コンピュータ実習

第4回 2025年10月30日

C言語の基礎

前回の続き、配列、ポインタ、構造体

担当：相澤直人・宍戸博紀

コンピュータ実習（第4回）

前回の続き
　条件判断・ループ処理

配列

ポインタ

（構造体）

コンピュータ実習（第4回）

Web教材の本日のページの前回の続き➡3番目の項目（条件判断とループ）を開いてください．

コンピュータ実習（第3回）

条件判断（Web教材　条件判断とループ）

条件によって処理を振り分ける場合に使用する．条件には比較演算子や論理演算子を使った条件式を指定します．

if (条件式) { 真の処理; else 偽の処理;}
真：条件式を満たしたとき
偽：条件式を満たさないとき
条件式：比較演算子　＝＝　＜　＞　＜＝　＝＞　！＝

switch (整数値式) { case n:処理; break; case n:処理; break; ...}
整数値式：整数の値
n: 整数値式で示される実際の整数の値

コンピュータ実習（第3回）

if文の例

この例では整数変数aが100より大きいか、等しいか、それ以外かで動作を3通りに変えています。

```
if ( a > 100 )
    {
      printf("aは100より大きい¥n");
    }
else if ( a == 100 )   /* "=="であることに注意！ "="では代入になります */
  {
    printf("aは100と等しい¥n");
  }
else   /* 上記のどちらでもない*/
  {
    printf("aは100より小さい¥n");
  }
```

コンピュータ実習（第3回）

switch文の例

この例では整数変数aの値により画面への表示を3通りに変えています。

```
switch( a )
  {
    case 1:
      printf("aは1だよ！¥n");
      break;
  case 2:
    printf("aは2だよ！¥n");
      break;
  case 3:
    printf("aは3だよ！¥n");
      break;
  default:
    printf("aは1でも2でも3でもないよ！¥n");
}
```

コンピュータ実習（第3回）

ループ

繰り返し処置を効率よく行うための制御文です．for文では普通はカウンタを用意して，その値によって何回繰り返すかを決めます．繰り返し回数が決まっていない場合にはwhile文を使用します．

for (初期条件; 繰返し条件; 繰返し時処理;){処理;}

for(n=1; n<=10; n++;){・・・・}

nを1から始めて，nが10以下の時は，nに1を加えろ

その間 {}の中を実行

while(繰返し条件){処理;}

n=1;

while(n<=10){・・・; n++;}

条件式がないと無限ループする

break; ループ終了　continue; ループ先頭へ

コンピュータ実習（第3回）

for文の例

iが0から9まで1ずつ増えながら繰り返し10回ループする例

```
int i;
for ( i=0; i<10; i++)     /* i++は i=i+1の意味*/
  {
   printf("i=%d¥n", i);
  }
```

条件式がないと無限ループ
break; ループ終了
continue; ループ先頭へ

コンピュータ実習（第3回）

while文の例

iが0から9まで1ずつ増えながら繰り返し10回ループする例

```
int i;
  i=0;
  while ( i<10 )
{
  printf("i=%d¥n", i);
  i++;
}
```

条件式がないと無限ループ
break; ループ終了
continue; ループ先頭へ

コンピュータ実習（第3回）

練習課題その４

じゃんけんゲームのプログラムを完成させてみてください

じゃんけんゲームの流れ

1. ゲーム終了フラグに0を代入する .
2. ゲーム終了フラグが1でなければ次のループを繰り返す .
   (1) printf(...)を用いてメニューを表示する
   「1.グー 2.チョキ 3.パー 4.終了」
   (2) scanf(...)を用いて整数を入力する.
   (3) if**文**で入力された数字を調べる.
   4ならゲーム終了フラグに”1”を代入する.
   4でなければ以下を実行する.
   (1)コンピュータの手を作る(drand48()を使用)
   (2)if文でコンピュータとプレーヤーの勝敗を判定し,
   結果を表示する.
1. ゲーム終了メッセージを表示する.

コンピュータ実習（第3回）

練習課題その４

じゃんけんゲームのプログラムを完成させてみてください

じゃんけんゲームプログラムの流れ

0. 必要な変数を準備

```
int game_end;          /* ゲーム終了フラグ */
int player;            /* プレーヤーの手 */
int computer;          /* コンピュータの手 */
```

コンピュータ実習（第3回）

練習課題その4

じゃんけんゲームプログラムの流れ

1. ゲーム終了フラグに0を代入する .
2. ゲーム終了フラグが1でなければ次のループを繰り返す .

```
game_end = 0;        /* ゲーム終了フラグの初期化*/

/* ゲームが終了するまでのループ */
while ( game_end != 1 ){
    /* メニュー表示*/
    printf("1. グー  2.チョキ  3.パー  4.終了¥n");
    printf("input>");
    /* 入力(変数playerに入る) */
    scanf("%d", &player);

    /* 入力が4なら*/
    if ( player == 4 ){
        game_end = 1;     /* ゲーム終了フラグを1にする */
    }
    /* 入力が1~3なら*/
    else{
    /* 入力が1~3の時の処理*/
    }
}
```

コンピュータ実習（第3回）

練習課題その4

じゃんけんゲームプログラムの流れ

2. の続き：ループ内での処理

(1) printf(...)を用いてメニューを表示する
「1.グー 2.チョキ 3.パー 4.終了」済

(2) scanf(...)を用いて整数を入力する. 済

(3) if文で入力された数字を調べる. 済

4ならゲーム終了フラグに”1”を代入する. 済

4でなければ以下を実行する.

(1)コンピュータの手を作る(drand48()を使用)

(2)if文でコンピュータとプレーヤの勝敗を判定し，
判定結果を表示する.

