情報工学演習III

3. 線形(連結)リスト

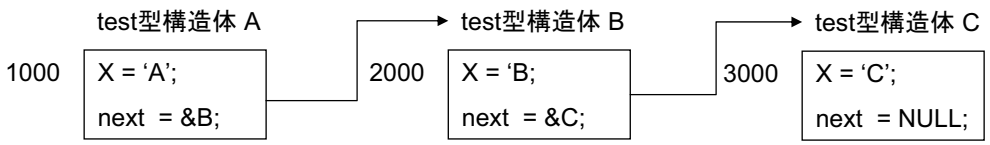
渡辺宏太郎　伊達　央

補助　リュ・ビョンジュン　岡田　美里
ブイ・クワン・ズン

データが順序付けられて並んだデータ構造は，リストとよばれます．そのリスト構造のうち，最も単純なのが線形リストあるいは連結リストとよばれる構造です．線形リスト上の個々のデータは要素(element)とよばれますが，それらは大抵構造体(自己参照構造体)で構成されています．

```
struct test {
        char x;
        struct test *next;
};
```

自己参照構造体の宣言



問39　test型構造体A,B,Cを作成し，上の図のような状態にせよ．変数Aのみを用いて，ABCと出力してみよ．

printf("%c%c%c¥n",A.x など);

テンプレートはここから得られる．

確認印

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

int main(void){

 struct test{
      char x;
      struct test *next;
 };

 struct test A, B, C;


   /*A, B, C 3つのstruct test型変数に直接入力および接続*/
   A.x = 'A';
   A.next = &B; /*変数Aの次のアドレスは&B*/
   B.x = 'B';
   B.next = &C; /*変数Bの次のアドレスは&C*/
   C.x = 'C';
   C.next = NULL; /*変数Cの次のアドレスはNULL（終了）*/

   printf("%c %c",A.x,(*(A.next)).x);
   // こんな感じで構造体C.xを表示するにはどうすればよいのか
}
```

メンバアクセス演算子	
.	構造体変数の実体のメンバを指定する.
->	構造体変数をポインタ表現したときのメンバを指定する.

*(A.next).xでいいような気がしますが，.の優先度が*より高いため，*((A.next).x)と解釈されてしまいますのでこのような表現をしています．アロー演算子は構造体変数をポインタ表現したときのメンバの指定ができます．

(*(A.next)).xは(A.next)->xでよいということになります．

こちらのほうが簡単ですね．このやり方でもOKです．

問40　問39の3つのtest型構造体A,B,Cをアドレスとして確保した場合，変数Aのみを用いて，ABCと出力してみよ．アロー演算子を用いること．

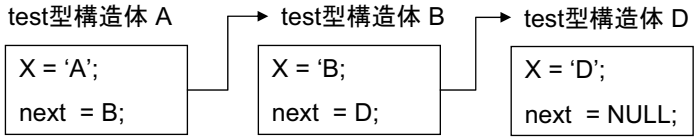
printf("%c%c%c¥n",A->x,など);

テンプレートはここから得られる．

```
/*3つのstruct test型変数をアドレスとして確保する場合*/
#include<stdio.h>
#include<stdlib.h>
int main(void){

 struct test{
      char x;
      struct test *next;
 };

 struct test *A, *B, *C;

 A = (struct test *)malloc(sizeof(struct test));
 B = (struct test *)malloc(sizeof(struct test));
 C = (struct test *)malloc(sizeof(struct test));


