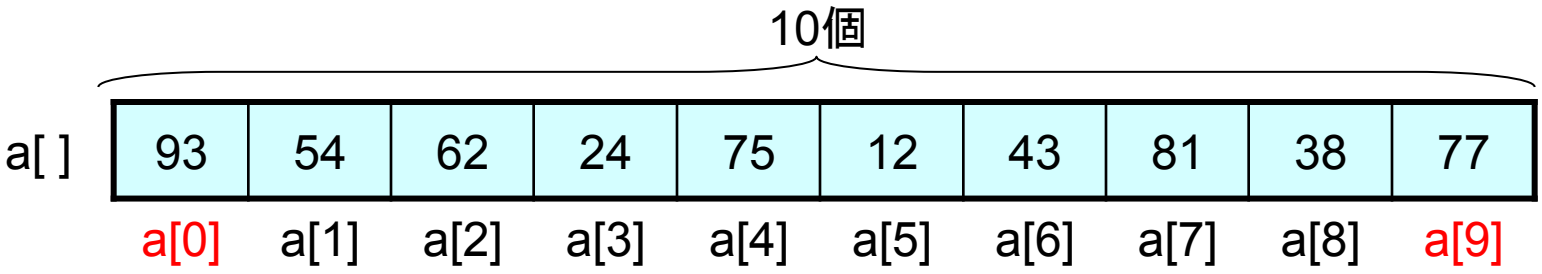
Programming II 0x04

配列と関数（2011.05.12）

塩澤秀和 http://vilab.org

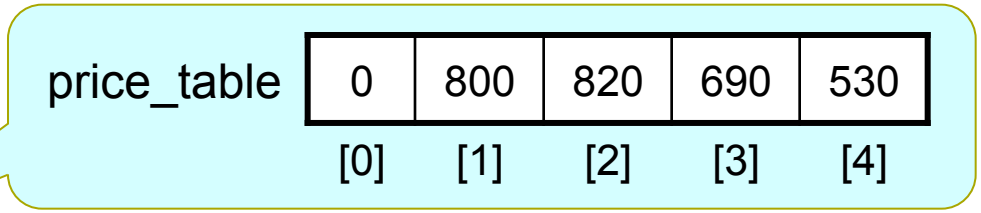
配列の作りかた（復習）

- 配列の定義
  - 定義（宣言）： 要素の型 配列名 [要素数]; （p.101）
    int a[10]; ← a[0] ～ a[9] の10個の変数をまとめて作る
  - 初期化（p.104）
    int a[10] = { 93, 54, 62, 24, 75, 12, 43, 81, 38, 77 };



- 配列の要素
  - 配列の要素（a[0]など）は、単独の変数と同じように使える
  - [ ] 内の添字（そえじ index）は、**0から始まる番号**（整数）
  - **【注意】** int a[10] と定義した配列には a[10] は**ない!!**

「番号→データ」の表（復習）




```
#include <stdio.h>


// 価格表（グローバル変数）
int price_table[] = {
  0, 800, 820, 690, 530 };

int main(void)
{
  int order, price;

  printf("定食メニュー\n");
  printf("1. 生姜焼き\n");
  printf("2. とんかつ\n");
  printf("3. さんま\n");
  printf("4. お子様ランチ\n");
  printf("定食の番号？ ");
  scanf("%d", &order);
  if (order < 1 || 4 < order) {
    printf("入力エラーです\n");
    return 1;  // プログラム終了
  }

  // 価格表で「番号→価格」の変換
  price = price_table[order];

  printf("%d円です\n", price);
  return 0;
}
```

2つの配列の和(復習)

```
// 2つのN次元ベクトルの和
#include <stdio.h>
// マクロ定数 N = 3 の定義
#define N 3

int main(void)
{
  // 同じ大きさの配列を3つ作る
  double v1[N], v2[N], v3[N];
  int i;

  // v1, v2をキーボードから読み込む
  for (i = 0; i < N; i++){
    printf("v1[%d] = ", i);
    scanf("%lf", &v1[i]);
  }
  for (i = 0; i < N; i++){
    printf("v2[%d] = ", i);
    scanf("%lf", &v2[i]);
  }

  // 要素ごとに和を計算する
  for (i = 0; i < N; i++){
    v3[i] = v1[i] + v2[i];
  }

  // 結果v3を表示する
  for (i = 0; i < N; i++){
    printf("v3[%d] = %f\n",
           i, v3[i]);
  }
  return 0;
}
```

