Типът bool

Булев тип (bool) - тип, състоящ се от две стойности - (истина - true - 1) и (лъжа - false - 0). За всяка променлива от този тип се заделя по 1 байт памет.

Булев израз - може да се получи при прилагането на операторите за сравнение на числови изрази върху n на брой булеви променливи. Примери: (1 < 2) -> true, (x != x + 1) -> true. Внимание трябва да се обърне на оператора “==”, понеже ако се замени с “=” има голяма вероятност да получим грешен резултат. Всичко това се получава, понеже ако имаме следното условие (if (x = 0) ...), то винаги е лъжа, понеже стойността на 0 се приравнява на false, а не сравнява стойността на x с тази на 0.

Логически операции - образуваме по-сложни булеви изрази, като използваме следните логически операции: (&& - логическо “и”), (|| - логическо “или”), (! - логическо отрицание). Логическото “и”, още наречено конюнкция, има стойност истина само когато стойността на двата и операнда е истина. Логическото “или”, познато като дизюнкция, има стойност лъжа само когато стойността и на двата и операнда е лъжа. Отрицанието сменя стойността на операнда си.

Пример и изпълнението му

```
int x = 5, y = 4, z = x;
bool first = (x == z) && (y < x) && (y != z);
first = true && true && true;
first = true && true;
first = true;
```

Таблица на истинност

x	y	x && y	x || y	!x
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Block - блок в програмата ще наричаме парчетата код, които са заградени с { и }. Конвенцията за писане изисква новите редове след отварящата скоба да започват навътре от предходния ред с една табулация място.

Scope - видимост на променливите, се нарича частта от програмата, където могат да бъдат достъпвани. Видимостта може да бъде разделена на 3 категории.

1. Вътре във функция или блок - локална променлива
2. В дефиниция на функция - формален параметър
3. Извън всички функции - глобална променлива

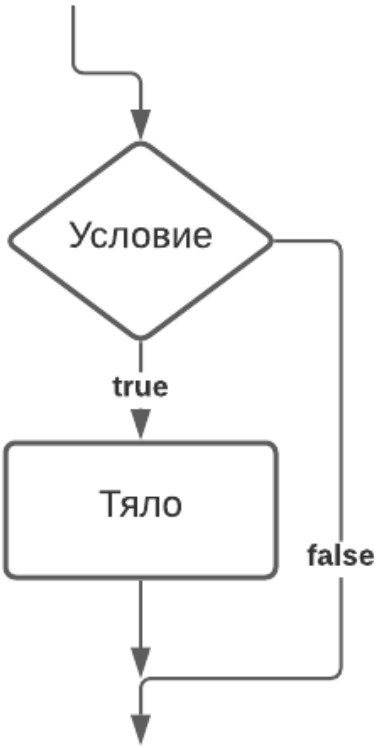
Важно - Локалните променливи могат да бъдат достъпвани в конкретния блок в който са декларирани и във всеки негов подблок. Глобалните променливи могат да бъдат достъпвани от всеки блок на програмата. Дефинират се най-отгоре, под препроцесорните директиви. Препоръчително е избягване използването на глобални променливи. Използвайте ги внимателно.

If statement

Предназначение: Контролирането потока на програмата, вземането на решение кога дадени фрагменти да се изпълнят и кога не, е важно умение за всеки програмист. За това умение се използват различни инструменти. Един такъв инструмент е условният оператор, той проверява дали дадено условие е изпълнено и спрямо този резултат изпълнява дадено следствие от програмиста. Условният оператор има две основни конструкции (форми): кратка и пълна.

Синтаксис: долупосочената конструкция, където <условие> има стойност от тип булева.