コンピュータ実習（第3回）

練習課題その4

(1) コンピュータの手を作る(drand48()を使用)
(2) if文でコンピュータとプレーヤの勝敗を判定し，判定結果を表示する

```
int main(void) {
    int game_end;    /* ゲーム終了フラグ */
    int player;    /* プレーやの手 */
    int computer;    /* コンピュータの手 */

    game_end = 0;    /* ゲーム終了フラグの初期化 */
    srand48(time(NULL));    /* 乱数列の初期化

}
```

現在時刻を種とした
疑似乱数列をつくる

コンピュータ実習（第3回）

練習課題その４

(1) コンピュータの手を作る(drand48()を使用)
(2) if文でコンピュータとプレーヤの勝敗を判定し，判定結果を表示する

```
else /* そうでなければ */{
   /* 乱数によりコンピュータの手を作る(1 or 2 or 3) */
   computer = (int)( 2.9999 * drand48() ) + 1;
/* 自分とコンピュータの手が同じ場合 */
if ( player == computer ){
  /* あいこの場合の処理 */
}
  /* 自分が勝った場合 */
else if((player==1&&computer==2) || (player==2&&computer==3) || (player==3&&computer==1)) {
 /* 勝ったの場合の処理 */
}
/* それ以外（自分が負けた）の場合 */
else{
/* 負けたの場合の処理 */
}
```

drand48()は0.0～1.0の間の乱数を生成する関数なので，0,1,2のいずれかの値となる

コンピュータ実習（第3回）

練習課題その４

じゃんけんゲームのプログラムを完成させてみてください.

じゃんけんゲームの流れ

1. ゲーム終了フラグに0を代入する .
2. ゲーム終了フラグが1でなければ次のループを繰り返す .
   (1) printf(...)を用いてメニューを表示する
   「1.グー 2.チョキ 3.パー 4.終了」
   (2) scanf(...)を用いて整数を入力する.
   (3) if文で入力された数字を調べる.
   4ならゲーム終了フラグに”1”を代入する.
   4でなければ以下を実行する.
   (1)コンピュータの手を作る(drand48()を使用)
   (2)if文でコンピュータとプレーヤーの勝敗を判定し,
   結果を表示する.
1. ゲーム終了メッセージを表示する.

コンピュータ実習（第3回）

練習課題その5

円周率の計算と計算精度
（Web教材　数値計算プログラムその2）

教材のマチンの公式によりπを求めよ.

この課題はこれまでの基礎部分のまとめです.

変数の宣言・型，ループ，条件判断，関数の定義

コンピュータ実習（第3回）

練習課題その5

円周率の計算と計算精度（Web教材　数値計算プログラムその2）

```
/* 課題 --- paiの数値計算プログラム */
#include <stdio.h>
#include <math.h>

/* 関数CalcATAN_Term()のプロトタイプ宣言 */
double CalcATAN_Term(int n, double x);

int main(void)
{
        int n;
        double PI;

        PI = 0.0;

    /* n=30 までΣを計算する
       n=1,2,…について、arctan()の項をPIに加算していく
     */
        for( n = 1; n <= 30; n++)
         {
              PI                                    PIの計算
                printf("PI[%d] = %1.111lf¥n", n, PI);
         }

        return 0;
}
```

コンピュータ実習（第3回）

練習課題その5

円周率の計算と計算精度（Web教材　数値計算プログラムその2）

```
/* arctan(x)のn番目の項を計算する関数 */
double CalcATAN_Term(int n, double x)
{
    double val;

    val = pow(x, (double)(2*n-1)) / (double)(2*n - 1);

    if ( (n%2==0) ) val = -val;

    return val;
}
```

コンピュータ実習（第4回）

ここから第4回の内容

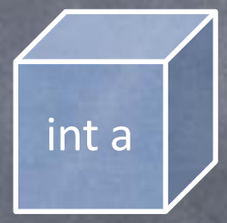
Web教材の本日のページの2番目の項目（配列、ポインタ）を開いてください．

コンピュータ実習（第4回）

配列（補助教材　配列，ポインタ）

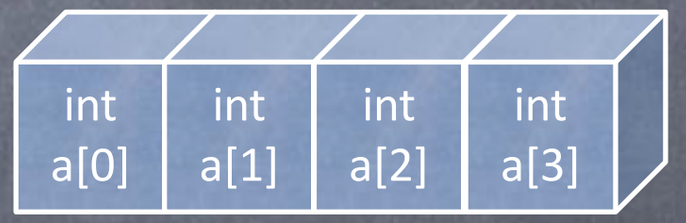
まずは変数の概念のおさらい．

変数：数値を入れておく入れ物



⇩

配列：複数の変数を集めて一列に並べた物



何か一連のデータを入れるのに200個の変数を使いたい．

int a0, a1, a2, a3‥‥a199；

- 一連のデータの何番目という指定がプログラム的に面倒
- そもそも宣言が面倒

そこで，配列変数　a[要素数]　を使う．

int a[200] これで200個（a[0]～a[199]）の変数を宣言できた．

また，何番目のデータを

int n; n=10; a[n];

で指定できる．

コンピュータ実習（第4回）

配列

さらに，行列のように

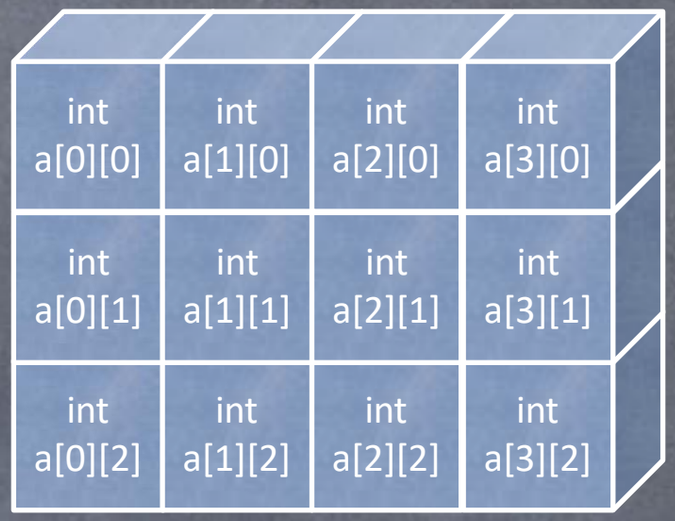
int a[200] [10]

a[0][0]・・・・a[199][0]

・・・・・・・・・・・・・・・・・・・・

a[9][0]・・・・a[199][9]

というのもできる．※これを2次元配列と言います．



注意点

- かならず[ ]の中の要素数は0から数える.
- 使用する要素数に注意を払う.
  - 特にcharの場合は入れたい文字の数+1の要素数を使用する必要がある.
  - 宣言した要素数以上を使うと，一見何もプログラムに影響を与えないように見えるが，いきなり暴走する事がある.

