情報基礎 B
第 14 回 アカデミック・スキル II
C 言語プログラミング (5)

長江 剛志

(nagae@m.tohoku.ac.jp)

東北大学大学院工学研究科
技術社会システム専攻

2015 年 7 月 110 日 (金)

今日やること

今日やること

配列を用いた複数のデータの処理

レポート課題 III-4: 素数と合成数に分類

目次

ディレクトリとサンプル・コードの準備 (1)

ディレクトリの準備

Terminal 上から mkdir コマンドを使って
~/Documents/prog/04_array_func というディレクトリを作る.

サンプル・コードのダウンロード

- ISTU (http://www.istu.jp) にアクセスし, 右下の 受講授業科目 から, 金曜の 3 時限 情報基礎 B を選択
- 科目共通教材 の中から 配布資料 第 14 回 C 言語サンプルプログラム (4) を選択

ディレクトリとサンプル・コードの準備 (2)

サンプル・コードの移動

- Home (メニューバーの「場所」でもよい) を開いて「ダウンロード」内にある上記のファイルを ~/Documents/prog/04_array へ移動.

現在の作業ディレクトリの変更

Terminal 上の cd コマンドを使って，現在の作業ディレクトリを ~/Documents/prog/04_array に変更する.

目次

配列にデータを格納・表示する (nonarray_reverse.c, array_reverse.c) (1)

やりたいこと

4 個 の整数を入力した後，それらを 入力されたのと逆の順 に表示させる.

配列を使わないサンプル・コード (nonarray_reverse.c)

```c
#include <stdio.h>                  /* 標準入出力ライブラリ */
int main(void)                      /* main 関数の引数と戻り値の定義 */
{                                   /* main 関数の始まり */
  int a0, a1, a2, a3;               /* 整数型の変数 a0〜a3 を宣言 */

  /* 4つの整数を読込む */
  printf("4つの整数? ");
  scanf("%d %d %d %d", &a0, &a1, &a2, &a3); /* 4つの整数を a0, a1, a2, a3 の順に読込む */

  /* 入力されたのと逆の順に表示 */
  printf("%d %d %d %d\n", a3, a2, a1, a0); /* a3, a2, a1, a0 の順に表示 */

  return 0;                         /* 戻り値として 0 を返す */
}                                   /* main 関数の終わり */
```

配列にデータを格納・表示する (nonarray_reverse.c, array_reverse.c) (2)

配列を使ったサンプル・コード (array_reverse.c)

```c
#include <stdio.h>                  /* 標準入出力ライブラリ */
int main(void)                      /* main 関数の引数と戻り値の定義 */
{                                   /* main 関数の始まり */
  int N = 4;                        /* 入力する整数の数(この場合4)を N に格納 */
  int a[N];                         /* N個の要素を持つ整数型配列 a を定義 */

  int i;                            /* 要素番号を格納するための変数 */

  /* N個の整数を読込む */
  printf("%d個の整数? ", N);
  for (i = 0; i < N; ++i)           /* iを0からN-1まで1つづつ増やしながら繰り返し */
    { scanf("%d", &a[i]); }         /* 配列aのi番目要素 a[i] に値を読込む*/

  /* 入力されたのと逆の順に表示 */
  for (i = N-1; i >=0; --i)         /* iをN-1から0まで1つづ減らしながら繰り返し */
    { printf("%d ", a[i]); }       /* 配列aのi番目要素 a[i] を出力する */
  printf("\n");                     /* 最後に改行を出力 */

  return 0;                         /* 戻り値として 0 を返す */
}                                   /* main 関数の終わり */
```

nonarray_reverse.c のコンパイルと実行

1. gcc コマンドを用いてソースファイルから 実行ファイル を生成する．-o オプションを用いて，実行ファイル名を nonarray_reverse とする．

```
$ gcc -o nonarray_reverse nonarray_reverse.c ↩
```

2. 生成された実行ファイル (./nonarray_reverse) を呼び出す．

```
$ ./nonarray_reverse ↩
4つの整数? 7 3 1 4          # 入力待ち状態になるので4つの数を適当に
4 1 3 7
```

3. array_reverse.c も同様にコンパイル・実行してみよう

array_reverse.c の解説 (1)

配列 は複数のデータをまとめて扱うのに必須の機能.

- nonarray_reverse.c では a0, a1, a2, a3 という 4 個の変数を個別に宣言し, それらに値を順に代入した後, a3, a2, a1, a0 の順に出力した. この方法だと, 入力する値が増減した場合,
  - 変数の宣言を行なう行 (第 4 行)

```
int a0, a1, a2, a3;             /* 整数型の変数 a0〜a3 を宣言 */
```

  - キーボードから入力された値を変数に取り込む行 (第 8 行)

```
scanf("%d %d %d %d", &a0, &a1, &a2, &a3); /* 4つの整数を a0, a1, a2, a3 の順に読
```

  - 出力する行 (第 11 行)

```
printf("%d %d %d %d\n", a3, a2, a1, a0); /* a3, a2, a1, a0 の順に表示 */
```

  を全て変更する必要がある.