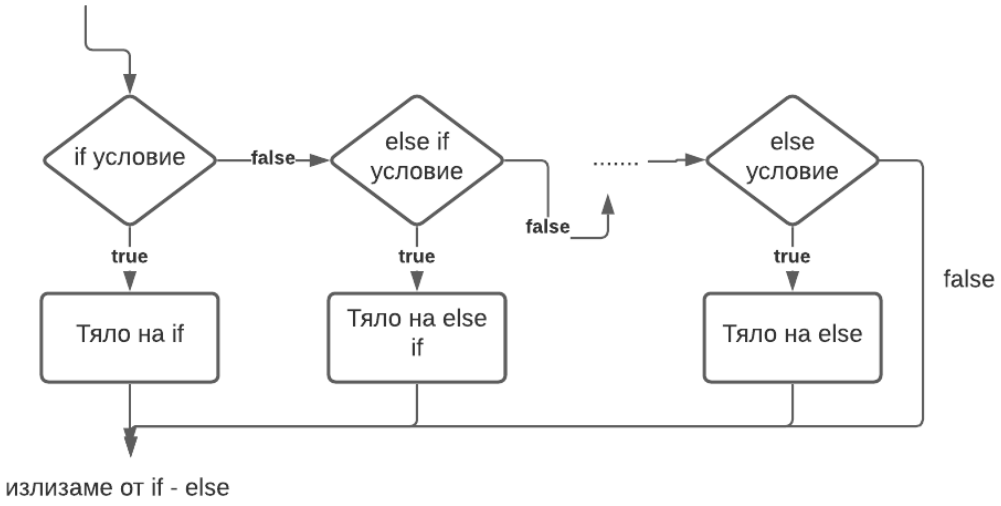
```
if (<условие>) {
        // блок
}
```



Семантика: Проверява се резултатът от <условие>, ако върнатата стойност е true тогава се изпълнява блокът от операции, ако върнатият резултат е false блокът се пропуска, т.е. няма да се изпълни нищо.

If - else: след оператор if може да бъде добавено и условие else. Между if - else може да бъдат вмъкнати произволен брой else if конструкции, които получават булев израз и биват изпълнение само ако стойността му е истина. Ако даден блок else - if бъде изпълнен, то останалите надолу по веригата биват

игнорирани, тоест излизаме от if - else конструкцията. До else изпълнението се стига само, ако всички предишни булеви изрази са били приравнени към стойност лъжа.



Тернарен оператор: форма наподобяваща тази на if - else. Операторът се записва като “?:”.

Синтаксис: <булев израз> ? <операция при истина> : <операция при лъжа>

Семантика: Изчисляваме булевия израз, ако върнатата стойност е истина, то изпълняваме първият блок, тоест след ?, а ако е лъжа, операцията след :.

Switch - case statement

Switch оператор: Операторът switch позволява тестването на дадена променлива за равенство измежду списък от стойности. Всяка стойност за проверка се нарича случай (case).

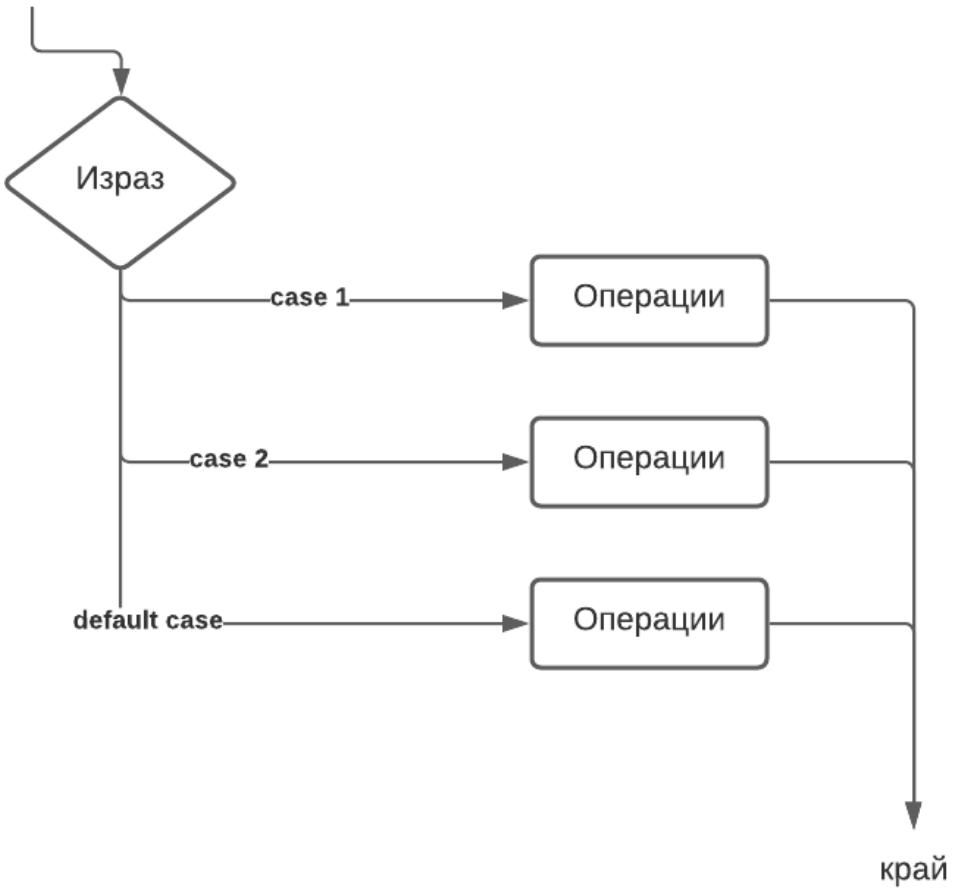
Семантика: Подаваме дадена стойност като параметър на switch, след което се проверява в кой случай стойността от този параметър ще съвпадне по стойност с някоя стойност за проверка, при което се изпълнява операциите подадени към този случай. Ако никоя стойност не съвпадне с подадената се изпълнява случаят default.

```
switch (<израз> | <стойност>) {
	case <стойност при проверка>: <операция за изпълнение>; break;
	…
	case <стойност при проверка>: <операция за изпълнение>; break;
	default: <операция при изпълнение>; break;
}
```