コンピュータ実習（第4回）

ポインタ（変数とアドレスの話）

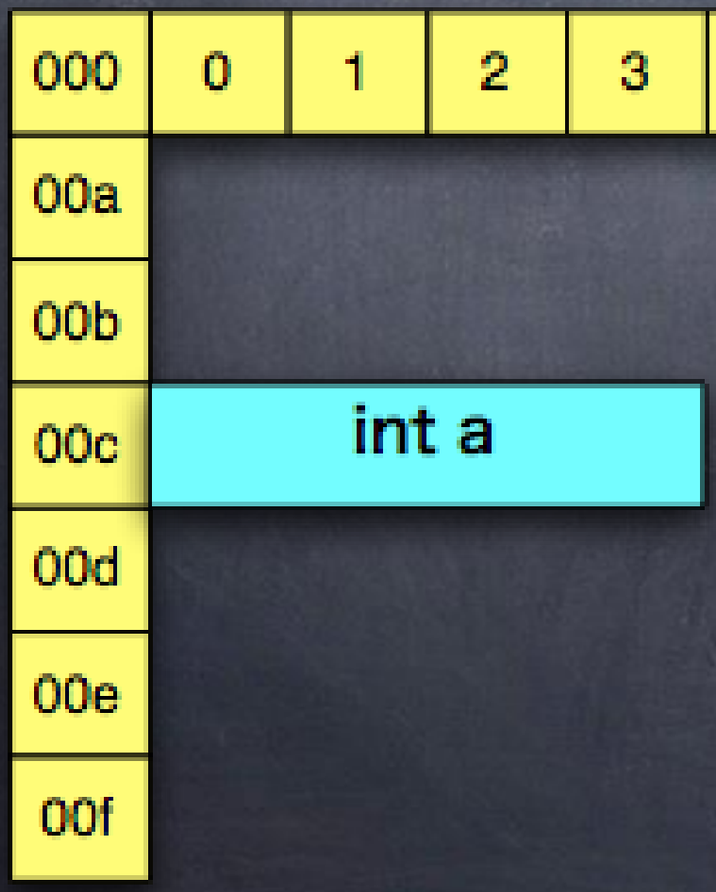
まずは変数の話をもう少し

変数の値はどこに記憶されているのか？

→コンピュータのメモリ上（アドレスで示される）

→変数の宣言→メモリの確保

→int変数一つにつき，4つのアドレス分（4バイト）使用，float変数は4バイト，double変数は8バイト使用する．



int a;　変数aの宣言

a=200;　変数aに値200を代入

コンピュータ実習（第4回）

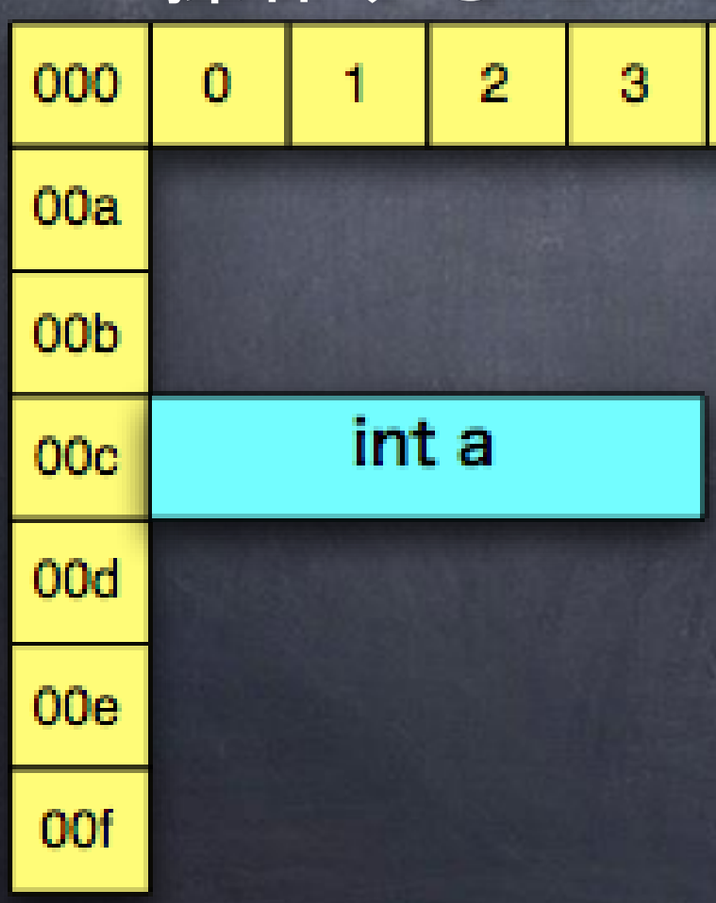
ポインタ（変数とアドレスの話）

ポインタとは，

変数の置かれたアドレスを指し示すもの

→アドレスは0から順にふってある整数で，ポインタも実体は整数値

→その数値を直接使うことなく，その数値の指し示した場所を操作するためのもの




```
int a;
int *p;   ポインタpの宣言
a=200;
p=&a;   ポインタpに変数aのアドレス00c0を代入
```