配列のサイズを変えたいときにはここだけ変えればいい

関数の引数（復習）

- 関数にデータを渡す方法
  - 関数呼び出しの () のなかに、データ（式）を並べる
    関数名 (式) ; ← 2つ以上なら「関数名(式, 式, …);」
  - 関数に受け渡しするデータを“引数（ひきすう）”という

- 引数をとる関数の定義
  - 関数定義の () のなかに、データを受け取る変数を定義

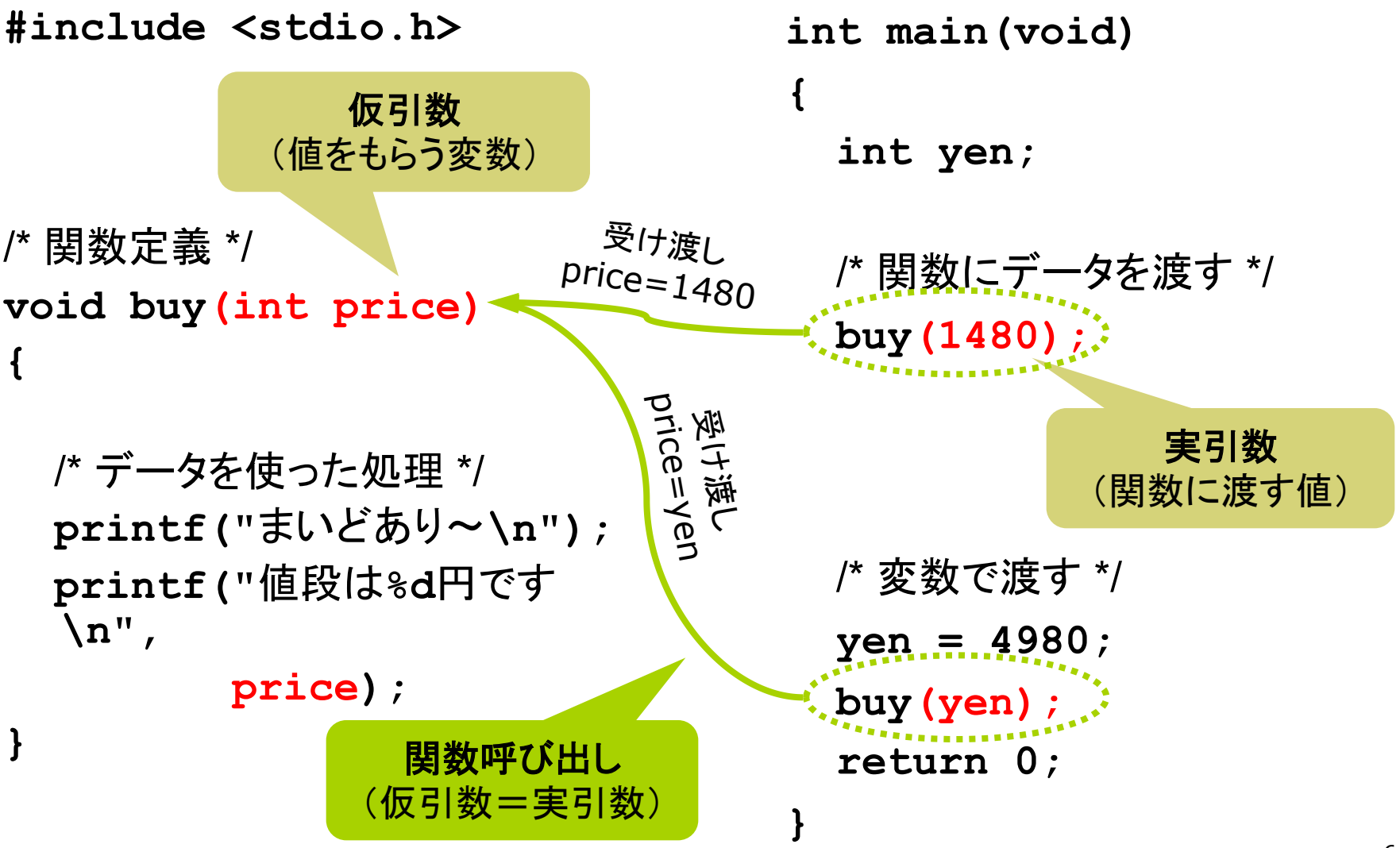
```
void 関数名 (データ型 変数, …)
{
    変数を使ったプログラム
}
```

