プログラミングII

第9回: リスト(その2)

2019年01月30日(水)

筑波大学 情報メディア創成学類
三河 正彦

ポインタのポインタ(double pointer)?

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NUM 5

void mem_alloc01(int n, double *ary);
void mem_alloc02(int n, double **ary);
void mem_free01(double *ary);
void mem_free02(double **ary);
void print_ary(int n, double *ary);

int main(void)
{
    double *array;

    /* 意図した通りに動かないバージョン */
    mem_alloc01(NUM, array);

    printf("\n");
    printf("array in main()):\n");
    print_ary(NUM, array);
    /* メモリをfreeしたいが，ダメ */
    //mem_free01(array);


    printf("\n");


    /* 意図通りに動くバージョン */
    mem_alloc02(NUM,&array);

    printf("\n");
    printf("array in main()):\n");
    print_ary(NUM, array);
    /* メモリをfree */
    mem_free02(&array);

    return 0;
}
```

```
/* 引数に通常のポインタを用い，関数内でmalloc() */
/* 意図した通りに動かないバージョン */
void mem_alloc01(int n, double *ary)
{
    int i;
    double *tmp;

    tmp = (double *)malloc(sizeof(double)*n);
    if (tmp == NULL) {
      fprintf(stderr, "memory allocation error!!\n");
      return;
    }

    /* Assign a number to array */
    for (i=0; i<n; i++) {
      tmp[i] = (double)(n - i);
    }

    ary = tmp;

    printf("tmp in mem_alloc01()):\n");
    print_ary(n, tmp);
    printf("\n");
    printf("ary in mem_alloc01()):\n");
    print_ary(n, ary);
}
```

```
/* 引数にポインタのポインタを用い，関数内でmalloc() */
/* 意図通りに動くバージョン */
void mem_alloc02(int n, double **ary)
{
    int i;
    double *tmp;

    tmp = (double *)malloc(sizeof(double)*n);
    if (tmp == NULL) {
      fprintf(stderr, "memory allocation error!!\n");
      return;
    }

    /* Assign a number to array */
    for (i=0; i<n; i++) {
      tmp[i] = (double)(2*n - i);
    }

    *ary = tmp;

    printf("tmp in mem_alloc02()):\n");
    print_ary(n, tmp);
    printf("\n");
    printf("ary in mem_alloc02()):\n");
    print_ary(n, *ary);
}
```

```
/* メモリを解法 */
void mem_free01(double *ary)
{
    free(ary);
}

/* メモリを解法 */
void mem_free02(double **ary)
{
    free(*ary);
}

/* 配列をプリントアウト */
void print_ary(int n, double *ary)
{
  int i;

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

ポインタのポインタ(double pointer)?

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NUM 5

void mem_alloc01(int n, double *ary);
void mem_alloc02(int n, double **ary);
void mem_free01(double *ary);
void mem_free02(double **ary);
void print_ary(int n, double *ary);

int main(void)
{
    double *array;

    /* 意図した通りに動かないバージョン */
    mem_alloc01(NUM, array);

    printf("\n");
    printf("array in main()):\n");
    print_ary(NUM, array);
    /* メモリをfreeしたいが，ダメ */
    //mem_free01(array);


    printf("\n");


    /* 意図通りに動くバージョン */
    mem_alloc02(NUM,&array);

    printf("\n");
    printf("array in main()):\n");
    print_ary(NUM, array);
    /* メモリをfree */
    mem_free02(&array);

    return 0;
}
```

```c
/* 引数に通常のポインタを用い，関数内でmalloc() */
/* 意図した通りに動かないバージョン */
void mem_alloc01(int n, double *ary)
{
    int i;
    double *tmp;

    tmp = (double *)malloc(sizeof(double)*n);
    if (tmp == NULL) {
       fprintf(stderr, "memory allocation error!!\n");
       return;
    }

    /* Assign a number to array */
    for (i=0; i<n; i++) {
       tmp[i] = (double)(n - i);
    }

    ary = tmp;

    printf("tmp in mem_alloc01()):\n");
    print_ary(n, tmp);
    printf("\n");
    printf("ary in mem_alloc01()):\n");
    print_ary(n, ary);
}
```

```c
/* 引数にポインタのポインタを用い，関数内でmalloc() */
/* 意図通りに動くバージョン */
void mem_alloc02(int n, double **ary)
{
    int i;
    double *tmp;

    tmp = (double *)malloc(sizeof(double)*n);
    if (tmp == NULL) {
       fprintf(stderr, "memory allocation error!!\n");
       return;
    }

    /* Assign a number to array */
    for (i=0; i<n; i++) {
       tmp[i] = (double)(2*n - i);
    }

    *ary = tmp;

    printf("tmp in mem_alloc02()):\n");
    print_ary(n, tmp);
    printf("\n");
    printf("ary in mem_alloc02()):\n");
    print_ary(n, *ary);
}
```