コンピュータ実習（第4回）

ポインタ（変数とアドレスの話）

ポインタは何のためにあるか？

- 関数の中で宣言した変数を別の関数で使いたい時に役立つ
  - 引数として大量の変数をコピーして渡す必要がなくなる
  - 複数の値を戻り値として返したい

- データの並べ替えなどの処理を行う時に役立つ
  - 指し示すもの（ポインタ）」だけを入れ替えることで，「指し示される対象（実データ）」は全く動かさなくて済む

- 大きさがよく分からないデータを扱う時に役立つ
  - 格納しておくための領域を無駄に大きく確保しておく必要がなくなる

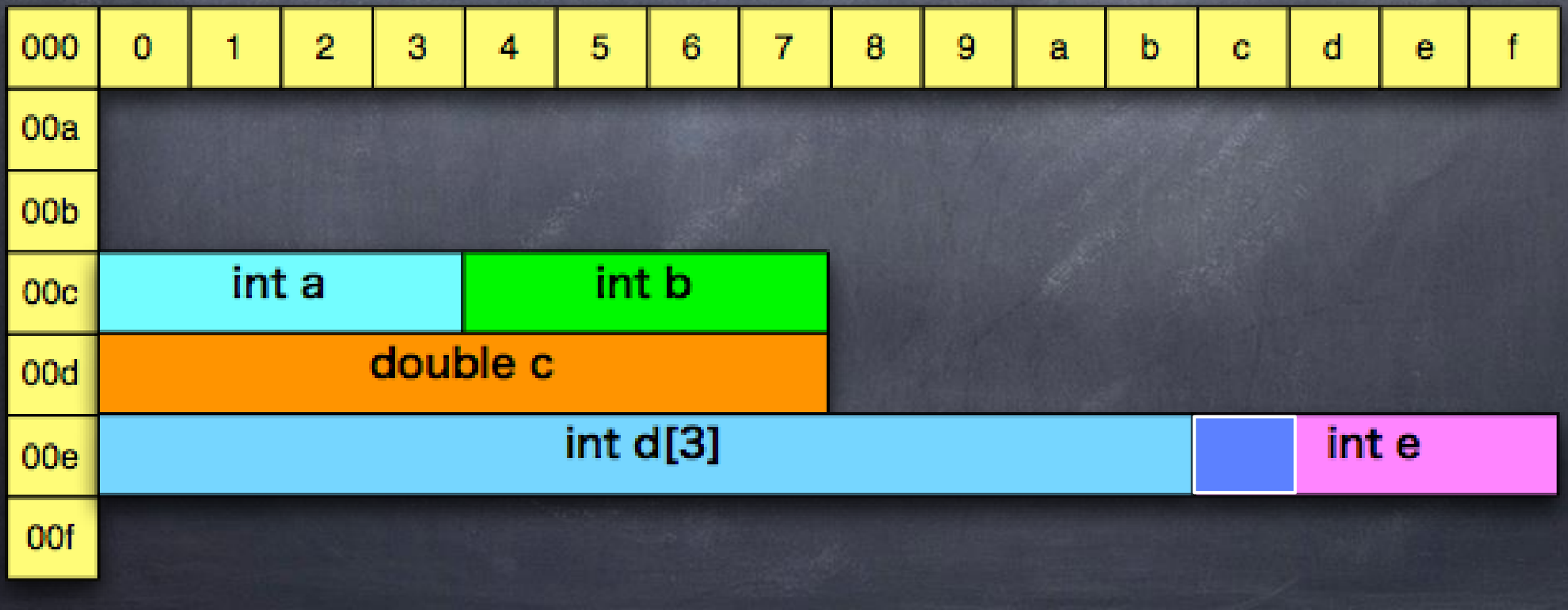
コンピュータ実習（第4回）

ポインタ（変数とアドレスの話）

さきほどの暴走の話： d[3]=5； でどうなるか？

int eのメモリを書き換えてしまう

予期しない動作→暴走



コンピュータ実習（第4回）

ポインタ（変数とアドレスの話）

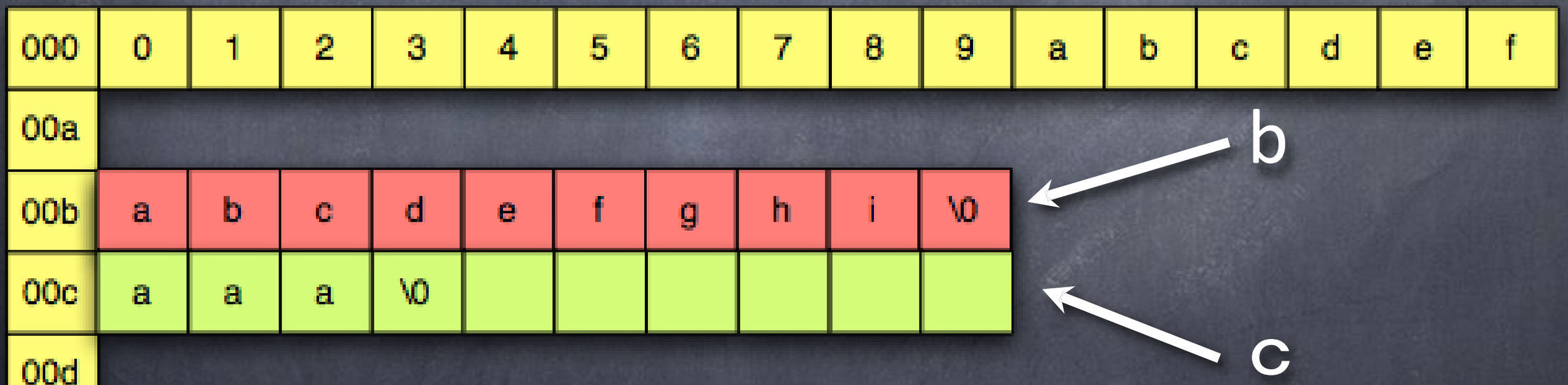
さらにまだ少し変数の話.

