C 言語検定必勝プリント　明日のために３級編

　第１回目に通常の変数、第２回目に char 型変数をやったのを覚えていますか。その第２回目の下のほうでこんなものがあったこと覚えていますか。

›例）Hellow　を char 型変数に入れてみる
›
›　char　a1='H', a2='e', a3='l', a4='l', a5='o', a6='w'　　こりは大変・・・
›文字列が変更になるとその部分変数を増やしたり減らしたりする必要
›があるし、順番もきちんと管理しなければいけなくなります。
›これは本当に大変！

　思い出しましたか？これを簡単に解決する方法として配列があります。でも仕組みは簡単です。一言で言うと**きちんと順番に並んでいる変数のかたまり**ってとこでしょうか。

　例）変数と配列の関係

　変数　char a='H';

a
'H'

　配列　char a[8]="Hello!"⇒

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
'H'	'e'	'l'	'l'	'o'	'w'	'!'	¥0

　配列　char a[8]="bus"⇒

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
'b'	'u'	's'	¥0	???	???	???	???

　この場合変数 a は単に'H'という文字が入っているだけになりますが、配列 a の場合(添え字がない状態)文字のデータが順番に入っている領域の先頭アドレスに意味が変わります。
　配列はもともと大きさが決まっています。そこで先頭アドレスさえ分かればそれから続くデータを簡単に参照できるようになります。
　文字列の場合、すべての領域にデータが入っているとは限りませんので、文字列の終わりに終端記号（ヌル）が入ります。だから 8 文字分とっていても 7 文字しか入らないので注意してください。通常は文字配列を定義する場合に余裕を持って定義しておきます。

　多次元配列の場合、次のようになります。
　配列　char a[2][8] = {　"Hello!",
　　"bus"}⇒

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
'H'	'e'	'l'	'l'	'o'	'w'	'!'	¥0
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]
'b'	'u'	's'	¥0	???	???	???	???

　最初の添え字が大きなブロック(ここで言うと行ブロック?)最後の添え字がその行の中の列を指定することになります。

ちなみに 1 文字を表す場合
‘　シングルクォーテーション
ex) 'H'

文字列は
“　ダブルクォテーション
ex) "Hello"

違いが分かるかな？
'H'

0	1	2
'H'	??	??

"H"

0	1	2
'H'	'¥0'	??

※文字列には終端を表す文字が付加される