УВОД В ПРОГРАМИРАНЕТО

СЪДЪРЖАНИЕ

- Основни оператори в C++
- Условни оператори
- if/if-else
- Вложени условни оператори
- Области на видимост на променливите
- Оператор Switch

STATEMENT(ИНСТРУКЦИЯ)

- Команди в дадена компютърна програма, които се изпълняват в определена последователност
- Инструкциите включват един или повече оператори
- В C++ има няколко основни вида оператори

ОПЕРАТОР ЗА ПРИСВОЯВАНЕ НА СТОЙНОСТ

• <променлива> = <израз>;

където

• <променлива> е идентификатор, дефиниран вече като променлива,

• <израз> е израз от тип, съвместим с типа на <променлива>

• <lvalue> = <rvalue>

• <lvalue> —място в паметта със стойност, която може да се променя

• Например – променлива

• <rvalue> —временна стойност, без специално място в паметта

ОПЕРАТОР ЗА ПРИСВОЯВАНЕ НА СТОЙНОСТ

- Съкратени оператори
- +=
- -=
- *=
- /=

Пример

a+=2 //Еквивалетно на a=a+2

a*=8 // a=a*8

ЕДНОМЕСТНИ ОПЕРАЦИИ

- ++a и –-a връщат a, което е **lvalue**
- a++ и a-- връщат **rvalue**

Увеличават стойността на “a”с единица, но имат различна същност.

СЪКРАТЕНИ ОПЕРАТОРИ ЗА ПРИСВОЯВАНЕ

- По-общо:

e1 op= e2;

е еквивалентно на

e1 = (e1) op (e2);

където op е някой от операторите

+, -, *, /, %, <<, >>, &, |, ^

АСОЦИАТИВНОСТ НА ОПЕРАТОРИТЕ

- Операторът за присвояване е дясноасоциативен
- a = (b = 2)
- <lvalue> = <rvalue>

- Освен дясно-, различаваме и ляво- асоциативни оператори
- Например аритметичните оператори са лявоасоциативни
- Какъв ще бъде резултатът от следния израз:

```
double value = 20/10/2;
```

ПРИОРИТЕТ НА ОПЕРАТОРИТЕ

Precedence	Operator	Description	Associativity
1	`::`	Scope resolution	Left-to-right
2	`++` `--`	Suffix/postfix increment and decrement	
	`()`	Function call	
	`[]`	Array subscripting	
	`.`	Element selection by reference	
	`->`	Element selection through pointer	
3	`++` `--`	Prefix increment and decrement	Right-to-left
	`+` `-`	Unary plus and minus	
	`!` `~`	Logical NOT and bitwise NOT	
	`(type)`	Type cast	
	`*`	Indirection (dereference)	
	`&`	Address-of	
	`sizeof`	Size-of	
	`new`, `new[]`	Dynamic memory allocation	
	`delete`, `delete[]`	Dynamic memory deallocation	
4	`.*` `->*`	Pointer to member	Left-to-right
5	`*` `/` `%`	Multiplication, division, and remainder	
6	`+` `-`	Addition and subtraction	
7	`<<` `>>`	Bitwise left shift and right shift	
8	`<` `<=`	For relational operators < and ≤ respectively	
	`>` `>=`	For relational operators > and ≥ respectively	
9	`==` `!=`	For relational = and ≠ respectively	
10	`&`	Bitwise AND	
11	`^`	Bitwise XOR (exclusive or)	
12	`|`	Bitwise OR (inclusive or)	
13	`&&`	Logical AND	
14	`||`	Logical OR	
15	`?:`	Ternary conditional	Right-to-left
	`=`	Direct assignment (provided by default for C++ classes)	
	`+=` `-=`	Assignment by sum and difference	
	`*=` `/=` `%=`	Assignment by product, quotient, and remainder	
	`<<=` `>>=`	Assignment by bitwise left shift and right shift	
	`&=` `^=` `|=`	Assignment by bitwise AND, XOR, and OR	
16	`throw`	Throw operator (for exceptions)	
17	`,`	Comma	Left-to-right