```c
/* メモリを解法 */
void mem_free01(double *ary)
{
```

```
# ./a.out
tmp in mem_alloc01()):
ary[0] = 5.000000
ary[1] = 4.000000
ary[2] = 3.000000
ary[3] = 2.000000
ary[4] = 1.000000

ary in mem_alloc01()):
ary[0] = 5.000000
ary[1] = 4.000000
ary[2] = 3.000000
ary[3] = 2.000000
ary[4] = 1.000000

array in main()):
ary[0] = 0.000000
ary[1] = -1138377162266632834252 8.000000
ary[2] = 0.000000
ary[3] = 3139512976027253935007643892333 8
ary[4] = 1400192780276268603297585434234 9

tmp in mem_alloc02()):
ary[0] = 10.000000
ary[1] = 9.000000
ary[2] = 8.000000
ary[3] = 7.000000
ary[4] = 6.000000

ary in mem_alloc02()):
ary[0] = 10.000000
ary[1] = 9.000000
ary[2] = 8.000000
ary[3] = 7.000000
ary[4] = 6.000000

array in main()):
ary[0] = 10.000000
ary[1] = 9.000000
ary[2] = 8.000000
ary[3] = 7.000000
ary[4] = 6.000000
#
```

ポインタのポインタ(double pointer)?

```
/* 引数に通常のポインタを用い，関数内でmalloc() */
/* 意図した通りに動かないバージョン */
void mem_alloc01(int n, double *ary)
{
  int i;
  double *tmp;

  tmp = (double *)malloc(sizeof(double)*n);
  if (tmp == NULL) {
     fprintf(stderr, "memory allocation error\n");
     return;
  }

  /* 適当な数値を代入 */
  for (i=0; i<n; i++) {
     tmp[i] = (double)(n - i);
  }

  ary = tmp;

  printf("tmp in mem_alloc01()):\n");
  print_ary(n, tmp);
  printf("\n");
  printf("ary in mem_alloc01()):\n");
  print_ary(n, ary);
}
```

```
/* 引数にポインタのポインタを用い，関数内でmalloc() */
/* 意図通りに動くバージョン */
void mem_alloc02(int n, double **ary)
{
  int i;
  double *tmp;

  tmp = (double *)malloc(sizeof(double)*n);
  if (tmp == NULL) {
     fprintf(stderr, "memory allocation error!!\n");
     return;
  }

  /* 適当な数値を代入 */
  for (i=0; i<n; i++) {
     tmp[i] = (double)(2*n - i);
  }

  *ary = tmp;

  printf("tmp in mem_alloc02()):\n");
  print_ary(n, tmp);
  printf("\n");
  printf("ary in mem_alloc02()):\n");
  print_ary(n, *ary);
}
```

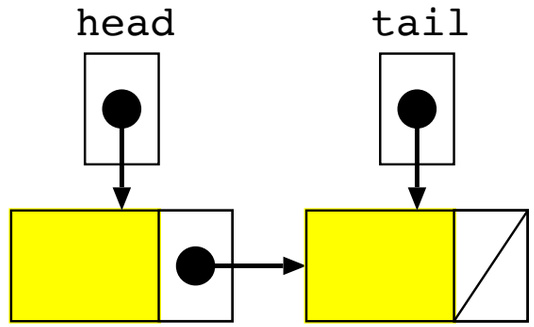
llist-func.cの改造: 新たな機能の追加

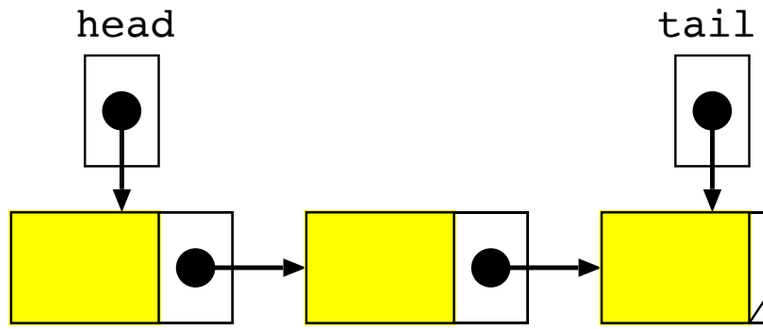
- `list-func.c` (先週の資料を参照) の関数`del_a_data()`を改造し，キーボードから数字の`n`を受け取り，リストの`n`番目の要素を削除する関数`del_n_data()`を作りなさい．

del_n_data()

```
int del_n_data(struct CELL **h, struct CELL **t, int n)
{
  int    i=0;
  struct CELL *tmpc;
  struct CELL *delc;

  if (*h == NULL) {
    return 0;
  } else {
    if (n < 0) {
      printf("エラー : セルの番号が負(%d)\n", n);
      return -1;
    } else if (n == 0) {
      del_a_data(h, t);    /* セルの先頭を削除 */
      return 0;
    } else {
      tmpc = *h;
      for (i=0; i<n-1; i++) {
        if (tmpc->next->next == NULL) {
          printf("セルの番号(%d)が不正\n", n);
          printf("セルの数は%d個\n", i+2);
          return -1;
        }
        tmpc = tmpc->next;
      }
      delc = tmpc->next;
      tmpc->next = tmpc->next->next;
      free(delc);
      return 0;
    }
  }
}
```