   /*A, B, C 3つのstruct test型変数に直接入力および接続*/
   A->x = 'A';
   A->next = ; /*変数Aの次のアドレスは&Bでいいのか？*/
   B->x = 'B';
   B->next = ; /*変数Bの次のアドレスは&Cでいいのか？*/
   C->x = 'C';
   C->next = NULL; /*変数Cの次のアドレスはNULL（終了）*/

   printf("%c %c %c\n",A->x,,);
}

/*
A, B, C を普通に宣言するのと，アドレス型で宣言するのとで
どちらが簡単に扱えるか実感できましたか？
-> 演算子はとても便利ですね.
/*
```

確認印

リストのつなぎ変え

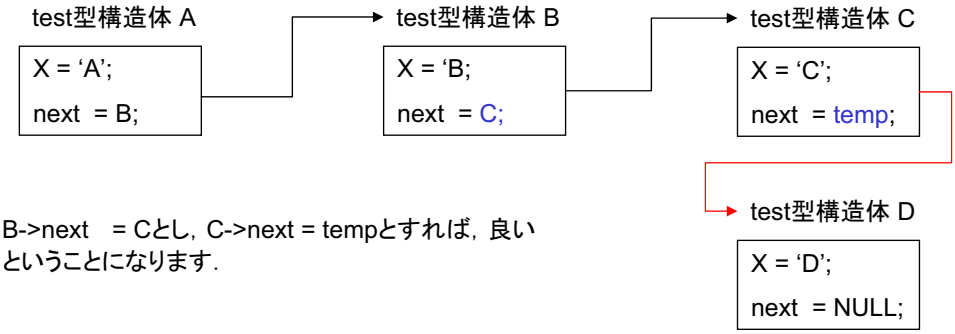


3つの構造体が上のようにつながっているとします．構造体BとDの間に構造体Cを入れるには，下図のようにします．

構造体が問40のようにmallocを使ってアドレスとして確保された場合，構造体Dのアドレスを

struct test *temp = D;

でとっておいた上で



B->next = Cとし，C->next = tempとすれば，良いということになります．

問41　以下のプログラムでデータ'A','B','D'の'B','D'の間に'C'を挿入せよ．テンプレートはここから得られる．

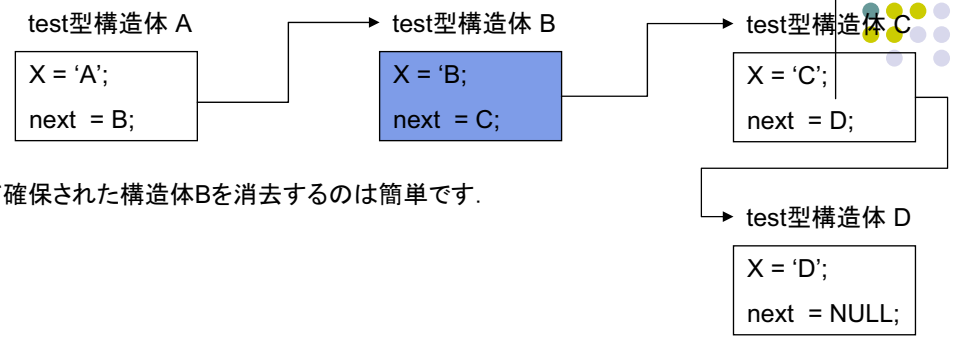
```
#include <stdio.h>
#include <stdlib.h>
int main(void){

  struct test{
    char x;
    struct test *next;
  };

  struct test *first, *p, *q, *temp;;

   /*先頭部分を作ります．char x は'A'．*/
   first = (struct test *)malloc(sizeof(struct test)); /*箱を作って*/
   first->x = 'A'; first->next = NULL;

   p = first;
   while(p->next != NULL) p = p->next;
   /*pの次にqという箱を作って挿入します（char x は 'B'）*/
   q = (struct test *)malloc(sizeof(struct test)); /*箱を作って*/
   q->x = 'B'; q->next = NULL;
   p->next = q;          /*つなげる*/

