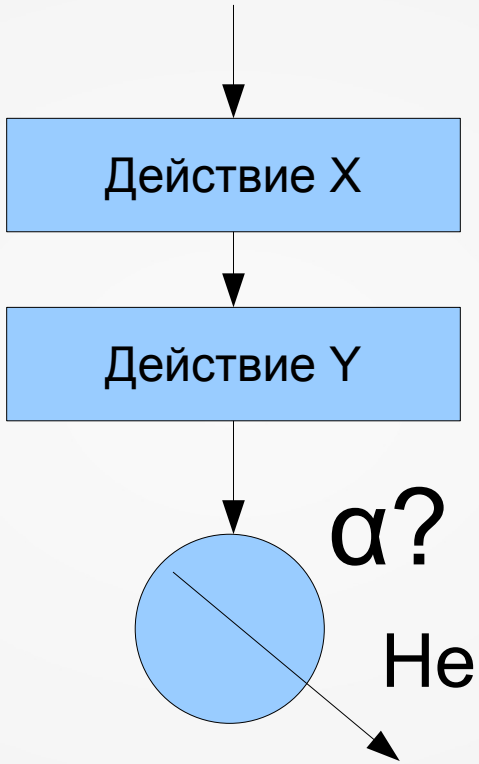


Базови елементи на управление

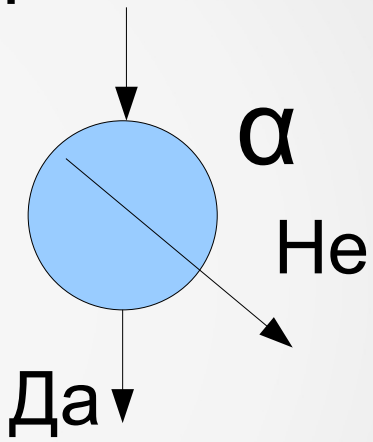




Среда



Базови елементи на управление



Най-елементарните проверки α се отнасят до сравнение на две стойности и имат смисъл на въпроси от вида на:

“Е ли съдържанието на променливата K по-голямо от това на променливата M?”

Начин на изпълнение

Изчислява се стойността на логическия израз, при резултат “да” (true) се изпълнява блокът A, при резултат “не” (false) се изпълнява блокът B. И в двата случая след това се изпълнява блокът, сочен от изходящата стрелка. Важното е, че двата изпълними блока се изпълняват *алтернативно*: или единият, или другият.

Представянето на тази структура според синтаксиса на C е следният :

```
if(<израз>){
// оператори, изпълнявани когато резултатът е true
}else{
// оператори,изпълнявани при стойност false на израза
}
```

Пример

Да се състави условен оператор, чрез който на променливата `a` се присвоява по-малката измежду стойностите на променливите `x` и `y`.

Решение:

```
if(x<y){
   a=x;
}else{
   a=y;
}
```

Операции за сравнение

Операциите за сравнение имат по-малък приоритет от коя да е аритметична операция.

Операции за сравнение	
знак	**смисъл**
>	По-голямо
>=	По-голямо или равно
<	По-малко
<=	По-малко или равно
==	Равно
!=	Не равно

Логически променливи и изрази

В C++ съществува отделен тип за представяне на величини със стойност “вярно” или “не вярно”. Тези стойности се представят чрез литералите true и false (които същевременно са и служебни думи), а съответният тип се означава като bool (*булев*, в чест на Дж.Бул или още *логически*).

Резултатът от изрази със сравнения и логически изрази, независимо колко сложни са те, винаги е от булев тип.

Логически операции		
приоритет	**знак**	**означава**
Най-висок	!	Отрицание
	&&	Логическо “и”
Най-нисък	||	Логическо “или“

Логически изрази

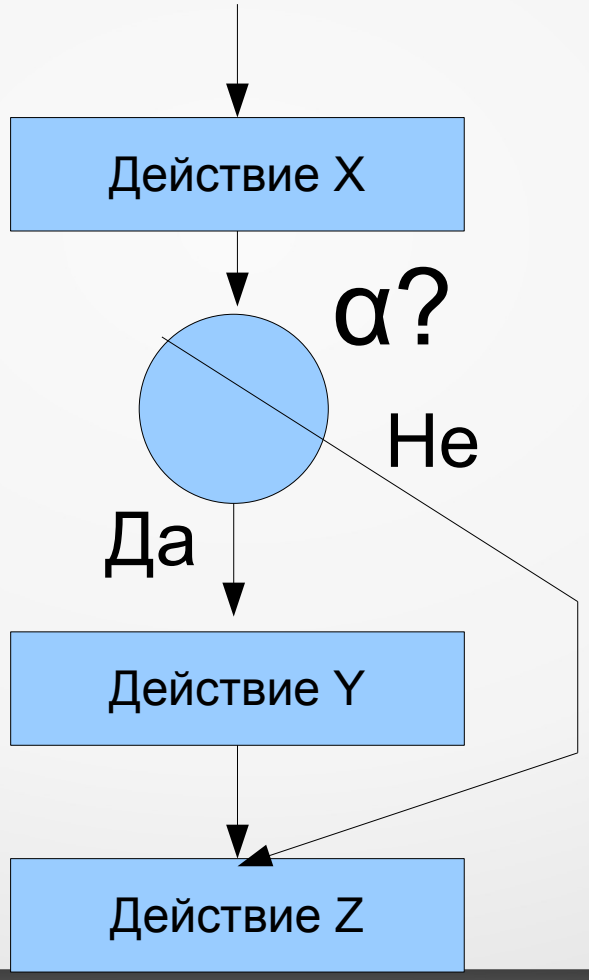
p	**q**	**!p**	**p&&q**	**p||q**
false	false	*true*	*false*	*false*
false	true	*true*	*false*	*true*
true	true	*false*	*true*	*true*
true	false	*false*	*false*	*true*

Често срещана грешка

Две особености на C са причина за трудно откриваеми грешки при невнимателно изписване на сравнение за равенство. Ако отляво на знака “==” стои променлива и знакът се изпише погрешно като “=”, сравнението се превръща в оператор за присвояване.