array_reverse.c の解説 (2)

- 一方, array_reverse.c では, まず, a[0], a[1], a[2], a[3] という N=4 個の要素を持つ配列 a を宣言し, それぞれの要素に値を順に代入した後, a[3], a[2], a[1], a[0] の順に出力している. 具体的には,
  - 第 5 行目 (配列の宣言):

```
  int a[N];                          /* N個の要素を持つ整数型配列 a を定義 */

  int i;                             /* 要素番号を格納するための変数 */

  /* N個の整数を読込む */
  printf("%d個の整数? ", N);
  for (i = 0; i < N; ++i)            /* iを0からN-1まで1つづつ増やしながら繰り返し */
    { scanf("%d", &a[i]); }          /* 配列aのi番目要素 a[i] に値を読込む*/

  /* 入力されたのと逆の順に表示 */
  for (i = N-1; i >=0; --i)          /* iをN-1から0まで1つづ減らしながら繰り返し */
    { printf("%d ", a[i]); }         /* 配列aのi番目要素 a[i] を出力する */
  printf("\n");                      /* 最後に改行を出力 */

  return 0;                          /* 戻り値として 0 を返す */
}                                    /* main 関数の終わり */
```

N 個の要素を持つ整数 (int) 型の配列 a を宣言している. 配列は以下の書式で宣言する:

array_reverse.c の解説 (3)

変数の型　配列名 [配列の要素数];

配列の各要素に値を代入したり，値を参照したりするには，a[0]，a[1]，... のように，

配列名 [配列の要素番号]

とすることで，通常の変数と同様にアクセスできる．

配列の要素番号は 0 から始まる．つまり，N 個の要素を持つ配列 a の先頭の要素は a[0]，最後の要素は a[N-1] で表される．

- 第 11〜12 行目 (キーボードから入力された値を変数に取り込む):

```
for (i = 0; i < N; ++i)          /* iを0からN-1まで1つづつ増やしながら繰り返し */
  { scanf("%d", &a[i]); }        /* 配列aのi番目要素 a[i] に値を読込む */
```

for 文を使って，i の値を 0 から N-1 まで 1 づつ増やしながら配列の要素 a[i] にキーボードから入力された値を読み込んでいる．このように「配列の各要素 に対して 同じ処理 を繰り返す」ことが簡単に実装できるのが配列のメリットである．

array_reverse.c の解説 (4)

- 第 15〜16 行目 (入力されたのと逆順に出力):

```
  for (i = N-1; i >=0; --i)      /* iをN-1から0まで1つづ減らしながら繰り返し */
    { printf("%d ", a[i]); }     /* 配列aのi番目要素 a[i] を出力する */
```

for 文を使って, i の値を N-1 から 0 まで 1 づつ減らしながら配列の要素 a[i] を出力している.

配列に任意の個数のデータを格納する (array_reverse_v2.c) (1)

やりたいこと

array_reverse.c を改良して, 10 個以下の任意の個数 の正の整数を入力した後, それらを 入力されたのと逆の順 に表示させる. 負の値 が入力された場合, 10 個未満でも入力を打ち切り, そこまで入力された整数を逆順に表示させる.

サンプル・コード (array_reverser_v2.c)

配列に任意の個数のデータを格納する (array_reverse_v2.c) (2)

```c
#include <stdio.h>                  /* 標準入出力ライブラリ */
int main(void)                      /* main 関数の引数と戻り値の定義 */
{                                   /* main 関数の始まり */
  int N_max = 10;                   /* 入力できる数の上限(10)を N_max に可能 */
  int a[N_max];                     /* N_max個の要素を持つ整数型配列 a を定義 */
  int N;                            /* 入力された整数の個数を格納する変数 */

  int i;                            /* 要素番号を格納するための変数 */

  /* -1が入力されるかN個の整数が入力されるまで繰り返す */
  for (i = 0; i < N_max; ++i)  /* iを0からN_max-1まで1つづつ増やしながら繰り返し */
    {
      scanf("%d", &a[i]);           /* 配列のi番目要素 a[i] に値を読込む */
      if (a[i] < 0)                 /* 入力値が負なら break でループを抜ける */
        { break; }
    }
  /* ループを抜けた時点で i には「入力された整数の個数」が格納されている */
  N = i;                            /* 入力された整数の個数を格納 */

  /* 入力されたのと逆の順に表示 */
  for (i = N-1; i >=0; --i)         /* iをN-1から0まで1つづ減らしながらnum 期化でiの値を1つ減らし
    { printf("%d ", a[i]); }       /* 配列aのi番目要素 a[i] を出力する */
  printf("\n");                     /* 最後に改行を出力 */

  return 0;                         /* 戻り値として 0 を返す */
}                                   /* main 関数の終わり */
```

array_reverse_v2.c のコンパイルと実行

1. gcc コマンドを用いてソースファイルから 実行ファイル を生成する．-o オプションを用いて，実行ファイル名を array_reverse_v2 とする．

```
$ gcc -o array_reverse_v2 array_reverse_v2.c ↩
```