   p = first;
   while(p->next != NULL) p = p->next;
   /*pの次にqという箱を作って挿入します（char x は 'D'）*/
   q = (struct test *)malloc(sizeof(struct test)); /*箱を作って*/
   q->x = 'D'; q->next = NULL;
   p->next = q;          /*つなげる*/
```

確認印

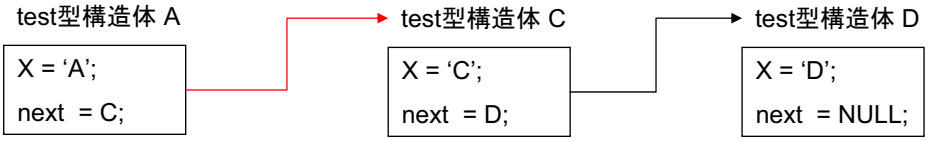
```
    /*********************************************************
    この時点で, first=Aの箱 --> Bの箱 --> Dの箱 --> NULL
    のようにつながっていることがわかりますか?
    *********************************************************/
    /*以下は先頭からリスト内を出力をする標準的方法です. */
    p = first;
    while(p != NULL){
        printf("%c",p->x); /* ABD と出力されます. */
        p = p->next;
    }
    printf("\n");

    /*問41 : BとDの間にCを挿入してみましょう*/
    /*この下にプログラムを記述*/


    /*以下は先頭からリスト内を出力をする標準的方法です. */
    p = first;
    while(p != NULL){
        printf("%c",p->x);
        p = p->next;
    }
    printf("\n");
    /* ABCDのように出力できましたか?*/

}
```

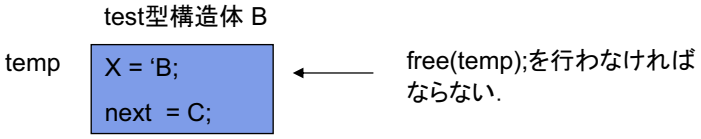
ヒント:文字'B'が格納されている構造体のアドレスを探す.

リストの削除



例えばアドレスとして確保された構造体Bを消去するのは簡単です.
temp = A->next;
として
A->next-> =C
とし,
free(temp);
とします. free(temp);をしないとメモリ空間上で, 構造体Bが使用してた部分がプログラム終了時まで使用不可になってしまいます.





問42　以下のプログラムでデータ'A','B','C','D'の'B'を削除せよ．テンプレートはここから得られる．

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

int main(void){

  struct test{
    char x;
    struct test *next;
  };

  struct test *first, *p, *q, *temp;;

  /*先頭部分を作ります．char x は'A'．*/
  first = (struct test *)malloc(sizeof(struct test)); /*箱を作って*/
  first->x = 'A'; first->next = NULL;

  p = first;
  while(p->next != NULL) p = p->next;
  /*pの次にqという箱を作って挿入します（char x は 'B'）*/
  q = (struct test *)malloc(sizeof(struct test)); /*箱を作って*/
  q->x = 'B'; q->next = NULL;
  p->next = q;          /*つなげる*/

  p = first;
  while(p->next != NULL) p = p->next;
  /*pの次にqという箱を作って挿入します（char x は 'C'）*/
  q = (struct test *)malloc(sizeof(struct test)); /*箱を作って*/
  q->x = 'C'; q->next = NULL;
  p->next = q;          /*つなげる*/

  p = first;
  while(p->next != NULL) p = p->next;
  /*pの次にqという箱を作って挿入します（char x は 'D'）*/
  q = (struct test *)malloc(sizeof(struct test)); /*箱を作って*/
  q->x = 'D'; q->next = NULL;
  p->next = q;          /*つなげる*/

   /*********************************************************
   この時点で，first=Aの箱 --> Bの箱 -->Cの箱 --> Dの箱
   --> NULL のようにつながっている
   *********************************************************/
```

確認印

```
   /*以下は先頭からリスト内を出力をする標準的方法です．*/
  p = first->next;
  while(p != NULL){
    printf("%c",p->x); /* ABCD と出力されます．*/
    p = p->next;
  }
  printf("\n");

  /*問42：Bを消去してみよ*/
  /*この下にプログラムを記述*/
  p = first->next;
  while(p->next->x != 'B') p = p->next;
  temp =


   /*以下は先頭からリスト内を出力をする標準的方法です．*/
   p = first->next;
   while(p != NULL){
   printf("%c",p->x);
   p = p->next;
   }
  printf("\n");
  /* ACDのように出力できましたか？*/

}
```

正しくプログラムできているかどうかの判定

valgrind –tool=memcheck ./a.out

を実行する．

==27337== malloc/free: 4 allocs, 1 frees, 64 bytes allocated.