Байпас

В процедурните езици, конструктите за управление могат да бъдат изказани с готови езикови фрази.

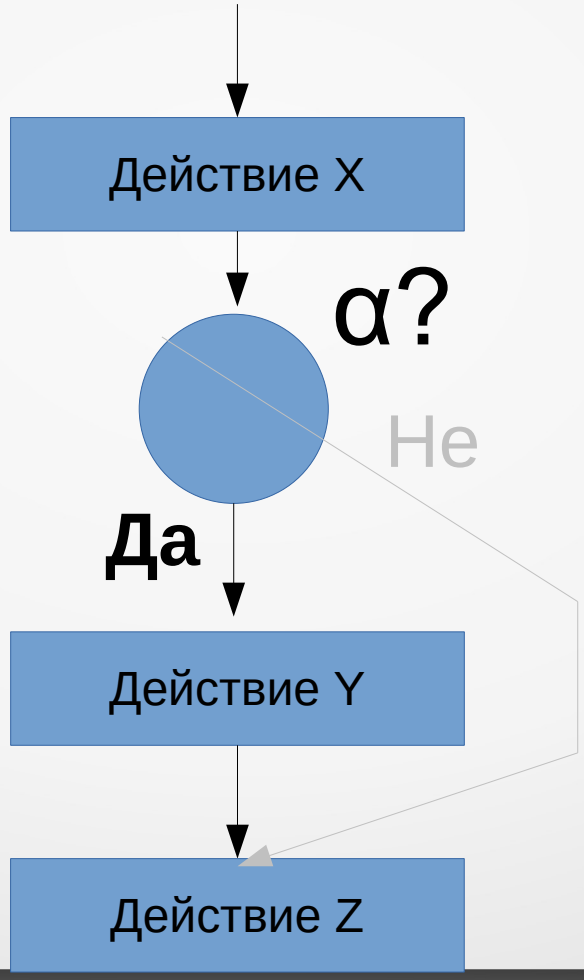



```
....;

if (α)
{
    .........;
    .........;
}
..........;
```

Байпас

В процедурните езици, конструктите за управление могат да бъдат изказани с готови езикови фрази.




```
….;

if (α)
{
    ……….;
    ……….;
}
……….;
```

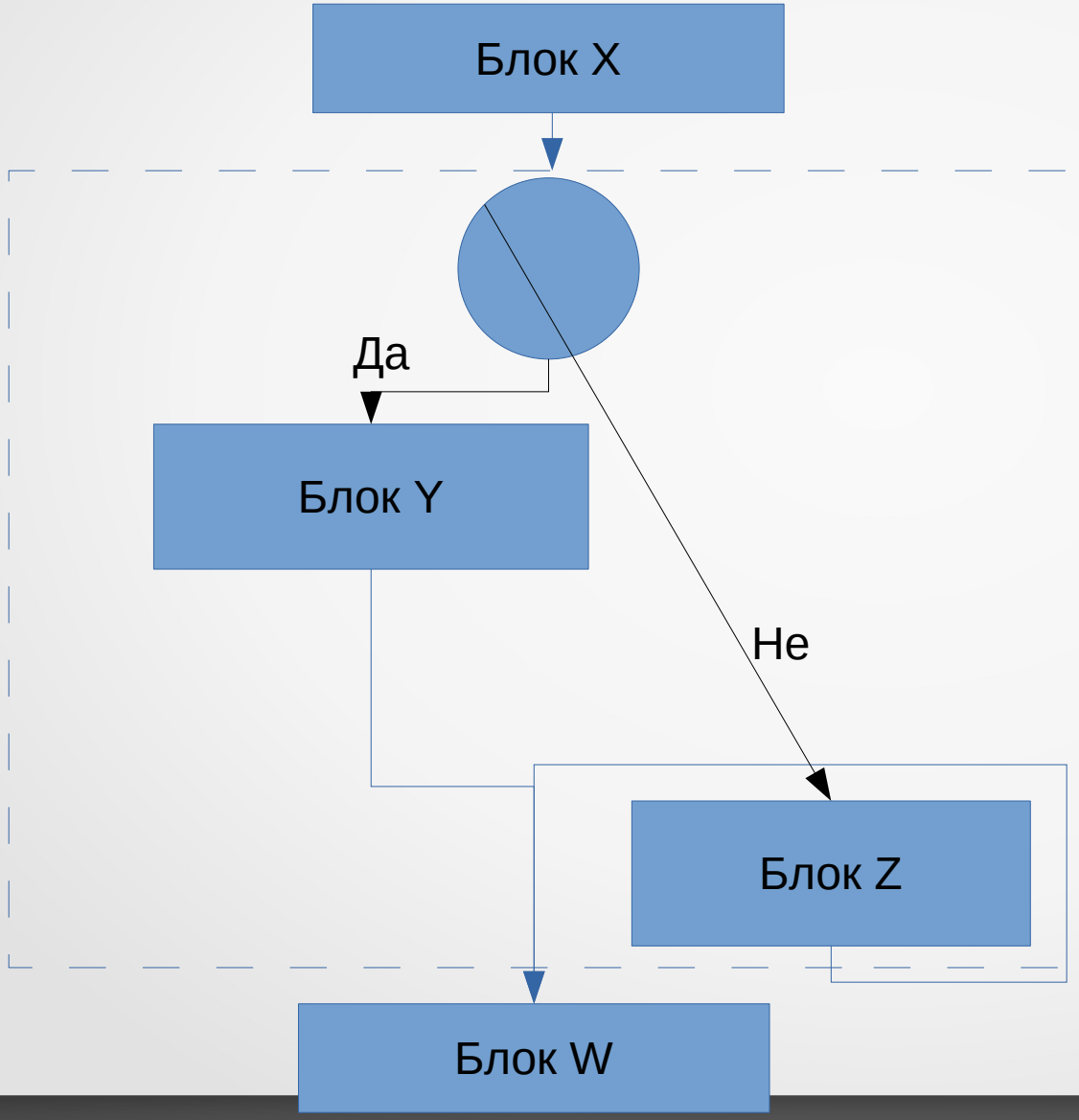
Байпас

В процедурните езици, конструктите за управление могат да бъдат изказани с готови езикови фрази.


```
....;

if (α)
{
    .........;
    .........;
}
..........;
```