double x, int a
などいくつでも

※ voidは空（なし）という意味

  - 引数の変数は関数内でのみ通用する（ローカル変数）

関数にデータを渡す（復習）

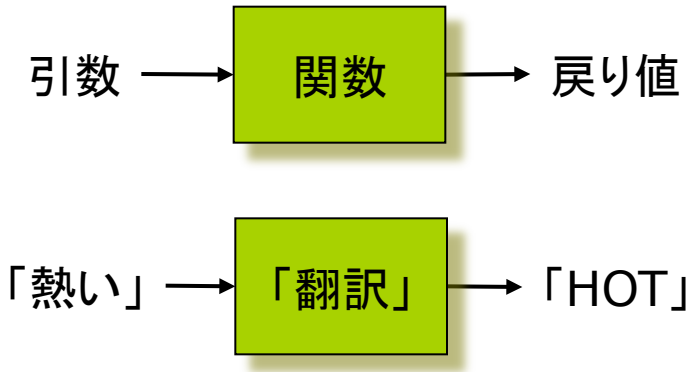


関数の戻り値(復習)

- 関数からデータを渡す方法
  - 計算結果などの値を、関数から呼び出し元(main)に渡す
  - “引数”とは逆方向 ＝ “戻り値”、“返り値”、“返却値”

```
戻り値の型　関数名 (引数並び)
{
    計算などの手順
    ...
    return　戻り値;
}
```

戻り値の式



関数の概念

- return文(p.76)
  - 関数を終了して、呼び出し元にすぐ戻る
  - 戻り値がある関数の場合、値を“戻す”ために絶対に必要

引数と戻り値（復習）

```
#include <stdio.h>

int plus(int x, int y)
{
   int z;

   z = x + y;

   return z;
}

int main(void)
{
   int a, b, c;

   printf("a b -> ");
   scanf("%d %d", &a, &b);

   c = plus (a, b);

   printf("c = %d\n", c);

   return 0;
}
```

戻り値のデータ型

引数（関数に渡す値）

戻り値（関数から返す値）

関数に配列を渡す

```
/* リスト5-17を一部変更(p.108) */
#include <stdio.h>
int gokei(int b[5]);

int main(void)
{
  int a[5], sum;
  int i;

  for(i = 0; i < 5; i++){
    printf("a[%d]=", i);
    scanf("%d", &a[i]);
  }

  sum = gokei( a );
  printf("sum=%d\n", sum);
  return 0;
}

/* 総和を計算する関数 */
int gokei(int b[5])
{
  int i;
  int sum = 0;

  for(i = 0; i < 5; i++) {
    sum += b[i];
  }
  return sum;
}
```

実引数では配列名だけを書く

仮引数では同じ型の配列変数を宣言して受け取る

関数で配列を書き換える

```
/* 関数内で配列の全要素を2倍する */
/* リスト5-18 (p.110) を一部変更    */
#include <stdio.h>
/* 定数マクロの利用例 */
#define N 3

void multi2(int b[N]);

int main(void)
{
  int i, a[N];

  for(i = 0; i < N; i++) {
    printf("a[%d] = ", i);
    scanf("%d", &a[i]);
  }

  multi2( a );

  for(i = 0; i < N; i++) {
    printf("a[%d] = %d\n",
           i, a[i]);
  }
  return 0;
}
```

aの内容が変わっていることを確認

```
/* 配列の全要素を2倍 */
void multi2(int b [N])
{
  int i;

  for(i = 0; i < N; i++) {
    b[i] = b[i] * 2;
  }
}
```