文字列のときはどうなるのか？

char b[10]; b=”abcdefghi”;

各アドレスに一文字づつ格納されていく

char c[10]; c=”aaa”;



しかし要素数以下の場合にはメモリに無駄が出る

コンピュータ実習（第4回）

ポインタ（変数とアドレスの話）

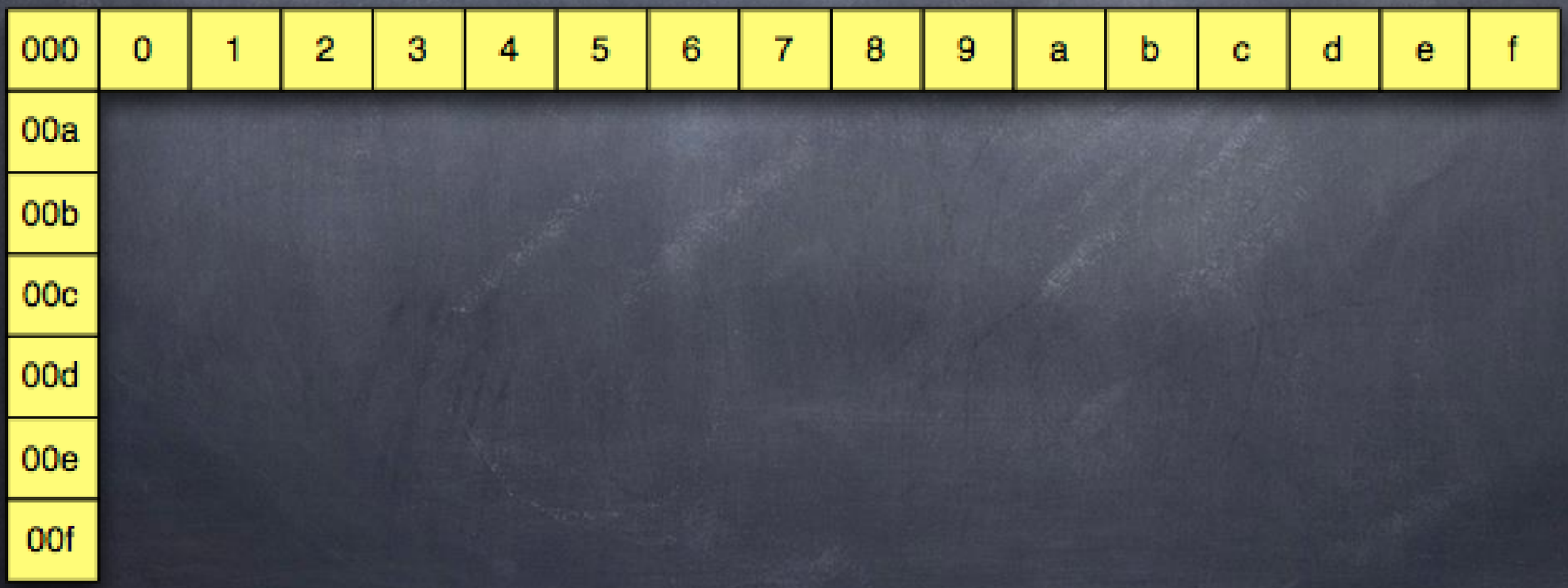
また，みかけは他の配列変数だが

```
char b[10]; b[1]=”123”;  b[2]=”456”;
```

などという事はできない．

```
char b; b=”123”
```

もできない．



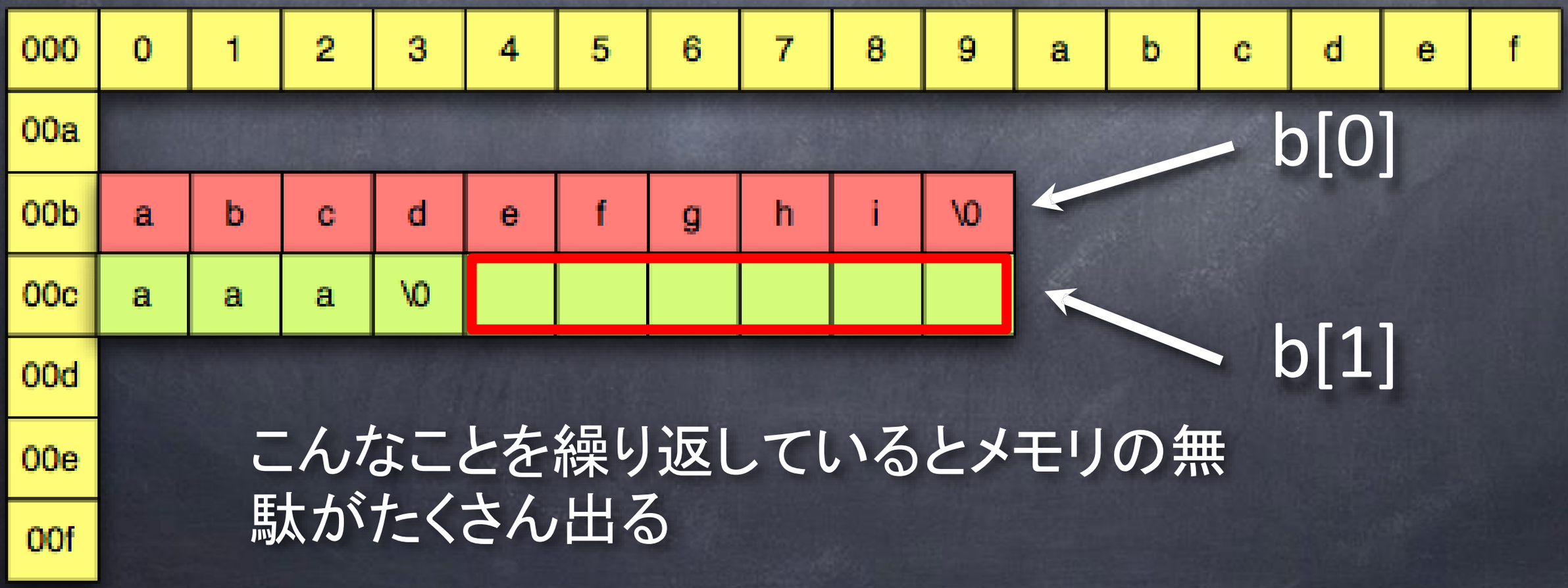
コンピュータ実習（第4回）

ポインタ

ではどうするのか

```
char b[10][2];
b[0]=”abcdefghij”; b[1]=”aaa”
```

とする手がある（二次元配列）.