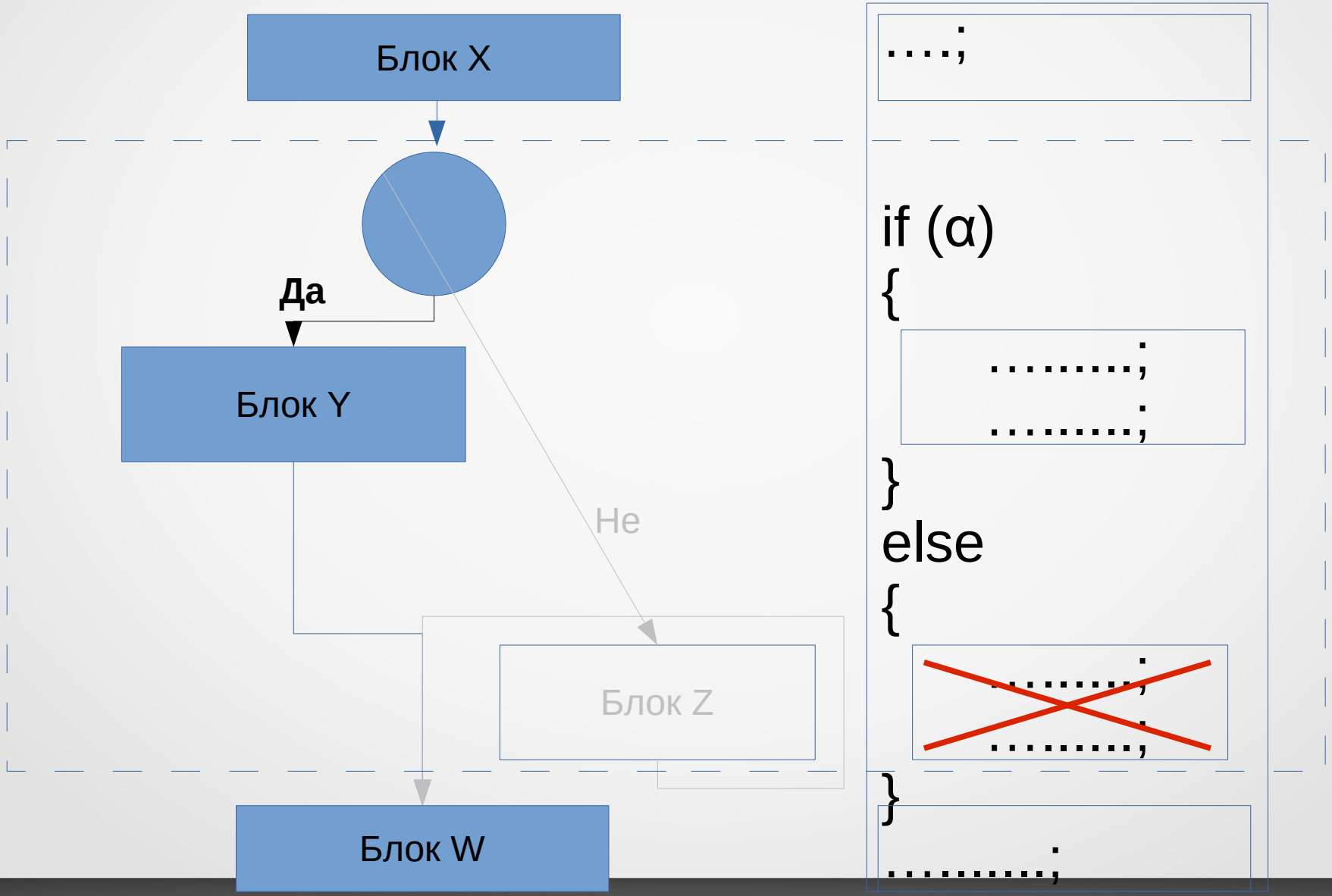
Двуклон



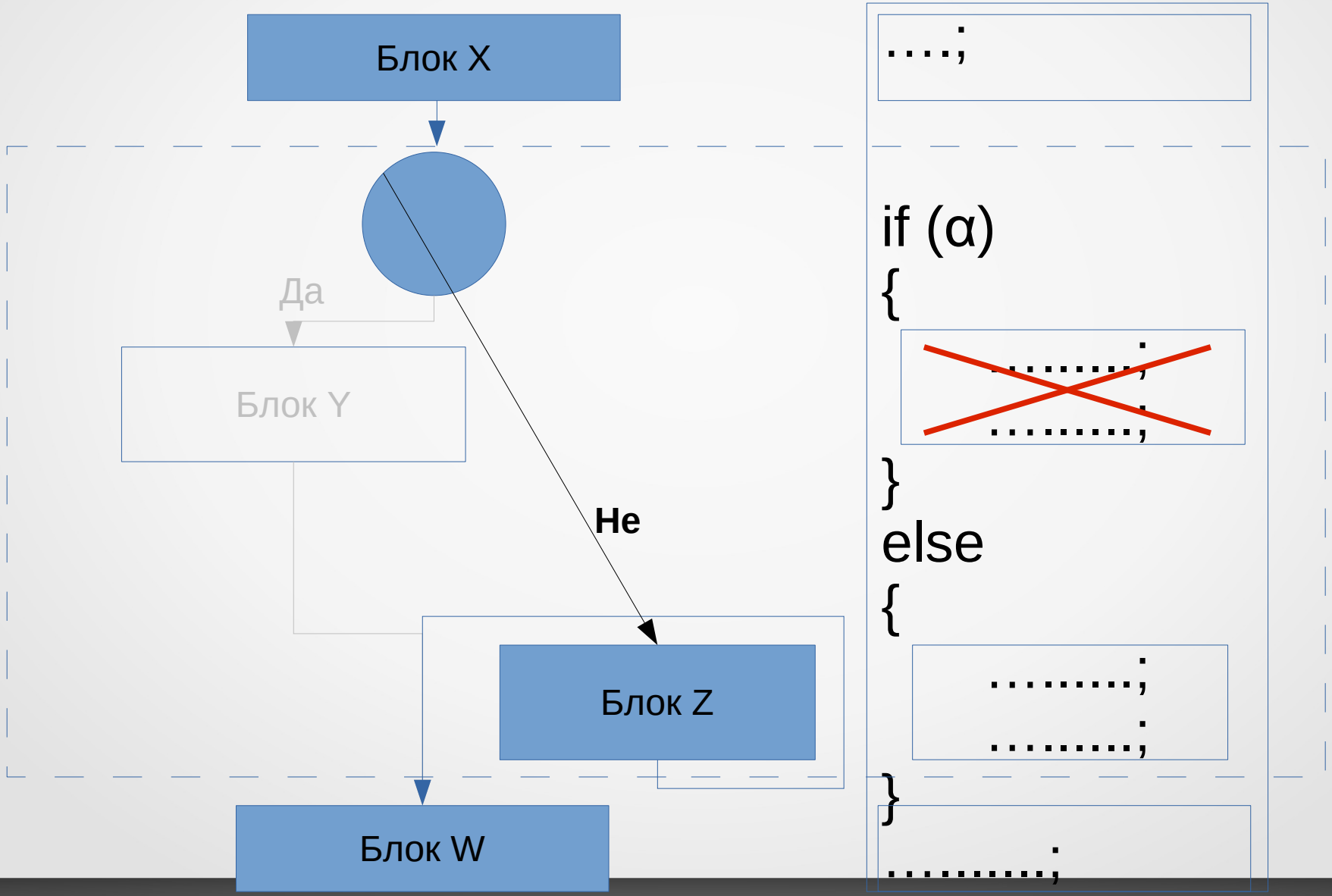

```
    ....;

if (α)
{
    .........;
    .........;
}
else
{
    .........;
    .........;
}
    .
.........;
```

Двуклон



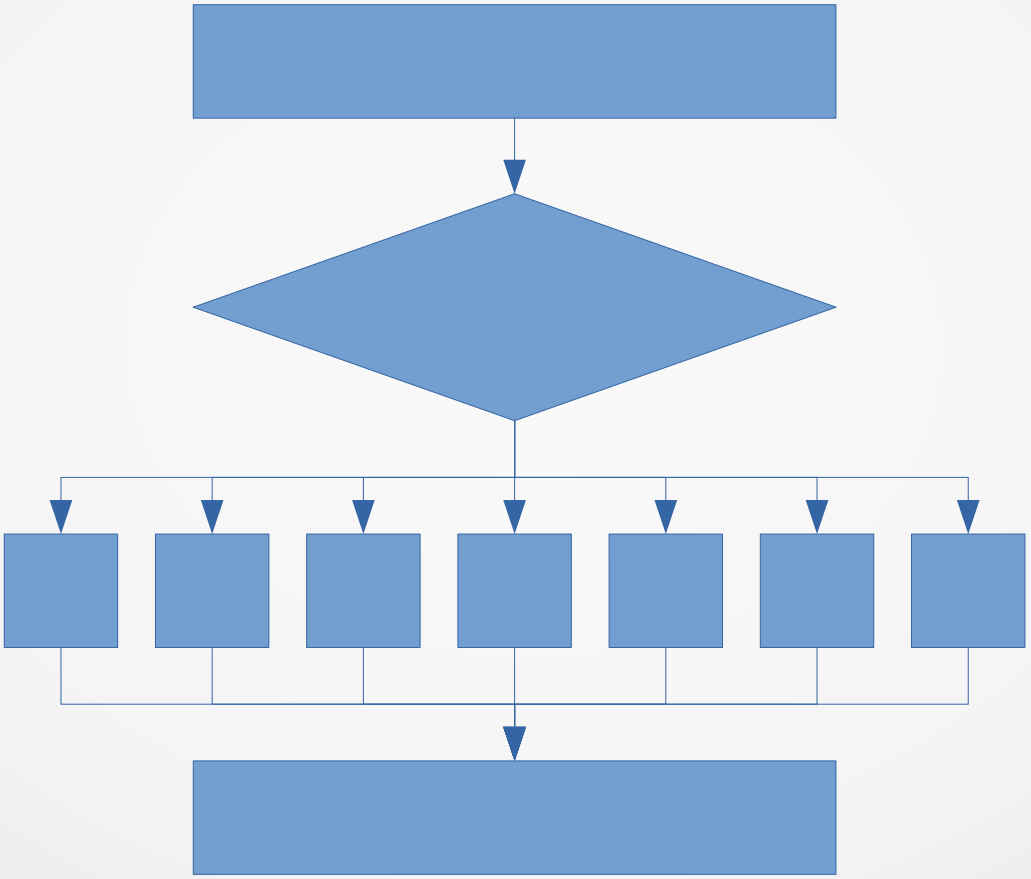
Двуклон



Многоклон

Операторът switch е оператор за условен преход, който позволява разклонение към повече от два клона в програмата. Чрез него се избира една от няколко взаимноизключващи се алтернативни възможности.

Избор на вариант



Избор на вариант

Синтаксисът на оператора е следният:

```
switch(израз_тест)
case етикет1:
//оператори
break;
case етикет2:
//оператори
break;
default:
//оператори
break;
```

Избор на вариант

Изпълнение: Селекторът се изчислява и получената стойност се сравнява последователно с етикетите в реда на следването им в блока на оператора `switch`. При несъвпадение се преминава към следващия `case`. Ако има `default`, тялото му се изпълнява безусловно. При съвпадение между етикет и стойността на селектора се изпълнява тялото след етикета. Ако това тяло е ограничено от `break`, всичко останало в блока на `switch` се пропуска и управлението се предава на първия оператор след съставния оператор `switch`. Ако `break` липсва, преминава се към обработка на следващия `case`.