СТРУКТУРИ ЗА УПРАВЛЕНИЕ

Структурата на една компютърна програма условно може да бъде категоризирана в 3 основни типа:

а. Линейна последователност от действия-верига

б. Алтернатива (разклонение)

в. Цикъл(повторение)

ОПЕРАТОРИ

- **Операторите са най-малката изпълнима част в програма на C++**

За да бъде синтактично завършена конструкцията на един оператор, тя завършва със символа „ ; “. Най-простият оператор е празният и има следния вид:

```
;
```

- Когато не е посочено нищо друго, програма написана на C++ се изпълнява от вхондата точка (първия оператор на функцията main()) до края на фунцкията main() всичко се изпълнява последователно
- Но понякога логиката на програмата изисква изпълнението на определена последователност от два или повече оператора. Тогава се използват **съставни оператори.**

- **Съставен оператор (блок) е последователност от оператори заградени във фигурни скоби. Той може да се поставя навсякъде вместо прост оператор. За разлика от простия оператор, той не завършва с „ ; “.**

СЪСТАВЕН ОПЕРАТОР (БЛОК)

• Синтаксис

{<оператор1>

<оператор2>

. . .

<операторN>

}

• Семантика

• Обединява нула или повече оператора в един. Може да бъде поставен навсякъде, където по синтаксис стои оператор

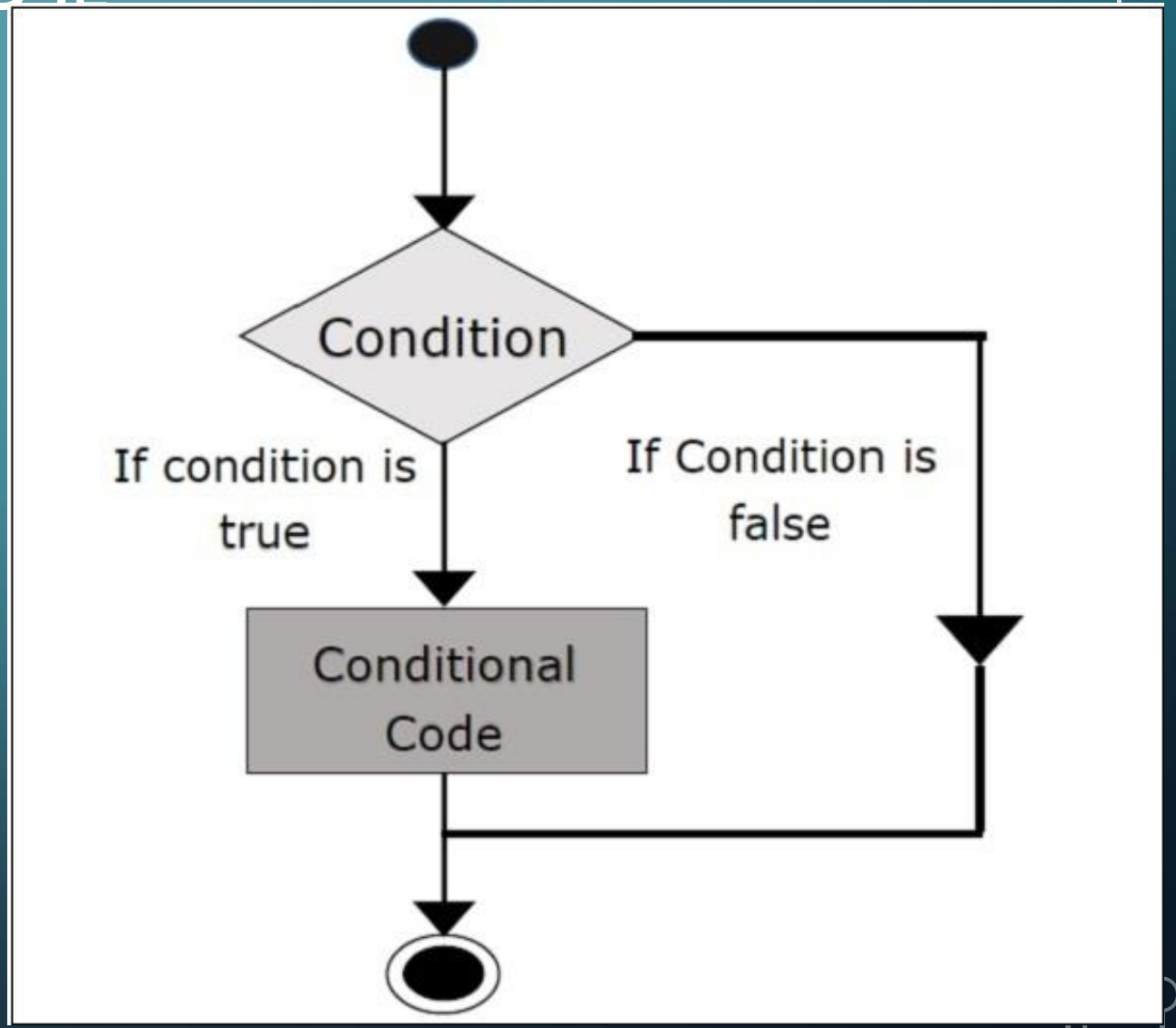
• Дефинициите в рамките на блока, се отнасят само за него, т.е. не могат да се използват извън него