仮引数

実引数

関数内で配列の中身を書き換えられる！

引数渡しのしくみ

- 普通の引数では(p.83)
  - 関数内で変数(仮引数)の値を変更しても、呼び出し元の変数(実引数)は変わらない
  - 右の例でplus10の中でxを変えても、mainのaには影響しない
  - 理由:引数は値がコピーされるから(「値渡し」call by value)
- でも、配列引数は例外
  - 配列の場合は、値のコピーではなく、**“場所”**が渡される
  - 関数の中で、呼び出し元の配列の要素を変更できてしまう
  - 「参照渡し」call by reference

```
#include <stdio.h>

void plus10(int x)
{
  /* 関数内で変数の値を変えても */
  x = x + 10;
}

int main(void)
{
  int a;

  scanf("%d", &a);
  plus10(a);
  /* 元の変数の値は変わらない */
  printf("%d\n", a);
  return 0;
}
```

C言語の文字列

- 文字列は文字（char）の配列（p.114）
  - 先頭の文字から1文字ずつ配列に入れて、最後の文字の次に、ヌル文字（'\0'＝文字コード0の文字）をつける
  - char str[6] = { 'H', 'e', 'l', 'l', 'o', '\0' };　← 5文字＋'\0'
  - 最低限、文字列の長さ＋1文字分の長さの配列が必要

- 文字列の初期化（p.115）
  - char str[6] = "Hello";
  - char str[ ] = "Hello";
  - char str[10] = "Hello";

H	e	l	l	o	\0

H	e	l	l	o	\0	空	空	空	空

  - 「= "文字列"」で初期化できるのは、変数定義のときだけ
    - 初期化以外では標準ライブラリ関数の strcpy でコピーする

文字列の関数

- 文字列の表示（p.116）
  - 例：printf("名前は%sです\n", str);
  - 書式：「%-10s」とすれば、10桁の出力欄に左詰で表示

文字配列

- 文字列の入力（p.117）
  - 例：char buf[100]; scanf("%s", buf);
  - 入力文字列がはみ出ない、十分な長さの配列を用意する
  - 安全策：「%99s」と書けば、入力を最大99文字に制限

&は不要

- 文字列のコピー（p.119）
  - 関数 strcpy を使う（#include <string.h> が必要）
  - strcpy(コピー先配列, コピー元文字列);

文字列の使用例

```
/* 文字列の初期化と表示 */
#include <stdio.h>

int main(void)
{
  char s1[5] = {
    'S','o','f','t','\0' };
  char s2[8] = "Machida";
  char s3[] = "Tamagawagakuen";
  char s4[] = "Yokohama";

  printf("s1:%s\n", s1);
  printf("s2:%10s\n", s2);
  printf("s3:%10s\n", s3);
  printf("s4:%-10s\n", s4);
  return 0;
}
```

これがないと誤動作

```
/* 文字列の入力とコピー */
#include <stdio.h>
#include <string.h>

int main(void)
{
  char buf[20], str[20];

  printf("お名前は？ ");
  scanf("%s", buf);

  strcpy(str, buf);
  printf("%sさん\n", str);
  return 0;
}
```

十分なサイズ

注意：%s では、スペースを含む文字列は入力できない（この問題の対策は難しい）

文字列の構造

フローチャートを書いてみよう

```
/* 文字列の中の1文字ずつを表示 */
#include <stdio.h>

int main(void)
{
  int i;
  char str[100];

  printf("文字列 -> ");
  scanf("%s", str);

  i = 0;
  while (str[i] != '\0') {
    printf("文字%c コード%d\n",
           str[i], str[i]);
    i++;
  }
  return 0;
}
```

文字列の終端でない間、継続

演習問題

4a. 要素数が10個の整数配列を引数として受け取り、その中のすべての数値を順に表示する関数を作成しなさい。

- この関数の動作を確認するmain関数をつけて実行させること。

4b. 配列 int a[2] を仮引数として受け取り、a[0] と a[1] の値を交換する関数を作成しなさい。つまり、この関数に渡す前に中身が { 1, 2 } だった配列は、{ 2, 1 } に変化する。

4c. 標準入力から4つの文字列s1, s2, s3, s4を読み込み、2行2列の表形式に配置して表示するプログラムを作成しなさい。

- 文字列は英数字で10文字以内としてよい。

4d. キーボードから文字列を読み込み、その長さ（文字数）を調べて表示するプログラムを作成しなさい。

- 文字列の終わりは、ヌル文字（文字コード＝'\0'）で判断すること。