Избор на вариант

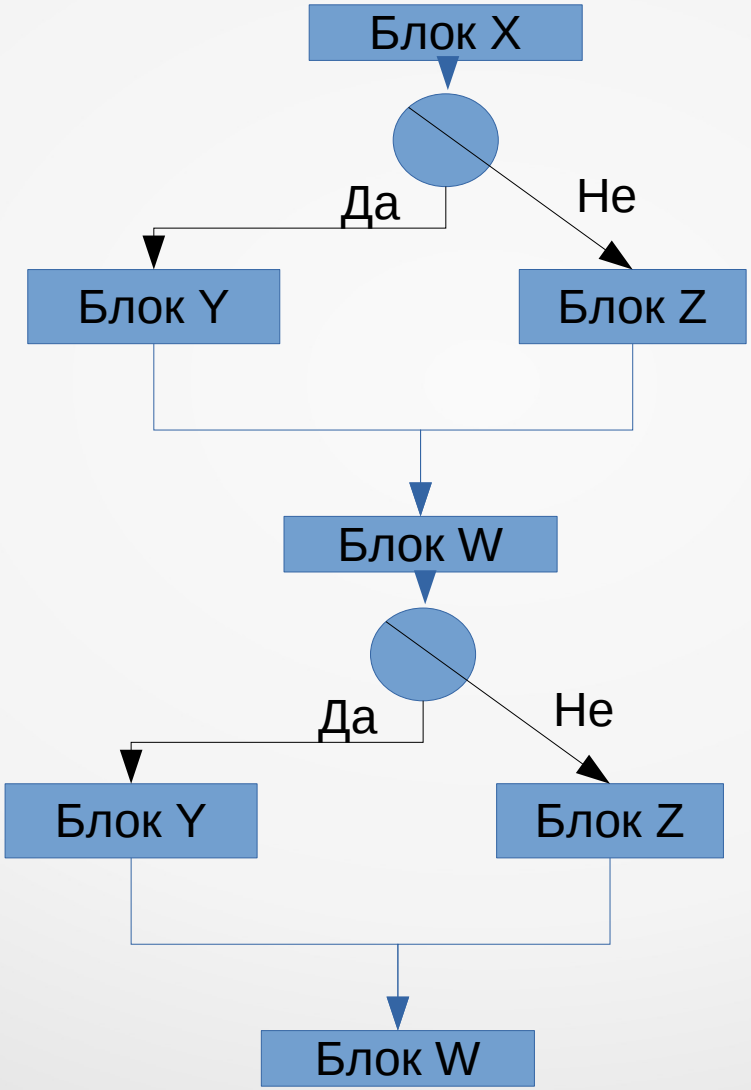
Пример:

```
switch(transport):
case ”p”:
cout<<”He came by plane.”;
break;
case ”t”:
cout<<”He came by train.”;
break;
default:
cout<<”We don't know how he came here.”;
break;
```

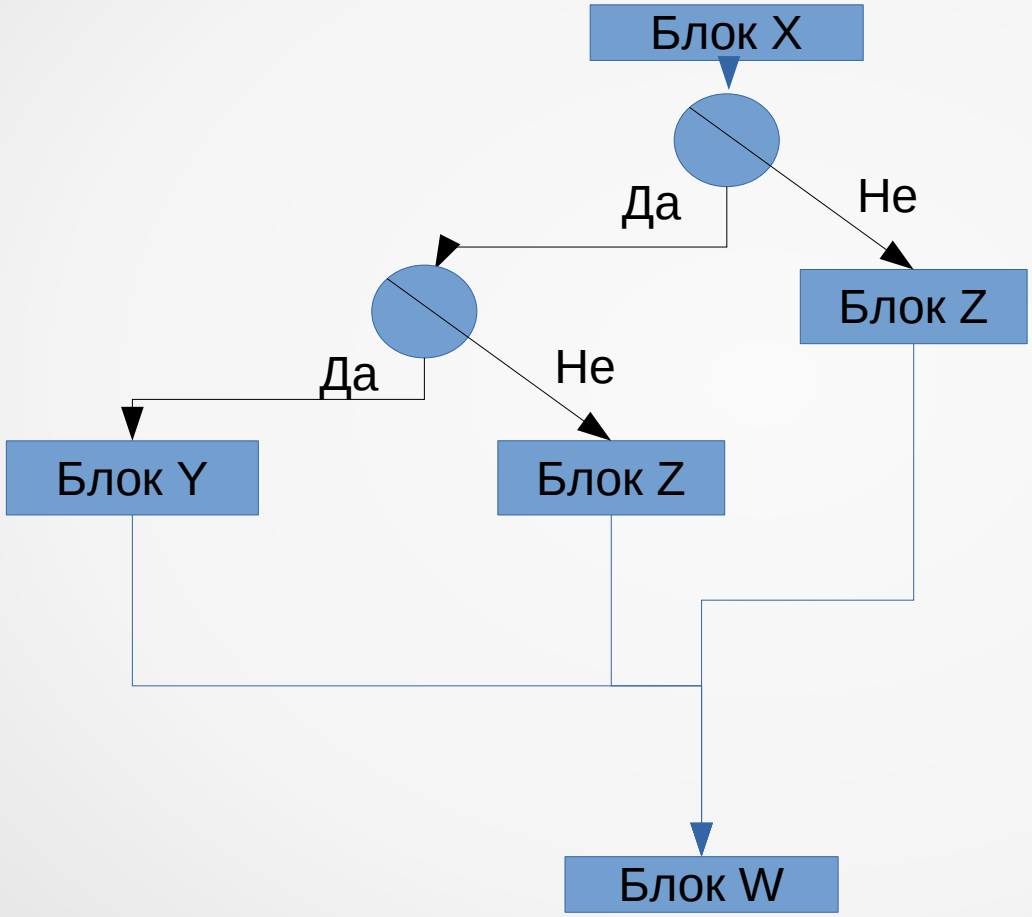
Вложени условни оператори

- Условните оператори могат да се влагат
- Това означава, че вътрешният условен оператор се намира в един от клоновете на външния