Правила:

- Може да имате неограничен брой случаи (case), като трябва за всеки един да дадете стойност по която да се проверява;
- Стойностите, по които проверяваме, трябва да бъдат от същия тип както подадената за проверка;
- При откриване на съвпадение се изпълняват подадените операции до срещане на оператор break, т.е ако е влезнал във втория case и няма break в него, ще се изпълнят всички оператори в останалите case-ове до достигане на оператор break;
- При стигане до break, изпълнението на оператора switch се прекратява и продължава изпълнението на програмата след него;

- Default е опционален, т.е не е от задължителния синтаксис на оператора switch. Той се използва когато никой от изброените условия не се е изпълнило и искаме да отбележим това;



Std namespace

Да приемем, че отиваме на парти в апартамента на приятел. Там е пълно с хора. Вашият приятел ви запознава с гостите: Пешо, Иван, Георги, Александра, Анди, Мария, Александра и тн. Малко по-късно, когато вече всички са на градус решавате, че ще играете игра по двойки и се опитвате да се разделите, като всеки тегли картонче с името на съотборника си. Когато идва вашия ред да

изтеглите са останали само две картончета и за съжаление и на двете пише Александра. Но вие искате да сте в отбор с по-красивата Алекс. Кое картонче за кое момиче е? Ако не е имало някаква допълнителна информация, която да ви помогне да решите кое картонче за кое момиче е (например да си спомните, че красивата Алекс винаги слага звезда до името си, когато го изписва), ще трябва да се обърнете към своя приятел и да поискате повече информация. Но това би било неуместно, защото не толкова хубава Алекс веднага би се разсърдила. А колко по-лесен щеше да и ибора, ако едната беше написала Алекс, а другата Александра..

По същия начин C++ изисква всички идентификатори (имена на променливи и/или функции) да не са двусмислени. Ако два идентификатора са въведени в една и съща програма по начин, който компилаторът не може да ги разграничи, съставителят или линкерът ще доведе до грешка. Тази грешка обикновено се нарича "конфликт с именуването". Тъй като програмите се увеличават и използват повече идентификатори, шансовете за конфликт между имената се увеличават значително.

Когато C ++ бил първоначално проектиран, всички идентификатори в стандартната библиотека C++ (като cin и cout) били на разположение за директно използване. Това обаче означава, че всеки идентификатор в стандартната библиотека може потенциално да е в конфликт с името, което сте избрали за вашите собствени идентификатори. Така че, разработчиците C++ преместили цялата функционалност на стандартната библиотека в специална област, наречена пространство с имена или namespace.

Подобно на град, който гарантира, че всички пътища в града имат уникални имена, пространството с имена гарантира, че идентификаторите в пространството на имената са уникални. Това предотвратяваме конфликтите на идентификаторите в пространство от имена с други идентификатори.

using namespace std

Оказва се, че името на оператора за изход std::cout, което използваме в повечето си примери, не е наистина "std::cout". Това всъщност е просто "cout", а "std" е името на пространството с имена, в което името живее. Цялата функционалност в стандартната библиотека C++ се дефинира в пространство с имена std (кратко за standard). По този начин не е нужно да се притесняваме за функционалността на стандартната библиотека, която има конфликт на именуване с нашите собствени идентификатори.

За да избегнем постоянното пояснение към компилатора, използваме ключовия ред using namespace std; веднага след добавянето на необходимите ни библиотеки: <iostream>, <cmath>, <string>, <vector>. Така вече няма да е необходимо в програмата ни пред всеки оператор за вход или изход да слагаме std::.