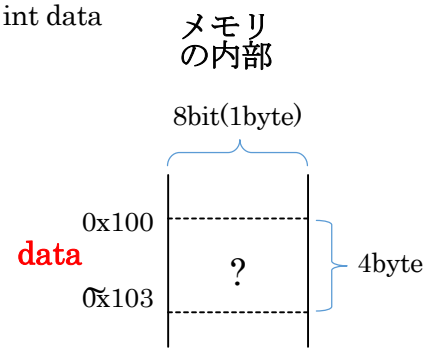
2016.7.20 小林 勝

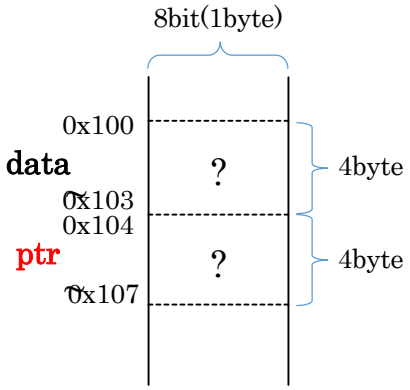
ポインタ学習プログラム(t001.c)の動作＜ポインタと変数＞

行番：６　int data



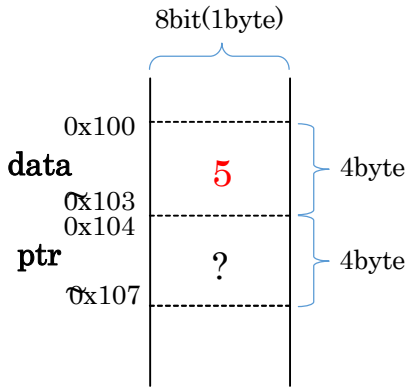
整数型の変数(data)を宣言すると、メモリ内のあるアドレス（0x100）から 4byte 分(0x100~0x103)の領域が確保される。このアドレスは、コンパイラがコンパイル時に決定する。

行番：７　int *ptr



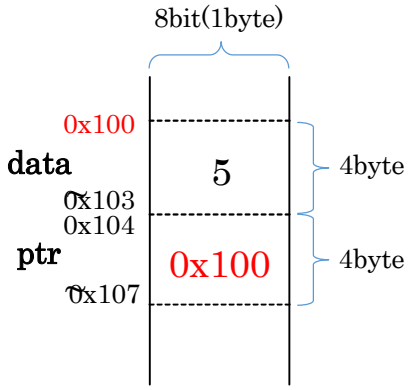
整数型のポインタ変数(ptr)を宣言すると、メモリ内のあるアドレス(0x104)から 4byte 分(0x104~0x107)のポインタ領域が確保される。通常は、先に宣言された変数のアドレスに連続してアドレスが割り当てられる。

行番：９　data = 5



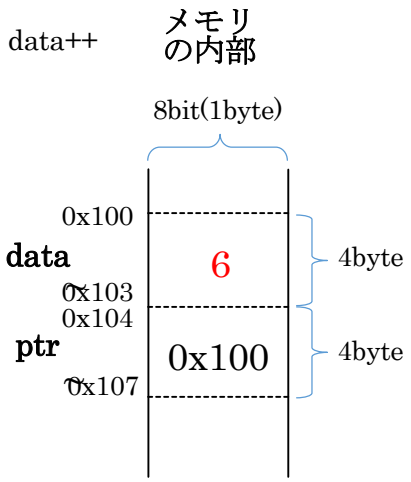
先に宣言した整数型の変数(data)のメモリ内の領域(0x100~1x103)に値:”5”を格納する

行番：２２　ptr = &data



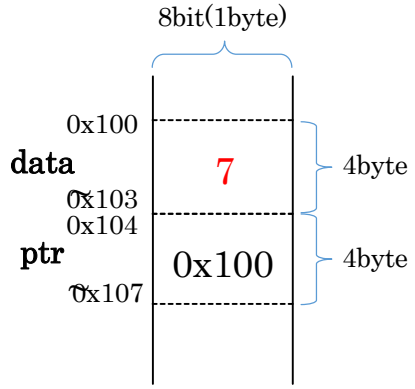
先に宣言した整数型のポインタ変数(ptr)のメモリ内の領域(0x104~1x107)に、変数 data の開始アドレス(0x100)を格納する