Последователни условни оператори



Вложени условни оператори



Пример

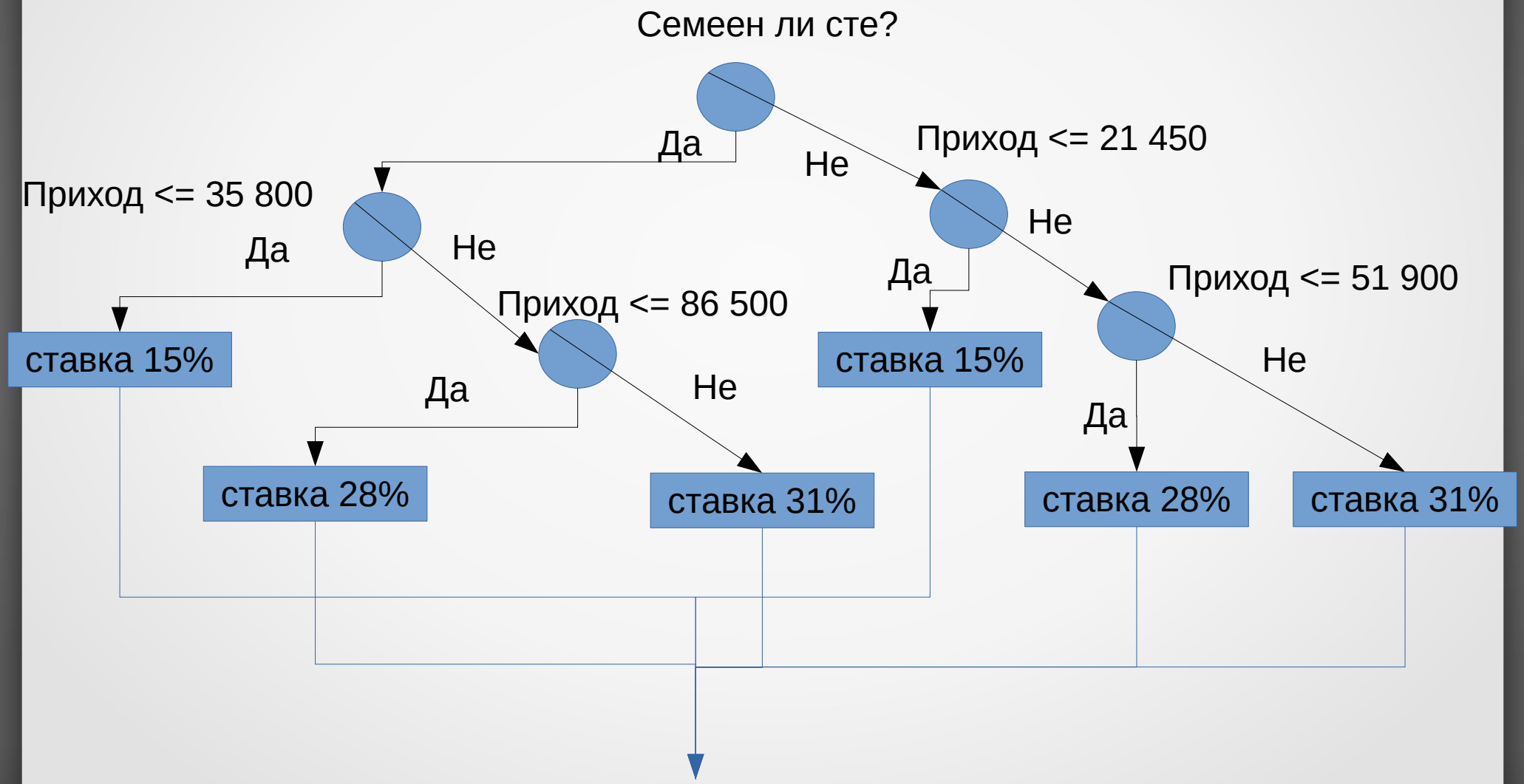
За неженени/неомъжени

Ако облагаемата сума е над	Но не повече от	Данъкът е	Върху сумата над
0	21 450	15%	0
21 450	51 900	3 217.50 + 28%	21 450
51 900		11 743.50 + 31%	51 900

За женени/омъжени

Ако облагаемата сума е над	Но не повече от	Данъкът е	Върху сумата над
0	35 800	15%	0
35 800	86 500	5 370.00 + 28%	35 800
86 500		19 566.00 + 31%	86 500

Пример


Семеен ли сте?
Да
Не
Приход <= 35 800
Да
Не
Приход <= 21 450
Не
Да
Приход <= 86 500
Да
Не
Приход <= 51 900
Не
Да
ставка 15%
ставка 28%
ставка 31%
ставка 15%
ставка 28%
ставка 31%

Често срещана грешка

Таксата за изпращане на малък пакет е 5 лв. за ЕС, 10 за Обединеното Кралство и 20 извън ЕС.

```
double shipping_charge = 5;
if (country == ”EU”)
    if (state == ”UK”)
/*За Обединеното Кралство е по-скъпо*/
        shipping_charge = 10;
else
    shipping_charge = 20;
```

Често срещана грешка

Таксата за изпращане на малък пакет е 5 лв. за ЕС, 10 за Обединеното Кралство и 20 извън ЕС.

```
double shipping_charge = 5;
if (country == ”EU”)
    if (state == ”UK”)
/*За Обединеното Кралство е по-скъпо*/
        shipping_charge = 10;
    else
        shipping_charge = 20;
```

Често срещана грешка

Таксата за изпращане на малък пакет е 5 лв. за ЕС, 10 за Обединеното Кралство и 20 извън ЕС.

```
double shipping_charge = 5;
if (country == ”EU”)
{ if (state == ”UK”)
/*За Обединеното Кралство е по-скъпо*/
        shipping_charge = 10;
}
    else
        shipping_charge = 20;
```