• Самият блок не завършва с ;

Възможни са и вложени блокове.

УСЛОВЕН ОПЕРАТОР IF

- Чрез този условен оператор се реализира разклоняващ се изчислителен процес



УСЛОВЕН ОПЕРАТОР IF

- **Синтаксис**

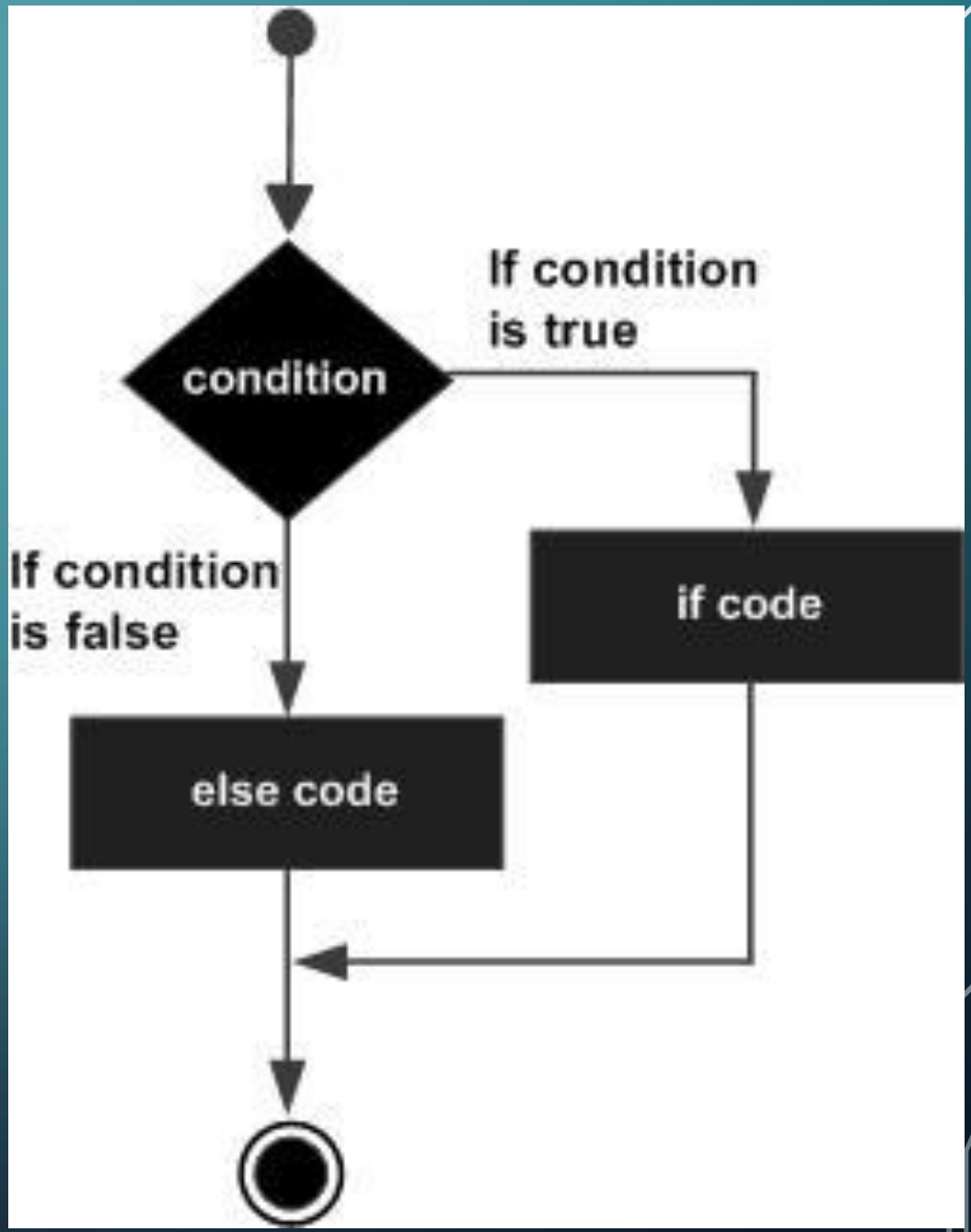
if (<условие>) <оператор>

където

- **if е запазена дума;**
- **<условие> е булев израз;**
- **<оператор> е произволен оператор.**
- **Семантика**
- Пресмята се стойността на булевия израз, представящ <условие>.
- Ако резултатът е **true**, изпълнява се <оператор>. В противен случай <оператор> не се изпълнява

ОПЕРАТОР IF/ELSE

- Операторът се използва за избор на една от две възможни алтернативи в зависимост от стойността на дадено условие.
- Чрез него се реализира разклоняващ се изчислителен процес от вида:



ОПЕРАТОР IF/ELSE

- **Синтаксис**

if (<условие>) <оператор1> else <оператор2>

където

- **if и else са запазени думи;**
- **<условие> е булев израз;**
- **<оператор1> и <оператор2> са валидни за езика оператори.**
- **Семантика**
- Пресмята се стойността на булевия израз, представящ <условие>. Ако

резултатът е **true**, изпълнява се <оператор1>. В противен случай се

изпълнява <оператор2>

ОБЛАСТ НА ВИДИМОСТ НА ПРОМЕНЛИВИТЕ (SCOPE)