こんなことを繰り返しているとメモリの無駄がたくさん出る

コンピュータ実習（第4回）

ポインタ

話は戻って，ポインタの話をもう一度

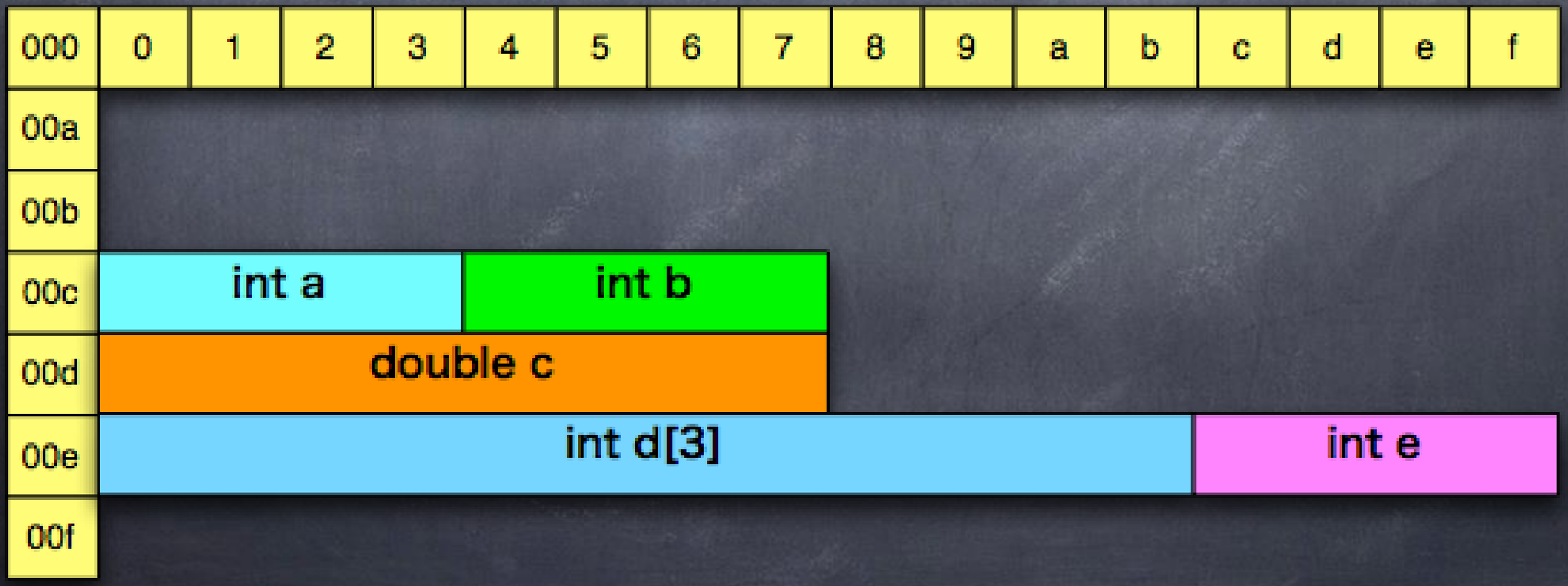
　ポインタとは，

ある変数の内容が書き込まれた先頭アドレス

　を内容とする変数.

