Podstawy programowania w C++

Zmienne typu znakowego

Opracował: Andrzej Nowak

Bibliografia:

CPA: PROGRAMMING ESSENTIALS IN C++ https://www.netacad.com

ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) to standard kodowania znaków najczęściej używany w prawie wszystkich nowoczesnych urządzeniach (takich jak komputery, drukarki, telefony komórkowe, tablety itp.) Kod pozwala na zakodowanie 256 różnych znaków, ale nas interesuje w zasadzie tylko pierwsze 128.

Komputery przechowują znaki jako liczby. Każda postać używana przez komputery odpowiada unikalnej liczbie i na odwrót, a jest o wiele więcej znaków, niż można by się spodziewać. Wiele z nich jest niewidocznych dla ludzi, ale niezbędnych dla komputerów.

Niektóre z tych znaków są nazywane "białymi przestrzeniami", podczas gdy inne nazywa się "znakami kontrolnymi", ponieważ ich celem jest sterowanie urządzeniami wejścia / wyjścia.

Przykładem białej przestrzeni, która jest całkowicie niewidoczna gołym okiem, jest specjalny kod lub para kodów (różne systemy operacyjne mogą traktować to zagadnienie inaczej), które są używane do oznaczania końców linii w plikach tekstowych.

Ludzie nie widzą tych znaków, ale widzą efekt ich zastosowania w miejscach, w których linie są zerwane.

Jeśli chcesz sprawdzić, w jaki sposób konstruowany jest kod, zajrzyj do tabeli

Character	Dec	Hex		Character	Dec	Hex		Character	Dec	Hex		Character	Dec	Hex
(NUL)	0	0		(space)	32	20		@	64	40		`	96	60
(SOH)	1	1		!	33	21		A	65	41		a	97	61
(STX)	2	2		"	34	22		B	66	42		b	98	62
(ETX)	3	3		#	35	23		C	67	43		c	99	63
(EOT)	4	4		$	36	24		D	68	44		d	100	64
(ENQ)	5	5		%	37	25		E	69	45		e	101	65
(ACK)	6	6		&	38	26		F	70	46		f	102	66
(BEL)	7	7		'	39	27		G	71	47		g	103	67
(BS)	8	8		(	40	28		H	72	48		h	104	68
(HT)	9	9		)	41	29		I	73	49		i	105	69
(LF)	10	0A		*	42	2A		J	74	4A		j	106	6A
(VT)	11	0B		+	43	2B		K	75	4B		k	107	6B
(FF)	12	0C		,	44	2C		L	76	4C		l	108	6C
(CR)	13	0D		-	45	2D		M	77	4D		m	109	6D
(SO)	14	0E		.	46	2E		N	78	4E		n	110	6E
(SI)	15	0F		/	47	2F		O	79	4F		o	111	6F
(DLE)	16	10		0	48	30		P	80	50		p	112	70
(DC1)	17	11		1	49	31		Q	81	51		q	113	71
(DC2)	18	12		2	50	32		R	82	52		r	114	72
(DC3)	19	13		3	51	33		S	83	53		s	115	73
(DC4)	20	14		4	52	34		T	84	54		t	116	74
(NAK)	21	15		5	53	35		U	85	55		u	117	75
(SYN)	22	16		6	54	36		V	86	56		v	118	76
(ETB)	23	17		7	55	37		W	87	57		w	119	77
(CAN)	24	18		8	56	38		X	88	58		x	120	78
(EM)	25	19		9	57	39		Y	89	59		y	121	79
(SUB)	26	1A		:	58	3A		Z	90	5A		z	122	7A
(ESC)	27	1B		;	59	3B		[	91	5B		{	123	7B
(FS)	28	1C		<	60	3C		\	92	5C		|	124	7C
(GS)	29	1D		=	61	3D		]	93	5D		}	125	7D
(RS)	30	1E		>	62	3E		^	94	5E		~	126	7E
(US)	31	1F		?	63	3F		_	95	5F			127	7F

Przyjrzyj się uważnie, ponieważ dzieje się tu kilka interesujących rzeczy.

