Преобразуване на типове в C++

C++ ни дава възможността да преобразуваме (кастваме) данни от един тип в друг. Данните се преобразуват автоматично в случаите, в които, неформално казано, компилаторът очаква един тип, а получва друг. Можем да попаднем в такава ситуация, например, когато:

- приравним променлива от един тип на израз от друг тип:
- стойността, върната от функция, е от тип, различен от типа на функцията
- приложим функция или операция, предназначена за един тип, върху друг

В тези случаи компилаторът прочита стойността от подадения тип, но вместо с нея, работи с подходяща стойност от другия. Ето кои са най-честите преобразувания и резултатът от тях:

Кастване	Върната стойност	Пример

double —> int	Числото без цифрите след десетичната запетая	double a = 4.49;
		int b = a;
		a = –3.1416;
		int c = a;
		cout << b << endl; // 4
		cout << c << endl; // –3

char —> int	ASCII кодът на символа	int number = 'A'; cout << number; // 65
int —> char	Символът със съответния ASCII код	char sym = 67; cout << sym; // C

char —> bool	false, ако символът е '\0' (символът с ASCII код 0), true, ако е друг символ
	
	Това ни позволява да поставим символ като условие на if или while.
		if ('a')
		  cout << "a is true" << endl;
		else
		  cout << "a is false" << endl;
		
		if ('\0')
		  cout << "this special 0 is true" << endl;
		else
		  cout << "this special 0 is false" << endl;
		
		if ('0')
		  cout << "the normal 0 is true" << endl;
		else
		  cout << "the normal 0 is false" << endl;
		
		// На екрана:
		// a is true
		// this special 0 is false
		// the normal 0 is true

int —> bool	false, ако числото е 0,
	true, ако не е 0
	
	Това ни позволява да поставим число или аритметичен израз като условие на if или while.
		int grade = 6;
		if (grade)
		  cout << “6 is true” << endl;
		else
		  cout << “6 is false” << endl;
		
		grade = 2;
		if (grade)
		  cout << “6 is true” << endl;
		else
		  cout << “6 is false” << endl;
		
		if (0)
		  cout << “0 is true” << endl;
		else
		  cout << “0 is false” << endl;
		
		// На екрана:
		// 6 is true
		// 2 is true
		// 0 is false

bool —> int	1, ако е true,
	0, ако е false
	
	Това ни позволява да включваме булеви променливи в аритметични изрази
		int x = true;
		int y = false;
		bool z = true;
		
		cout << x * 9 + y * 6 << ” ”;
		cout << z * 9 + y * 6 << endl ;
		
		// 1 * 9 + 0 * 6 = 9, затова
		// на екрана ще имаме:
		// 9 9

Можем да напишем изрично с какъв тип искаме да работим. Горните правила за върнатата стойност са в сила. Един случай, в който това е полезно е, когато в израз с два типа данни компилаторът предпочете единия тип, а ние искаме другия. За целта пишем <желаният тип>(<израз>) или (<желаният тип>) <изразът>.

Примери:

```
cout << int (21.04) << endl;          // На екрана: 21
cout << (int) 21.04 << endl ;         // Двата реда са еквивалентни: 21

// Да предположим, че искаме да изведем третата буква след A

cout << ‘A’ + 3 << endl;              // Компилаторът „избира“ да отпечата число: 68

cout << char(‘A’ + 3) << endl;        // Сега вече няма как – ще се изведе символ: D
```