```
/*********************************************/
/* 関数 del_n_data()                         */
/* 機能 : リンクトリストのn番目のセルを削除する     */
/* 引数 : struct CELL **h : リンクトリストの先頭   */
/*      struct CELL **t : リンクトリストの末尾      */
/*      int n : 削除するセルの番号                 */
/* 戻値 : エラー発生時 : -1                       */
/*      正常終了時 : 0                            */
/*********************************************/
```

その他の機能

- `list-func.c`の関数`add_a_data()`を改造し，キーボードから数字の`n`を受け取り，リストの`n`番目に要素を追加する関数`add_n_data()`.

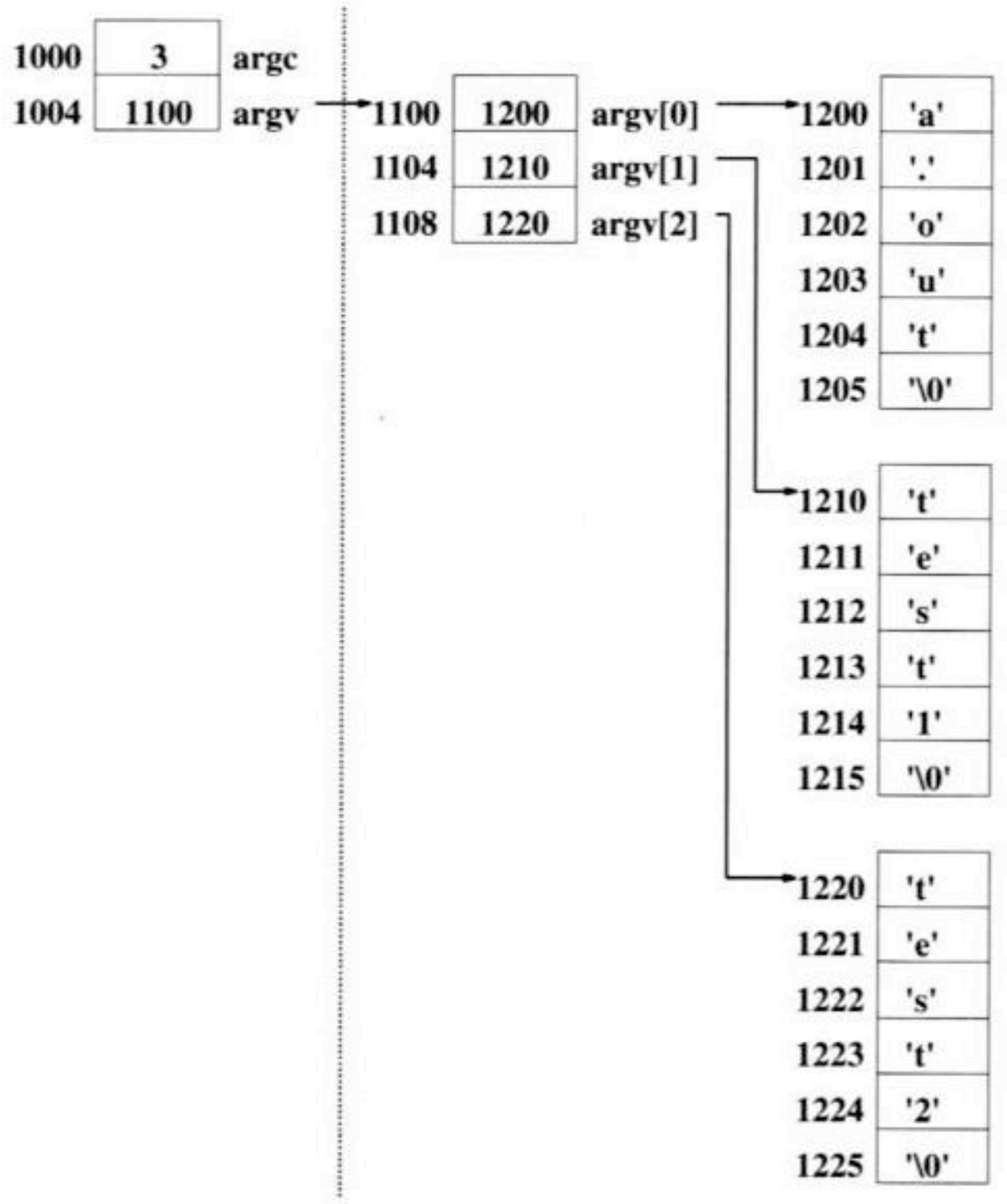
各自考えてみること.

main()関数の引数

- `main (int argc, char *argv[])`
  - `argc`: コマンド名を含めた引数の数
  - `argv`: 引数が格納されている文字列へのポインタ
  - `argv[0]`: コマンド名へのポインタ
  - `argv[1]`: 第1引数へのポインタ
  - `argv[argc-1]`: 最後の引数へのポインタ
- `% a.out test1 test2`

main()関数の引数

```
% a.out test1 test2
```



おわり

筑波大学情報メディア創成学類 三河 正彦