Sprawdź, jaki jest kod dla małej litery "a". To 97, prawda?

A teraz znajdź wielką literę "A". To 65.

Teraz odejmij kod "a" od "A", a co otrzymasz? 32!

Tak, to kod dla (ang. space)przestrzeni. Wkrótce użyjemy tej interesującej funkcji kodu ASCII.

Litery są ułożone w tej samej kolejności, co w alfabecie łacińskim.

Uwaga:

Kod **`ASCII`** jest zastępowany (lub raczej rozszerzany) przez nowy międzynarodowy standard o nazwie **`UNICODE`**.

Na szczęście zestaw `ASCII` jest podzestawem `UNICODE`.

`UNICODE` jest w stanie reprezentować praktycznie wszystkie postacie kodowania znaków używane na całym świecie.

Do tej pory traktowaliśmy język C ++ (i sam komputer) jako narzędzie do wykonywania obliczeń na liczbach. Jest to zgodne z powszechnym przekonaniem, że komputer jest tylko kalkulatorem, aczkolwiek bardzo inteligentnym.

Wiesz, że to nieprawda, ponieważ komputer może być łatwo wykorzystany do przetwarzania tekstu.

Możemy zdefiniować "słowo" jako ciąg znaków (liter, cyfr, znaków interpunkcyjnych itp.).

Teraz jednak zignorujemy łańcuch składający się z wielu znaków, a skupimy naszą uwagę na pojedynczych znakach.

Przetwarzanie znaków sprowadza się do problemu przetwarzania ciągów/łańcuchów znakowych, przetwarzanie których w języku C++ realizuje się za pomocą tablic.

Zmienna znakowa char

char (skrót od słowa ang. "character - postać ") - specjalny rodzaj danych służący do przechowywania i manipulowania znakami .

```
char znak;
```

W jaki sposób używamy wartości typu char w języku C ++?

Możemy to zrobić na dwa sposoby, z których oba różnią się nieznacznie.

1. Pierwszy sposób pozwala nam określić sam znak ujęty w pojedyncze cudzysłowy (apostrofy).

   ```
   znak = 'A';
   ```

2. Druga metoda polega na przypisaniu nieujemnej liczby całkowitej, która jest kodem `ASCII` pożądanego znaku.

   Oznacza to, że zapis widoczny poniżej spowoduje wstawienie litery "`A`" do zmiennej `znak`.

   ```
   znak = 65;
   ```

Uwaga:

Metoda druga jest mniej godna polecenia , ponieważ jest nieczytelna dla ludzi bez znajomości kodu ASCII.

Po drugie, dziwne, ale wciąż prawdziwe, istnieje znaczna liczba komputerów na świecie, które używają kodów innych niż ASCII.

Na przykład wiele komputerów mainframe IBM używa kodu zwanego powszechnie EBCDIC (Extended Binary Coded Decimal Interchange Code), który jest bardzo różny od ASCII i opiera się na radykalnie różnych koncepcjach.

Literały

Literał jest symbolem, który jednoznacznie identyfikuje jego wartość.

Definicja z Wikipedii:

Literał – jednostka leksykalna reprezentująca ustaloną wartość (liczbową, tekstową itp.) wpisaną przez programistę bezpośrednio w danym miejscu w kod programu.

Wybierz definicję, która Twoim zdaniem jest bardziej przejrzysta i spójrz na następujące proste przykłady:

`Postac` - to prawdopodobnie nazwa zmiennej; kiedy na to patrzysz, nie możesz odgadnąć, jaka wartość jest aktualnie przypisana do tej zmiennej

`"A" –` kiedy na niego spojrzysz, natychmiast poznasz jego wartość; wiesz nawet, że jest to literał typu char

`100` - to również literał (typu int)

`100.0` - tym razem typ float

`i + 100` - jest to kombinacja zmiennej i literału połączona z operatorem +; taka struktura nazywa się wyrażeniem.