Структури за цикъл

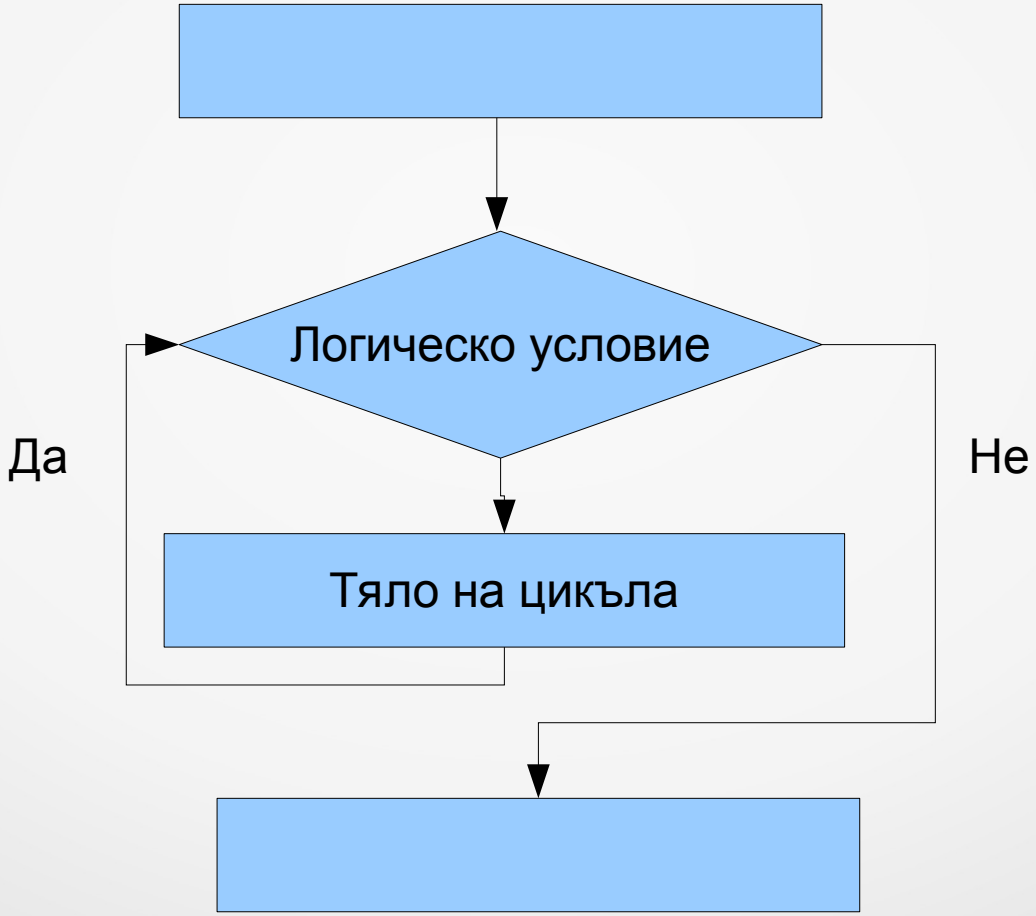
Когато в програмата има последователност от операции, които трябва да се извършат многократно върху различни данни, тя се оформя като като *циклична структура за управление* (цикъл). Цикълът се състои от *тяло и условие*. Тялото на цикъла е блока, чието изпълнение ще се изпълнява многократно. Условието за повторение е логически израз, който определя кога да се напусне цикъла и да се продължи изпълнението на операциите след него.

Структури за цикъл

Съществуват три основни вида циклични структури:

- Цикъл с предусловие
- Цикъл със следусловие
- Цикъл с брояч

Цикъл с предусловие



Цикъл с предусловие

Общ вид:

```
while(<лог. израз>){//изразът е условието за изпълнение на тялото
//оператори от тялото на цикъла
}
```

Цикъл с предусловие

```
int main(){
int a=0;
label1: if (a>10) goto label2;
a++;
goto label1;
label2: cout<<"a="<<a;
return 0;
}
```

Цикъл с предусловие

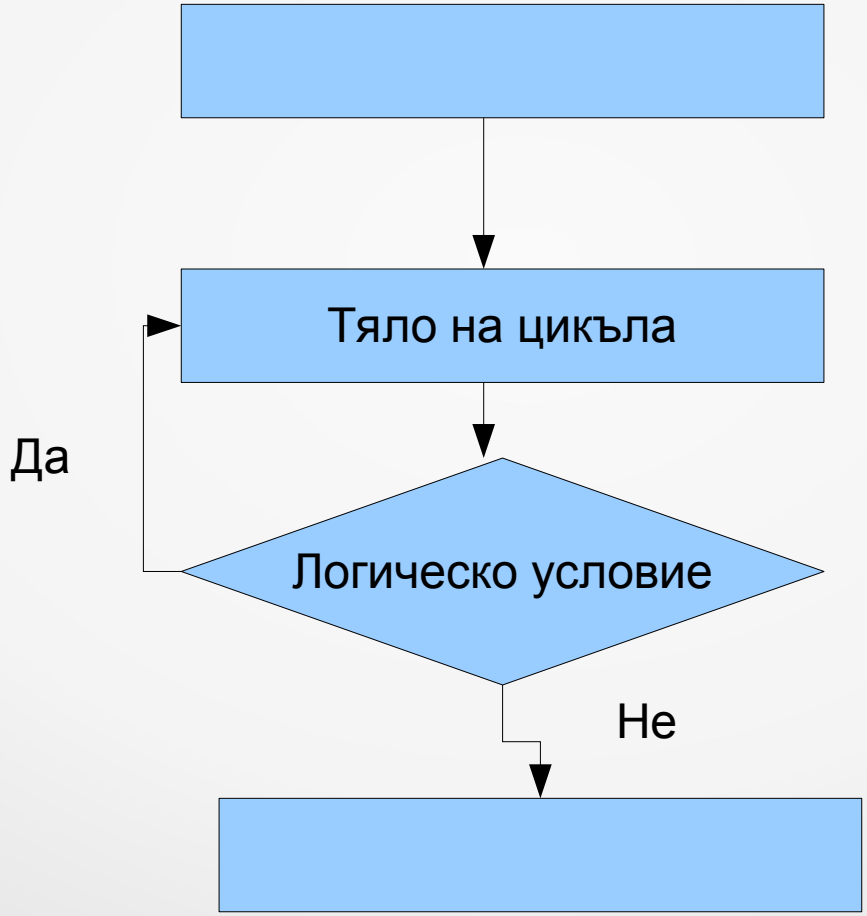
Пример 1:

```
while (i<5){
printf(„%d“,i);
i++;
}
```

Пример 2:

```
while (c!=“s“){
printf(„press s to stop\n“);
c=getchar();
}
```

Цикъл със следусловие



Цикъл със следусловие

Общ вид:

```
do{
//оператори от тялото на цикъла
}while(<лог. израз>);
//изразът представя условието за повторно изпълнение на тялото
```

Цикъл със следусловие

Пример 1:

```
do{
printf(„%d“,i);
i++;
}while (i<5)
```

Пример 2:

```
do{
printf(„press s to stop\n“);
c=getchar();
}while (c!=“s“)
```

Внимание!