- Досега споменатите блокове задават локална **олбаст на действие.** Това означава, че променлива дефинирана в даден блок „съществува“ в паметта на компютъра от момента на нейното дефиниране до края на блока, в който тя е дефинирана.
- Нарича се още „област на действие“ (scope)
- Дефиниция на променлива със същото име в същия блок е забранена
- Дефиницията на променлива във вложен блок припокрива всички дефиниции със същото име във външните блокове

ОПЕРАТОР SWITCH

Общ вид:

```
switch ( <израз> )
{
case <константен-израз-1>:
<case-израз-1>
...
case <константен-израз-N>:
<case-израз-N>
[ default:
<default-израз> ]opt
}
```

ОПЕРАТОР SWITCH

Семантика:

1. Оценява се <израз>

2. Оценката се сравнява с константните изрази

A. Ако е равна на някой от тях, влиза се в първия **case-израз** след съответния константен израз;

B. Ако не е равна на никой от тях, влиза се в **default-израза**, ако има такъв

C. В противен случай не се прави нищо.

УСЛОВЕН ОПЕРАТОР (?:)

- Общ вид:
- **<условие> ? <израз 1> : <израз 2>**
- Ако <условие> е истина (има стойност true), тогава се оценява <израз 1> и тази стойност се използва за стойност на целия израз.
- В противен случай се използва <израз2>

Пример:

```
int a = (1 < 2) ? 100 : 200;
```

БЛАГОДАРЯ ВИ!

изготвил: Божидар Навчев