Основи на програмирането на C/C++

Всеки език за програмиране е съвкупност от правила и изключения, в която преобладават извлеченията. Общата концепция на езика C/C++ сякаш се старае да опровергае тази мисъл. Езикът има много добре замислени синтаксис и семантика, които позволяват във всички езикови конструкции да се следва принципът от общото към частното.

C/C++ е език за програмиране от средно ниво с общо предназначение. С него можеш да пишеш софтуер под DOS, Windows, Linux, софтуер за мобилни устройства, можеш да пишеш както системен така и приложен софтуер, игри. Поради ниското ниво на абстракция, програмистите имат повече контрол върху хардуера и програмите написани на него обикновено работят по-бързо от тези, написани на езици от високо ниво. Затова C/C++ е подходящ за създаване както на операционни системи, драйвер за хардуерно устройство или да програмирате микроконтролери (embedded device), поради липса на алтернатива и поради нуждата да се управлява много внимателно хардуера. То е един от най-универсалните езици използвани в момента. Много се спори дали C/C++ е подходящ за начинаещи или не - и двете твърдения са еднакво верни. За много от начинаещите C/C++ е сложен именно в работата на по-ниско ниво - указатели, управление на паметта и други. От друга страна факта, че не ти ги спестява, дава една по-добра и по-широка представа от други езици. Въпроса е в това как и колко се работи, защото в C++ има някои не-прости концепции, които е трудно да се разберат от първи път. В общи линии както и при другите езици всичко идва с практиката. Колкото повече пишеш на C/C++, толкова повече го разбираш.

Елементи на C++

Базисните елементи с които работи компилатора на C++ се наричат "токени". В сорс-програмите токените имат свои еквиваленти. Това са:

- ключови думи
- идентификатори (имена на функции и променливи)
- константи
- стрингови-литерали
- оператори
- пунктуации

Ключовите думи, идентификаторите, константите стринговите-литерали и операторите се състоят от символи, които са в основата на токените. Към пунктуациите спадат интервала, символа за нов ред, както и скобите ({}), ([]) и (()).

Допустими символи в C++

При изучаването на даден програмен език трябва да започнем с неговата азбука, тоест допустимите символи- букви, цифри и комбинация от други символи, които са необходими за изграждането на останалите елементи на езика. В основата на езика е заложена

латинската азбука. Като елементи на езика C/C++ са възприети следните символи.

Главни и малки латински букви както и символът за подчертаване.

A B C D E F G H I J K L M N O P R S T U V W X Y Z

a b c d e f g h i j к l m n o p r s t u v w x y z

_

Всички арабски цифри

0 1 2 3 4 5 6 7 8 9

служебни символи на езика:

**, . ? / ; : ' " [ ] { } ~ ! @ # $ % ^ & * ( ) - + = | **

празни символи

интервал, нов ред, табулация, коментари

Служебни думи

Служебните думи, резервирани от C++, не могат да се използват като идентификатори. Те трябва да са записани винаги с малки букви. Това се налага от обстоятелството, че езикът C++ прави разлика между големи и малки букви

В таблицата по-долу са дадени всички служебни думи в C++ според ISO C++ C++98.

Ключова дума	Предназначение
and	Алтернативен оператор на &&
and_eq	Алтернативен оператор на &= (побитово И с присвояване)
asm	Вмъкване на асемблерски инструкции
auto	Дефиниране на автоматични променливи
bitand	Алтернативен оператор на & (побитово И)
bitor	Алтернативен оператор на | (побитово ИЛИ)
bool	декларатор на логическа променлива
break	Прекъсва изпълнението на цикъл
case	Деклариране на блок за избор от switch оператор
catch	Начало на блока прихващат изключения
char	Декларатор на символна променлива
class	Декларира клас
compl	Алтернативен оператор на ~ (побитово инвертиране)
const	Декларатор на константа
const_cast	Операцията const_cast премахва или добавя спецификаторите const и volatile чрез преобразуване. Спецификаторът за тип volatile се използва, когато достъпът до обектите от този тип ще се осъществява по-начин, който не може да бъде предвиден от компилатора. Той се използва