Поради това, че при цикъла със следусловие тялото се изпълнява преди да се провери условието, то този цикъл винаги се изпълнява поне веднъж.

За разлика от него, цикълът с предусловие може да не се изпълни нито веднъж, ако условието му не е изпълнено.

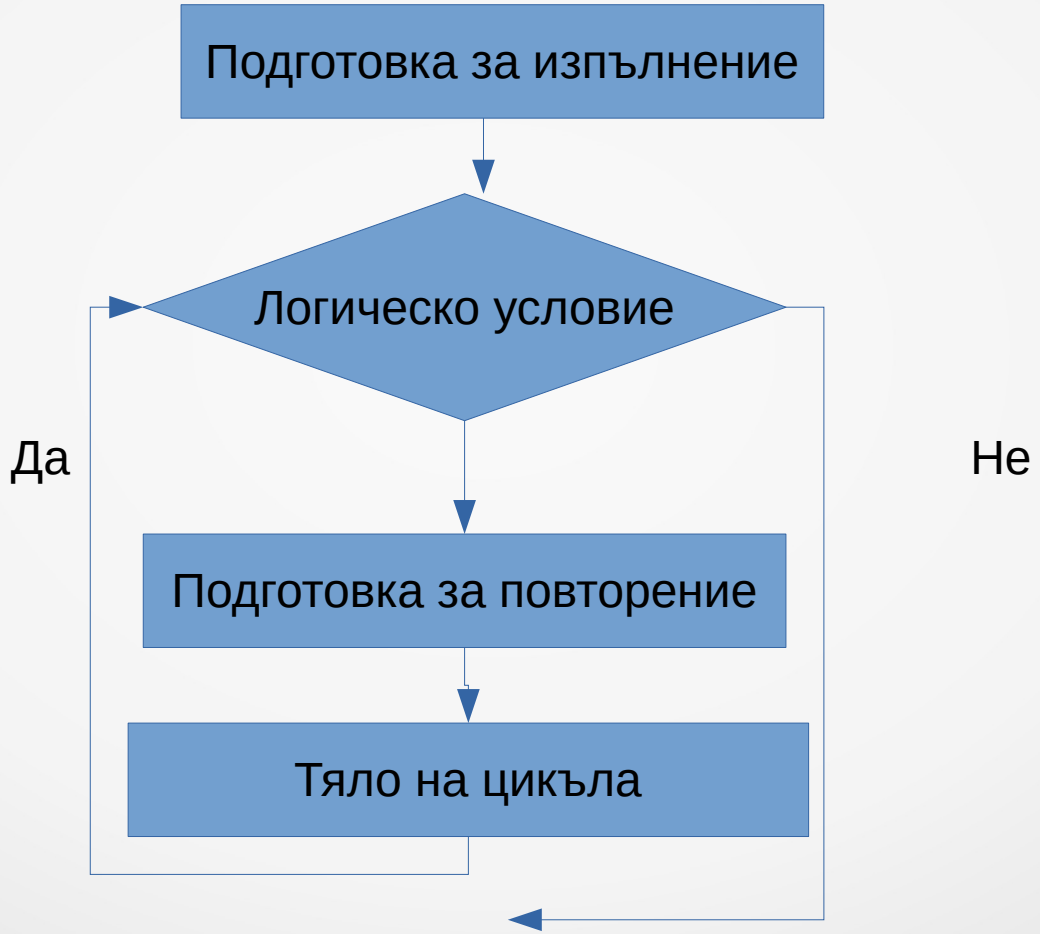
Цикъл със следусловие

```
int main(){
int a=0;
cycle1:
a++;
if (a<10)
goto cycle1;
cout<<"a="<<a;
return 0;
}
```

Цикъл с брояч

```
for(<подготовка за изпълнение> ; <условие>; <подготовка за повторение>){
//тяло на цикъла
}
```

Цикъл с брояч



Цикъл с брояч

Пример:

```
for (i=1;i<5;i++) {
printf(”%d”,i);
}
```

Оператор break

Операторът **break** е предназначен за прекъсване на изпълнението на цикли **for**, **while** и **do-while** в произволно място в тялото на цикъла. Изпълнението на оператора break предизвиква излизане от цикъла и предаване на управлението на оператора, записан непосредствено след цикъла.

На практика break представлява безусловен преход към първия ред след края на цикъла.

Често срещани грешки

- Условието за край на цикъла не се отнася до обработваната в тялото на цикъла среда
- Условието, въпреки че се отнася до обработваната в тялото среда, не се изпълнява никога
- Прескочи кобила

Пример

- Нека следната игра се играе от двама играчи.
- Играч 1 скрива предмет в една от стаите на безкраен коридор, но без първите три. Стаите са номерирани с естествените числа.
- Играч 2 търси предмета в стаите, като прескача врати и има право да ”влиза” с определена стъпка, избрана от него в началото на играта. Стъпката не може да е 1. Играч 2 тръгва от първа, от втора или от трета стая.

	0	2	3	5	a
	9	7	8	4	b
					Sum
					c
					пренос
K=4	0	1	2	3	

	0	2	3	5	a
	9	7	8	4	b
					Sum
					c
					пренос
K=4	0	1	2	3