という実行結果の行がありますが，今Bを開放していますから，

1 frees

と出ていればOKです．

```
[wata@csimc13 ensyuu3-10]$ gcc ex42.c
[wata@csimc13 ensyuu3-10]$ valgrind  --tool=memcheck ./a.out
==21668== Memcheck, a memory error detector.
==21668== Copyright (C) 2002-2006, and GNU GPL'd, by Julian Seward et al.
==21668== Using LibVEX rev 1658, a library for dynamic binary translation.
==21668== Copyright (C) 2004-2006, and GNU GPL'd, by OpenWorks LLP.
==21668== Using valgrind-3.2.1, a dynamic binary instrumentation framework.
==21668== Copyright (C) 2000-2006, and GNU GPL'd, by Julian Seward et al.
==21668== For more details, rerun with: -v
==21668==
ABCD
ACD
==21668==
==21668== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 4 from 1)
==21668== malloc/free: in use at exit: 48 bytes in 3 blocks.
==21668== malloc/free: 4 allocs, 1 frees, 64 bytes allocated.
==21668== For counts of detected errors, rerun with: -v
==21668== searching for pointers to 3 not-freed blocks.
==21668== checked 63,432 bytes.
==21668==
==21668== LEAK SUMMARY:
==21668==    definitely lost: 48 bytes in 3 blocks.
==21668==      possibly lost: 0 bytes in 0 blocks.
==21668==    still reachable: 0 bytes in 0 blocks.
==21668==         suppressed: 0 bytes in 0 blocks.
==21668== Use --leak-check=full to see details of leaked memory.
[wata@csimc13 ensyuu3-10]$
```

メモリチェック用のツールです．

問43　問42でデータfirst,'A','B','C','D'を全て削除せよ.

valgrind –tool=memcheck ./a.out
の実行結果が次のようになればOK

```
[wata@csimc13 ensyuu3-10]$ valgrind  --tool=memcheck ./a.out
==21703== Memcheck, a memory error detector.
==21703== Copyright (C) 2002-2006, and GNU GPL'd, by Julian Seward et al.
==21703== Using LibVEX rev 1658, a library for dynamic binary translation.
==21703== Copyright (C) 2004-2006, and GNU GPL'd, by OpenWorks LLP.
==21703== Using valgrind-3.2.1, a dynamic binary instrumentation framework.
==21703== Copyright (C) 2000-2006, and GNU GPL'd, by Julian Seward et al.
==21703== For more details, rerun with: -v
==21703==
ABCD
==21703==
==21703== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 4 from 1)
==21703== malloc/free: in use at exit: 0 bytes in 0 blocks.
==21703== malloc/free: 4 allocs, 4 frees, 64 bytes allocated.
==21703== For counts of detected errors, rerun with: -v
==21703== All heap blocks were freed -- no leaks are possible.
[wata@csimc13 ensyuu3-10]$
```

確認印

問44　問41ではfirst,A,B,D,の順に線形リストを作成し，CをBとDの間に挿入した．以下のプログラムでは，一つのtest型構造体をポインタpの後に挿入する関数

struct test *insert(char a, struct test *p, struct test *init);

がある．これを利用して問41と同じことを行え．テンプレートはここから得られる．

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

struct test{
    char x;
    struct test *next;
  };

struct test *insert(char a, struct test *p, struct test *init);

int main(void){
  struct test *front,*p;

  front = insert('A',NULL,NULL);
  front = insert('B',front,front);  /*何じゃこりゃ？でも関数insertがわかっていれば
                                      わかるはず*/

  printf("%c\n",front->x);
  printf("%c\n",front->next->x);


  /* この後にDを挿入せよ */
  /* front = insert('D',front,front); では間違い*/
  /*関数insertはポインタpの指すセルの　次に　aのセルを挿入，p=NULLならば先頭に挿入
    ということに注意せよ*/


  /* さらにデータ'C'を構造体BとDの間に挿入せよ */
  p = front;
  while(/*適当に埋める*/){
    /*適当に埋める*/
  }
  front = /*適当に埋める*/