	за подтискане на различни видове оптимизации
continue	Прекъсва текущата итерация на цикъл
default	Подразбиращ се блок от switch оператор
delete	Изтрива динамично заделена памет
do	Елемент от оператор за цикъл do while
double	Декларатор на реално число с двойна точка
dynamic_cast	Осигурява правилно преобразуване на типове само за указатели и псевдоними по време на изпълнение.
else	Алтернативна част на оператора за разклонение if
enum	Деклариране на изброим тип
exit()	Прекъсва процеса
explicit	Добавя се към обява на конструктор за избягване на неявно преобразуване.
export	Позволява в шаблони да се сепарират декларациите на спецификациите
extern	Деклариране на външни функции или променливи
false	Дефиниране на логическа неистина
float	декларатор на реално число
for	оператор за цикъл
friend	деклариране приятел на клас
goto	безусловен преход до етикет
if	оператор за разклонение
inline	декларира вградена функция
int	декларатор на цяло число
long	Деклариране на дълго число
mutable	override a const variable
namespace	поставен в началото на блок указва, че всички елементи от това пространство от имена могат да се използват в блока без да се задава името на пространството.
new	Заделя динамична памет
not	Алтернативен оператор на !(НЕ)
not_eq	Алтернативен оператор на !=
operator	Дефинира оператор в C++
or	Алтернативен оператор на || (ИЛИ)
or_eq	Алтернативен оператор на |=
private	Деклариране на частни елементи от клас
protected	Деклариране на защитени елементи от клас
public	Деклариране на публични елементи от клас
register	Указва на компилатора за променливата да задели един регистър на процесора
reinterpret_cast	Операцията reinterpret cast се използва за

	преобразуване на един тип указател към друг тип указател. Използва за преобразуване на указател към цяло число и обратно.
short	Модификатор за Деклариране на късо число
signed	Модификатор за Деклариране на число със знак
sizeof	Връща размера на обект в байтове
static	Модификаторът static се използва за промяна времето за съществуване или видимостта на променливите.
static_cast	Операцията **static_cast** служи за преобразуване в случаите, в които проверката на типа се извършва по време на компилация. С нея се извършват стандартни преобразувания – например **void*** в **char*, int** в **double** и т.н., както и обратните им преобразувания;
struct	Декларира структура
switch	Оператор за многопосочно разклонение
template	Дефинира шаблон
this	Указател към текущия обект
throw	"създава" ("генерира") изключение
true	Дефиниране на логическа истина
try	Начало на блока за изключения
typedef	Деклариране на потребителски типове
typeid	Връща информация за типа по време на изпълнение
typename	Деклариране на име на клас при темплейти
union	Деклариране на обединение
unsigned	Модификатор за деклариране на число без знак
using	Включва конструкция за промяна пространството на видимост на променливите
virtual	Деклариране на виртуална функция
void	Деклариране на функция или данни които не са асоциирани с даден тип. Използва се за обявяване на функция, която не връща стойност
volatile	Той показва, че обектът може да бъде променян не само от програмиста, но и от процеси извън програмата, като прекъсванията или входно/изходни портове
wchar_t	Деклариране на wide-character променлива
while	Оператор за цикъл
xor	Алтернативен оператор на ^ (изключващо или)
xor_eq	Алтернативен оператор на ^= (изключващо или с присвояване)
#if	Предпроцесорна директива
#ifdef	Предпроцесорна директива
#ifndef	Предпроцесорна директива

#elsif	Предпроцесорна директива
#else	Предпроцесорна директива
#endif	Предпроцесорна директива
#include	Предпроцесорна директива включване на файл
#define	Предпроцесорна директива за деклариране на константи и макроси.
#undef	Предпроцесорна директива
#line	Предпроцесорна директива
#pragma	Предпроцесорна директива
#error	Предпроцесорна директива

При C++ са добавени следните ключови думи:

C++ ключови думи				
asm	dynamic_cast	namespace	reinterpret_cast	try
bool	explicit	new	static_cast	typeid
catch	false	operator	template	typename
class	friend	private	this	using
const_cast	inline	public	throw	virtual
delete	mutable	protected	true	wchar_t

Горе посочените резервирани думи могат да се разделят на следните групи по предназначение:

Оператори за управление в C				
break	default	for	return	typedef
case	do	goto	switch	while
continue	else	if		

Типове данни в C/C++			
char	int	double	float
enum	union	struct	

Модификатори на изрази и оператори в C/C++			
asm	extern	register	unsigned
auto	fastcall	short	volatile
const	inline	signed	
cdecl	long	static	
export	pascal	stdcall	

Модификатори на изрази и оператори в C++		
friend	inline	private
protected	public	virtual

Предпроцесорни директиви			
#if	#elsif	#include	#line
#ifdef	#else	#define	#pragma
#ifndef	#endif	#undef	#error