例えば，

　aの内容が書き込まれている先頭アドレス「00c0」が内容



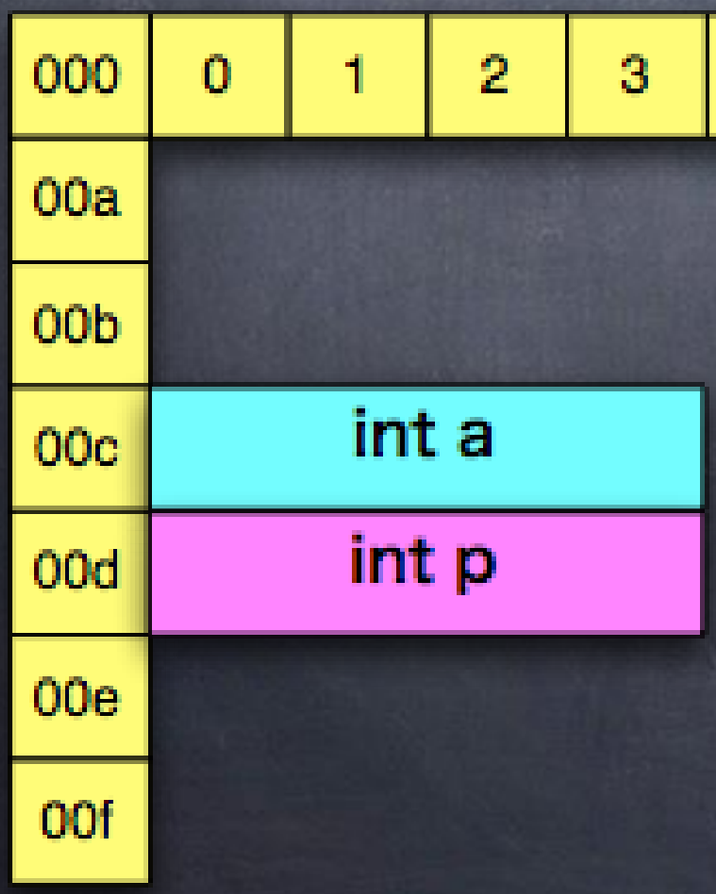
コンピュータ実習（第4回）

ポインタ

宣言は

アドレスを記憶しようとする変数の型　＊ポインタ名

int ＊p



int型の変数のポインタを作りたい

コンピュータ実習（第4回）

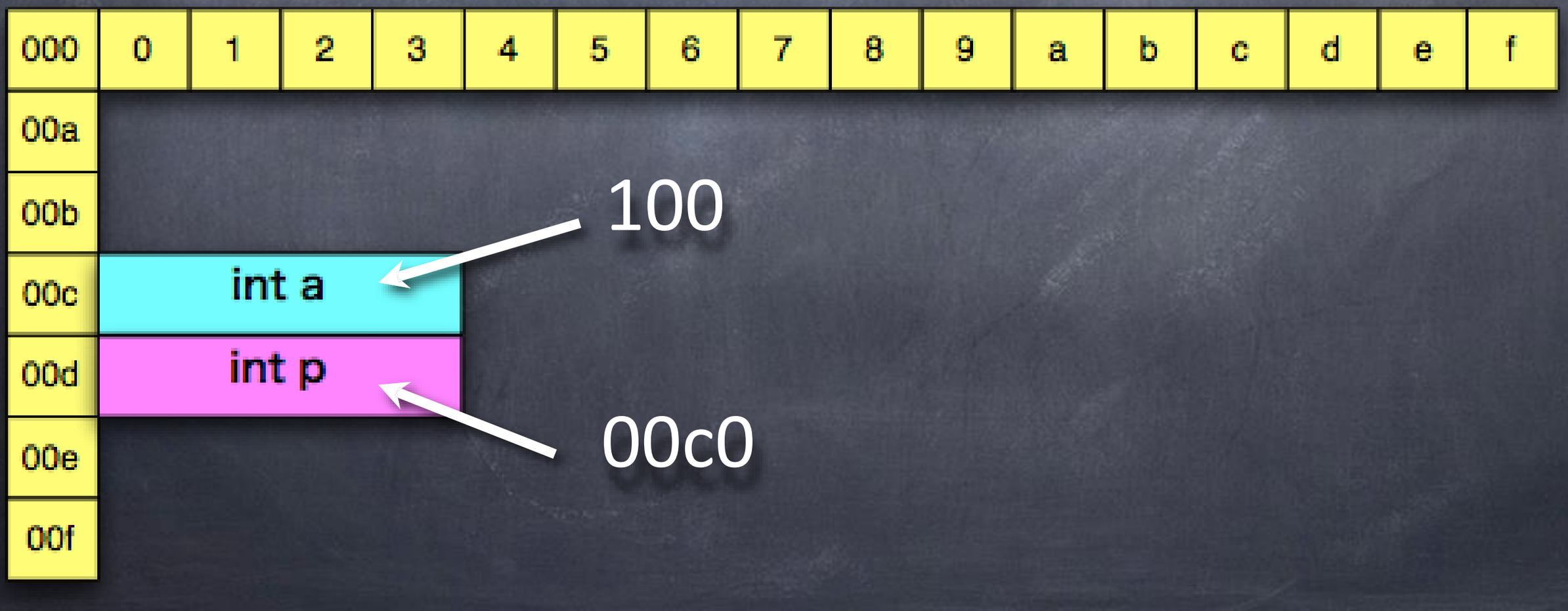
ポインタ

ポインタの使い方
まずポインタの値（アドレス）を設定

p=&a; （＊は付かない）

次にポインタの示すアドレス内容（データ）にアクセス

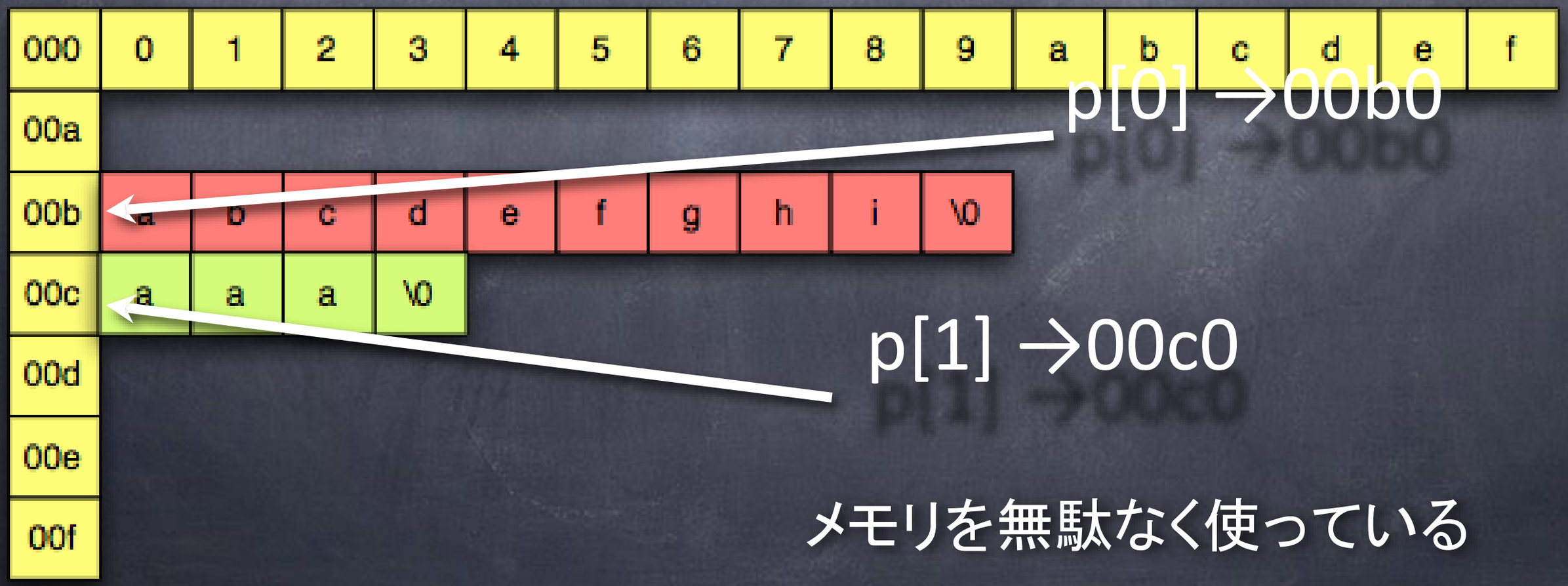
＊p=100;