行番：３１　data++



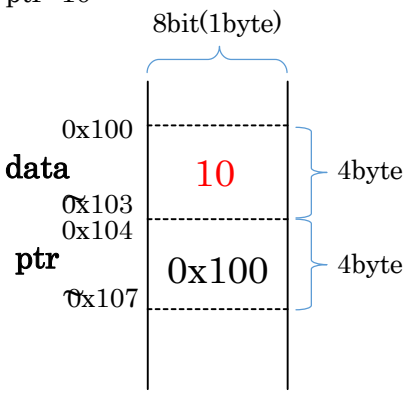
変数 data のメモリ内の領域(0x100~1x103)に格納した値:”5”に値:”1”を加えて、同領域に格納する

行番：３９　*ptr+=1



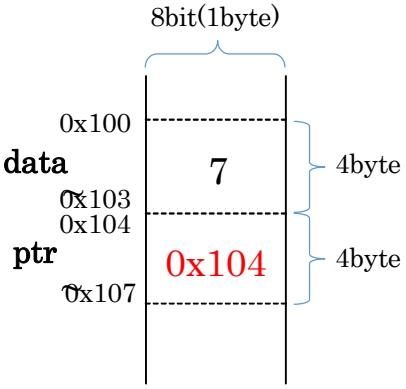
ポインタ変数 ptr に格納した（開始）アドレス(0x100)に収められている値:”6”に値:”1”を加えて、同領域に格納する。
【注意】アスタリスク(*)を付けた変数は、インクリメント演算子(++)が使えない

行番：４９　*ptr=10



ポインタ変数 ptr に格納した（開始）アドレス(0x100)から 4byte 分の領域に値:”10”を格納する。行番号 39 で実行した結果、変数 data の値:”7”は、値:”10”に変わる

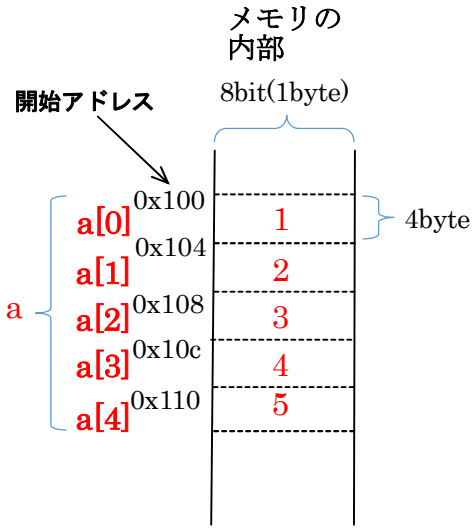
行番：５９　ptr++



ポインタ変数 ptr に格納した、変数 data の（開始）アドレスに値:”1”を加えて、同領域に格納する。加えた値は”1”であるが、アドレスでは、4Byte を加えたことになる