Użycie znaku \ (backslash)

**** - odwrotny ukośnik (ang. *backslash*) działa jak tzw. znak ucieczki/przeskoku, ponieważ przez użycie \ można „uciec” od normalnego znaczenia znaku następującego po ukośniku.

```
znak = '\';
```

Możesz także użyć znaku ucieczki/przeskoku (*backslash)*, aby uciec/przeskoczyć przed postacią znaku ucieczki/przeskoku.

```
znak = '\\';
```

\n - oznacza przejście do nowej linii i jest czasami nazywane LF (Line Feed), ponieważ drukarki reagują na tę postać, przesuwając papier do przodu o jedną linię tekstu.

```
napis = "Ala ma kota \n";
```

\r - oznacza powrót do początku linii i jest czasami nazywany CR (powrót karetki, był synonimem "głowicy drukującej" w czasach przed-cyfrowych); drukarki odpowiadają na tę postać tak, jakby kazano im ponownie rozpocząć drukowanie od lewego marginesu już wydrukowanej linii.

Aby drukarka zaczęła drukować nową linię, musisz wysłać te dwa znaki w określonej kolejności: LF, aby wysunąć papier i CR, aby przesunąć głowicę drukującą na początek nowej linii.

\a (jak w alarmie) jest reliktem przeszłości, gdy do komunikowania się z komputerami często używano telegrafów (czy wiesz, czym jest teletekst?); wysłanie tego znaku do teletype włącza jego dzwonek, stąd ten znak jest oficjalnie nazywany BEL (jako dzwonek); jeśli wyślesz te znaki na ekran, usłyszysz dźwięk - nie będzie to prawdziwy dzwonek, ale krótki sygnał dźwiękowy.

\0 (uwaga: znak po odwrotnym ukośniku to zero, a nie O): wywoływana jest wartość null (od łacińskiego słowa nullus - none) jest postacią, która nie reprezentuje żadnego znaku.

Przykłady użycia znaku backslash

Zadanie 1

Jaki znak zostanie wyświetlony po wykonaniu instrukcji:

```
znak = '\47';
```

Cyfry po odwrotnym ukośniku to cyfry ósemkowe (cyfry z zakresu od 0 do 7). Numer zakodowany w ten sposób będzie traktowany jako wartość ASCII.

Odpowiedź:

> Liczba ósemkowa 47 równa się liczbie 39 dziesiętnie. Z tabeli kodów ASCII odczytujemy, że jest to kod ASCII dla apostrofu.
>
> (ale tylko dla komputerów implementujących kod ASCII).

Zadanie 2

Jaki znak zostanie wyświetlony po wykonaniu instrukcji:

```
znak = '\x27';
```

Cyfry po odwrotnym ukośniku to cyfry szesnastkowe - litera X (mała lub wielka litera - nie ma znaczenia).

Odpowiedź:

> Jak pewnie się domyślasz, liczba 27 w systemie szesnastkowym równa się liczbie 39 w systemie dziesiętnym. W efektem wykonania będzie wyświetlony znak apostrofu.

UWAGA:

W języku C ++ istnieje założenie, które może wydawać się zaskakujące:

typ `char` jest traktowany jako szczególny rodzaj typu `int`.

To znaczy że:

Zawsze możesz przypisać wartość `char` do zmiennej `int`, ale tylko w zakresie liczb od 0 do 127

Wartość typu `char` może być przedmiotem tych samych operacji co dane typu `int`.

Zadanie 1

Jakie wyniki zostaną wyświetlone w wyniku wykonania poniższego fragmentu programu?

```
char Char;


Char ='A';
Char += 32;
Char -= ' ';
```

Zadanie 2

Jakie wyniki zostaną wyświetlone w wyniku wykonania poniższego fragmentu programu?

```
char Char;


Char = 'A' + 32;
Char = 'A' + ' ';
Char = 65 + ' ';
Char = 97 - ' ';
Char = 'a' - 32;
Char = 'a' - ' ';
```