2. 生成された実行ファイル (./array_reverse_v2) を呼び出す．

```
$ ./array_reverse_v2 ↩
7 3 1 4 5 -1          # いくつかの数の後 -1 を入れて ↩
5 4 1 3 7
1 2 3 4 5 6 7 8 9 10 11 # 10個以上の値を入れた場合は -1 は不要
10 9 8 7 6 5 4 3 2 1
```

array_reverse_v2.c の解説 (1)

- 4〜6 行目

```
int N_max = 10;                 /* 入力できる数の上限(10)を N_max に可能 */
int a[N_max];                   /* N_max個の要素を持つ整数型配列 a を定義 */
int N;                          /* 入力された整数の個数を格納する変数 */
```

実際に入力される整数の個数 が 実行されるまで判らない ので, 入力できる整数の個数の上限を N_max に格納しておき, max_N 個の要素を持つ配列 a を宣言.
実際に入力された整数の個数 は, 上限 N_max とは 別の整数型変数 N (第 6 行目で宣言) に格納する.

- 11〜16 行目

```
for (i = 0; i < N_max; ++i)   /* iを0からN_max-1まで1つづつ増やしながら繰り返し */
  {
    scanf("%d", &a[i]);         /* 配列のi番目要素 a[i] に値を読込む */
    if (a[i] < 0)               /* 入力値が負なら break でループを抜ける */
      { break; }
  }
```

array_reverse_v2.c の解説 (2)

for 文を使って i を 0 から N_max-1 まで 1 づつ増やしながらキーボードから入力された値を a[i] に格納している.
array_reverse.c とは違い, 入力された値が負だった場合には, break を使って for ループから抜け出している. これによって, N_max より少ない個数でも入力を打ち切ることができる.
なお, break によって for ループが **中断** された場合でも, i===N_max が満足されて for ループが **終了** した場合でも, i には, それまで入力された整数の個数 が格納されていることに注意せよ.

- 18 行目:

```
N = i;                          /* 入力された整数の個数を格納 */
```

array_reverse_v2.c の解説 (3)

上記の for ループ終了後に変数 i に格納された「入力された整数の個数」を変数 N に保存している. これにより, 入力された最後の整数を格納した配列の要素に a[N-1] でアクセス可能となる.

目次

レポート課題 III-4 (1)

レポート課題 III-4 (素数と合成数に分類)

「入力された 15 個以下の任意の個数 の正の整数を入力した後，それらを 素数 とそれ以外 (合成数) とに分類し，それぞれを 入力されたのと逆の順 に表示させる」プログラムを作り，その ソースファイル と，以下の 9 個の整数:

```
83 32 21 19 97 76 63 3 35 51
```

に対する結果を提出せよ．ただし，下記を満足すること．

- 提出するファイル名は `B5TB9999_prime_reverse.c` (ソースファイル) および `B5TB9999_prime_reverse.txt` (実行結果) とせよ．
- ソースファイルには適宜 コメント を記入せよ．

(続く)

レポート課題 III-4 (2)

レポート課題 III-4 の仕様 (続き)

- 15 個未満 で入力を打ち切る場合は 負の値 を入力する.
- 出力は以下の形式で行なうこと.

  ```
  素数（3 個）: 7 3 11
  合成数（2 個）: 20 8
  ```

  なお，上記は, 5 つの整数 `11 8 3 7 20` に対する出力である.
- 今回までの講義で紹介されていない C 言語の機能 (関数など) を使ってもよい．ただし，該当する部分でどのような処理が行なわれるのかをコメント として記載すること

(さらに続く)

レポート課題 III-4 (3)

レポート課題 III-4 の仕様 (さらに続き)

- 「入力されたのと逆順に出力するだけでは物足りない」人は「素数と合成数のそれぞれを小さい順に出力するプログラム」を作成して提出せよ．(ソースコードにその旨をコメントすること).

提出期限：2015 年 7 月 17 日 (金)

レポート III-4 の評価基準 (1)

必須要素

守られていない場合は減点

- 提出ファイル名 は適切か
- C 言語ソースファイル と 出力結果 を提出しているか
- gcc でコンパイルでき，生成したファイルを実行できるか
- 83 32 21 19 97 76 63 3 35 51 の入力に対して 適切な出力 がされるか
- 式や処理について 充分なコメント が記載されているか

加点要素 (1)：技術の習得

- 講義で使っていない機能 の利用
  - 充分なコメント が付されている場合に限る
  - 関数, ファイル入出力などの利用
- 下記のような入力に対しても適切に動作する頑健性
  - 15 個以上の整数が入力された場合
  - 0 個の整数が入力された場合 (e.g. 最初に -1 が入力された場合)
  - 入力された整数が全て素数 (もしくは合成数) の場合

レポート III-4 の評価基準 (2)

加点要素 (2)：創意工夫

- 指定されていない数値 についての実行結果 (ただし多くても 10 パターン程度まで)
- 「素数と合成数のそれぞれを 小さい順 に出力するプログラム」を作成できた場合は 大幅加点 (レポート III-1, 2, 3 の出来が悪かった場合, そちらもカバーできるものとする).