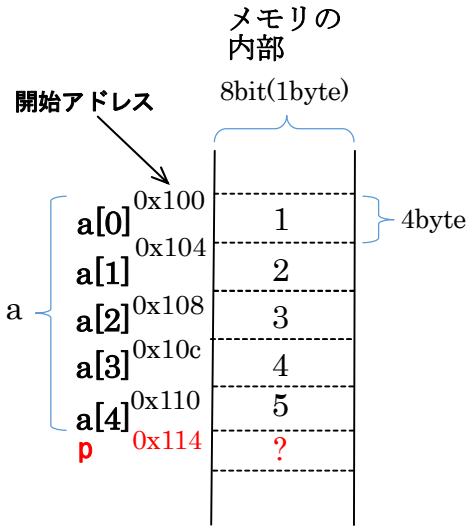
ポインタ学習プログラム(t002.c)の動作＜ポインタと配列＞

行番：6　int a[5] = {1, 2, 3, 4, 5}



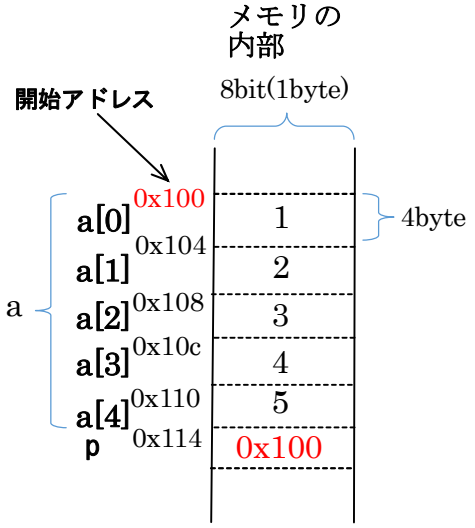
整数型の配列変数 a[5]={1,2,3,4,5}を宣言すると、各配列要素にそれぞれ値 1,2,3,4,5 を格納する。メモリでは、あるアドレス（0x100）から 4byte 毎に各配列要素の領域が確保される。このアドレスは、コンパイラがコンパイル時に決定する。

行番：7　int *p



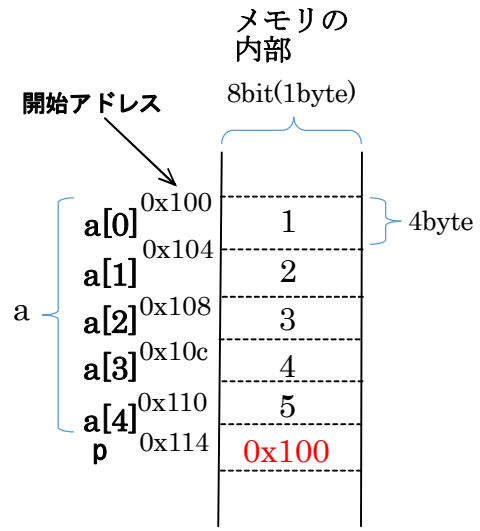
整数型のポインタ変数 p を宣言すると、メモリにあるアドレス(0x114)から 4byte 分のポインタ領域が確保される。通常は、先に宣言された変数に連続したアドレスが割り当てられる。

行番：１５　p = a



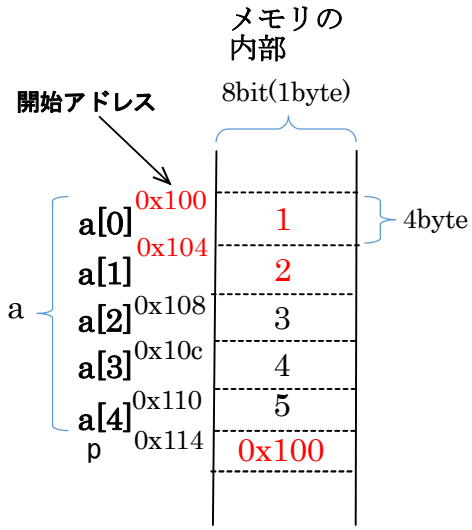
先に宣言したポインタ変数 p のメモリの領域(0x114~0x117)に、先頭の配列要素 a[0]の開始アドレス(0x100)を格納する

行番：１９　printf("ポインタ変数 p の中身= %p¥n",p)



ポインタ変数 p の領域(0x114~1x117)に格納されている内容を 16 進数で表示する。
ポインタなどの変数に格納されているアドレスを、出力（表示）する場合は、%p を用いる。16 進数で表示される。
変数に格納されているアドレスでない値を、16 進数で表示する場合は、%x を用いる

行番：２４　printf("a[0] 値= %d アドレス= %p¥n",*p,p)
行番：２５　printf("a[1] 値= %d アドレス= %p¥n",*(p+1),p+1)



先頭の配列要素 a[0]の値:”1”は*(p+0)または*p で、アドレス:”0x100”は p で取り出すことができる。また、２番目の配列要素 a[1]の値:”2”は*(p+1)で、アドレス:”0x104”は p+1 で取り出すことができる。