  /* 先頭からデータの表示を行え　*/


  return(0);
}
```

確認印

```c
struct test *insert(char a, struct test *p, struct test *init){
  /*ポインタpの指すセルの次にaのセルを挿入，p=NULLならば先頭に挿入*/
  struct test *q, *r;

  r = (struct test *)malloc(sizeof(struct test));
  if(p==NULL){
    q = init;
    init = r;
    init->x = a;
    init->next = q;
  }else{
    q = p->next;
    p->next = r;
    r->x = a;
    r->next = q;
  }
  return(init);
}
```

Initが先頭の線形リストのポインタpで指す構造体の次にデータaを格納する構造体を挿入する関数．もし，pがNULLならば，先頭(init)に挿入します．

教科書「Cによるアルゴリズムとデータ構造」 茨木　俊秀著

p.30を参照

問45　以下のプログラムでは，ポインタpの指すアドレスのテスト型構造体を削除する関数

struct test *delete(struct test *p, struct test *init);

がある．これを利用して問42と同じことを行え．ただし，削除するリストは(1)'C'を含むリスト，(2)'A'を含むリスト　とする．テンプレートはここから得られる．

```c
struct test *delete(struct test *p, struct test *init){
  /* ポインタpが指すセルを除去 */
  struct test *q;

  if(p == init){/*先頭を削除するかどうかの判定*/
    q = p->next;
    free(p);
    return(q);
  }

  q = init;
  while(q->next != p){
    q = q->next;
  }
  q->next = p->next;
  free(p);

  return(init);

}
```

確認印

問46　test型構造体のi番目のセルの内容を返す関数

struct test *find(int i, struct test *init);

を作成せよ．もちろん，適当な例で確かめること．

確認印

問題(1)

- 問47　問40で空白が入るまでリストを延ばし続ける関数
  void make_list(struct test *p)
  を作成せよ.

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

struct test{
    char x;
    struct test *next;
  };

struct test *insert(char a, struct test *p, struct test *init);
struct test *make_list(void);

int main(void){
  struct test *front,*p;

  front = make_list();

  /* 先頭からデータの表示を行え  */


}


struct test *make_list(void){
  char z;
  struct test *front,*p;
  front = NULL;  /*初期値はNULLポインタにしておく*/

  while(1){
    printf("Input Character:");
    scanf("%c",&z);
    getchar();

    /* 適当に埋める*/
  }

}
```

改行がバッファに残っているため改行をバッファから除きます.

問題(2)

- 問48　test型構造体の線形リストで位置pの後および前のセルの位置(そのセルを指すポインタ)をポインタ配列m[2]へセットする関数
  void next(struct test *p, struct test *init, struct test **m);
  を作成せよ. もちろん, 適当な例で確かめること. 以下同様.

- 問49　test型構造体の線形リストで要素xがリスト中に存在すれば, その位置を返す関数
  struct test *locate(char x, struct test *init);
  を作成せよ.

- 問50　test型構造体の線形リストで先頭から位置iのセルの内容を返す関数
  char retrieve(int i, struct test *init);
  を作成せよ.

- 問51　struct test{
  　　char x;
  　　struct test *next;
  　　struct test *previous;
  　}
  のようにtest型構造体を定義し, 線形リストで現在の位置から直前のリストの要素, 直後のリストの要素に移動できるようにした線形リストを双方向リストという. 教科書「Cによるアルゴリズムとデータ構造」p.32参照.
  struct test *insert(char a, struct test *p, struct test *init);
  struct test *delete(struct test *p, struct test *init);
  を双方向リスト版に拡張してみよ.

レポート問題（3）

- 問52
  struct test *next(struct test *p, struct test *init);
  struct test *locate(char x, struct test *init);
  char retrieve(int i, struct test *init);
  を双方向リスト版に拡張してみよ.

予告問題

問53　各自の所持しているc言語のマニュアル（教科書）を提示せよ．以後，演習IIIの時には携行する事.

これは，レポート問題ではありません.