コンピュータ実習（第4回）

ポインタ

そこで，さきほどのメモリの無駄の話にもどる
ポインタを使用した文字列の格納では，文字数にあわせてメモリに配置していく．そこで，
char *p[2] = {"abcdefghij","aaa"};
というようなやり方をよく使う．

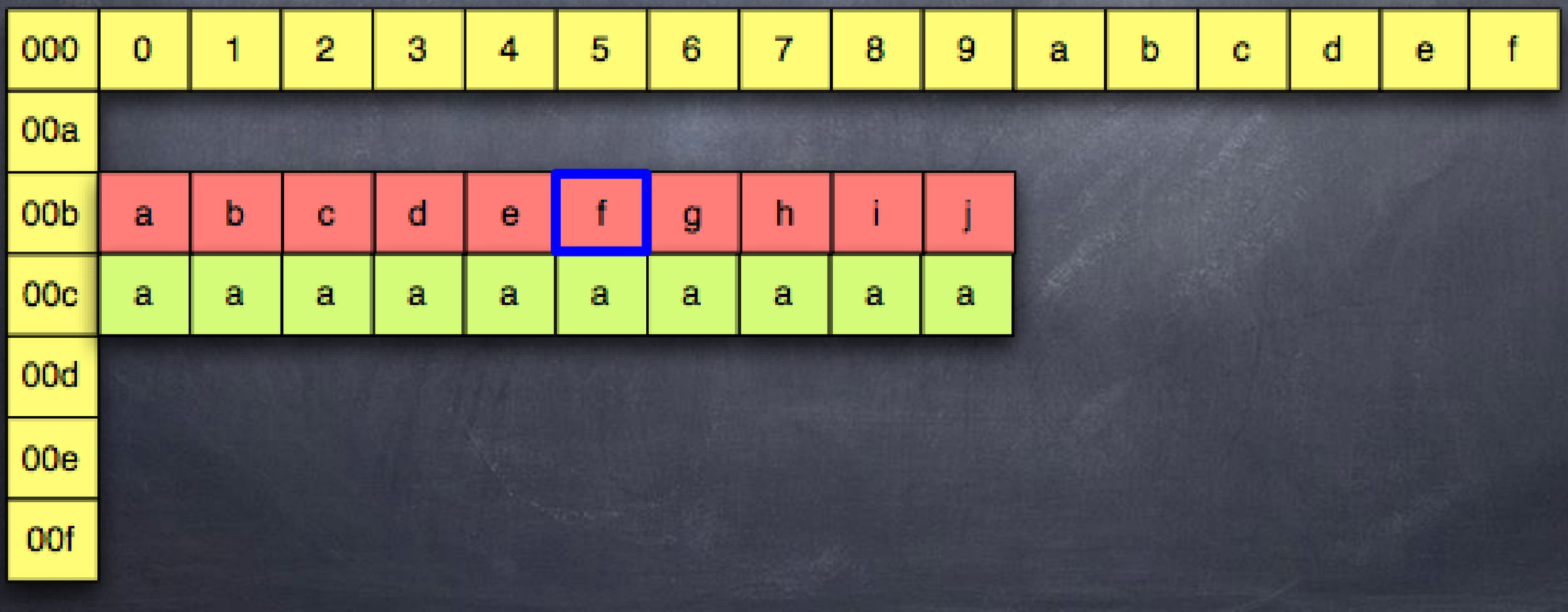


メモリを無駄なく使っている

コンピュータ実習（第4回）

ポインタ

また，文字列を扱う上で一番よく出てくる問題
　先頭からn番目の文字を抜き出したい
などの一つの答えもポインタ.
ポインタをポインタで示すこともできる.



コンピュータ実習（第4回）

ポインタまとめ

- 「変数がしまってある場所」を指し示す
- データがおかれているメモリの場所(番地)を指定する
- データのやりとり(変数にデータをいれるなど)に役立つ．特に数が多いとき．

このポインタというものは理解すれば便利なものなのですが，慣れるまでは時間がかかります．ですから，じっくりと時間をかけて理解してください．ポインタはC言語を極める第一歩です．

最低限、文字列を扱うときにchar *str を使うことと、scanf 等で文字列を読み込む場合にはchar str[200]; のような配列を使うことを覚えておいて下さい。

コンピュータ実習（第4回）

構造体（構造体と入出力プログラム）

構造体の概念

構造体とは，複数の型の変数をひとまとめにしたもの

なぜ必要か

- 違う型だけど関係のあるデータだからまとめておきたい
- 機能ごとに分けてわかりやすくしたい

コンピュータ実習（第4回）

構造体（構造体と入出力プログラム）

構造体の宣言

```
struct info{
   char name[256];
   int age;
};
```

構造体infoを宣言

```
typedef struct info Info;
```

構造体テンプレート名　別名

これで　info型という型を自分で定義した.
この型で定義された変数は　name　という文字配列変数と　age　という整数変数をひとまとめにして扱える.
構造体によってまとめた要素ひとつひとつのことを「メンバ」といいます.

コンピュータ実習（第4回）

構造体（構造体と入出力プログラム）

使い方の基本型：構造体メンバへのアクセス

構造体変数のメンバを参照するには.「.（ピリオド）」を使って，どのメンバを参照するのか指定します．

```
Info information;
information.age = 18;
information.name = hogehoge;
```

構造体変数名 ↵　　↳ メンバ名

構造体配列：構造体変数についても配列を考えることができる

```
Info information [10];
int n; n=1;
information[n].age = 18;
information[n].name = hogehoge;
```

コンピュータ実習（第4回）

「構造体、入出力プログラムによる演習」の
練習と課題1・2・3 に取り組んでください

コンピュータ実習（第4回）

これからのカーナビ作成にむけて