

2 部データ構造とアルゴリズム I レポート課題 2

18NC021

カトリ スザン

Contents

1	課題 1 のプログラム	2
1.1	shohin.h	2
1.2	shohin.c	2
1.3	uriage.h	2
1.4	uriage.c	3
2	課題 2	4
2.1	課題 2-1	4
2.1.1	uriagelist.h	4
2.1.2	uriagelist.c	4
2.1.3	テストプログラム 2-1	4
2.1.4	実行結果	5
2.1.5	プログラムの解説	5
2.2	課題 2-2	5
2.2.1	uriagelist.h	5
2.2.2	uriagelist.c	6
2.2.3	テストプログラム 2-2	6
2.2.4	実行結果	8
2.2.5	プログラムの解説	8
2.3	課題 2-3	8
2.3.1	uriagelist.h	8
2.3.2	uriagelist.c	8
2.3.3	テストプログラム 2-3	9
2.3.4	実行結果	11
2.3.5	プログラムの解説	11
2.4	課題 2-4	12
2.4.1	uriagelist.h	12
2.4.2	uriagelist.c	12
2.4.3	テストプログラム 2-4	13
2.4.4	実行結果	14
2.4.5	プログラムの解説	14
2.5	考察	14

1 課題 1 のプログラム

課題2では課題1で作った以下のプログラムを使う。

1.1 shohin.h

```
#ifndef C_ALGO_REPORT2_SHOHIN_H
#define C_ALGO_REPORT2_SHOHIN_H

#endif //C_ALGO_REPORT2_SHOHIN_H

#define ZEI (1.08)
typedef struct {
    char* name;
    int tanka;
    int sotozei;
} SHOHIN;
extern SHOHIN shohin[];
void printshohin(SHOHIN s);
```

1.2 shohin.c

```
#include <stdio.h>
#include "shohin.h"

SHOHIN shohin[] = {{ "Apple", 150},
                   { "Orange", 100},
                   { "Banana", 200},
                   { "Book1", 500, 1},
                   { "", 0}};

void printshohin(SHOHIN s) {
    printf("%s\t単価%d円(%s)", s.name, s.tanka, s.sotozei ? "外税" : "内税");
}
```

1.3 uriage.h

```
#ifndef C_ALGO_REPORT2_URIAGE_H
#define C_ALGO_REPORT2_URIAGE_H

#endif //C_ALGO_REPORT2_URIAGE_H

typedef struct {
    int code;
    int num;
} URIAGE;

int printUriage(URIAGE* q);
int printUriageArray(URIAGE q[]);
int printUriageTrans(URIAGE **p);
```

1.4 uriage.c

```
#include <stdio.h>
#include "uriage.h"
#include "shohin.h"

int printUriage(URIAGE *p) {

    char *const product_name = shohin[p->code].name;
    const int item_price = shohin[p->code].tanka;
    const int tax = shohin[p->code].sotozei;
    const int total_sales = p->num;
    const int total_price =
        tax ? (item_price * total_sales) * ZEI : item_price * total_sales;
    printf("%s\t単価%d円(%s)\t%d\t個\t%d円\n", product_name, item_price,
        tax ? "外税" : "内税", total_sales, total_price);
    return total_price;
}

int printUriageArray(URIAGE u[]) {
    int shokei = 0;
    for (int i = 0; u[i].code != -1; i++) {
        shokei += printUriage(&u[i]);
    }
    return shokei;
}

int printUriageTrans(URIAGE **u) {
    int sub_total = 0;
    int total = 0;
    for (int i = 0; u[i] != NULL; i++) {
        sub_total = printUriageArray(*(u + i));
        total += sub_total;
        printf("小計:\t\t\t\t%d円\n", sub_total);
        printf("_____\n");
    }
    printf("合計:\t\t\t%d円\n", total);
    return total;
}
```

2 課題 2

2.1 課題 2-1

- 線形リストの内容を表示する関数 `void printUriageList(URIAGELIST* l)` を作成しなさい。

2.1.1 uriagelist.h

```
#ifndef C_ALGO_REPORT2_URIAGELIST_H
#define C_ALGO_REPORT2_URIAGELIST_H
#endif //C_ALGO_REPORT2_URIAGELIST_H

typedef struct uriagelist {
    struct uriagelist* next;
    URIAGE *uriage;
} URIAGELIST;

void printUriageList(URIAGELIST* l);
```

2.1.2 uriagelist.c

```
#include <stdio.h>
#include <stdlib.h>
#include "uriage.h"
#include "shohin.h"
#include "uriagelist.h"

void printUriageList(URIAGELIST *l) {
    if (l->uriage == NULL) {
        return;
    }
    while (l->next != NULL) {
        // 課題 1 の uriage.c で定義されている printUriage 関数を使っている。
        printUriage(l->uriage);
        l = l->next;
    }
}
```

2.1.3 テストプログラム 2-1

```
#include <stdio.h>
#include <stdlib.h>
#include "shohin.h"
#include "uriage.h"
#include "uriagelist.h"
int main(void){
    URIAGE u1={1,2};
    URIAGE u2={2,3};
    URIAGE u3={3,4};
    URIAGELIST l0={NULL};
    URIAGELIST l1={&l0,&u1};
    URIAGELIST l2={&l1,&u2};
    URIAGELIST l3={&l2,&u3};
    printf("-----\n");
    printUriageList(&l0);
```

```

printf("-----\n");
printUriageList(&l1);
printf("-----\n");
printUriageList(&l2);
printf("-----\n");
printUriageList(&l3);
return 0;
}

```

2.1.4 実行結果

```

falcon@falcon: ~/Projects/2nd_year_algorithm_first_sem/C_Algo_Report2
File Edit View Search Terminal Help
falcon@falcon: gcc problem2-1.c -o problem2-1 uriage.c shohin.c uriagelist.c
falcon@falcon: ./problem2-1
-----
-----
Orange   単価100円(内税)      2 個    200円
-----
Banana   単価200円(内税)      3 個    600円
Orange   単価100円(内税)      2 個    200円
-----
Book1    単価500円(外税)      4 個   2160円
Banana   単価200円(内税)      3 個    600円
Orange   単価100円(内税)      2 個    200円
falcon@falcon:

```

図 1: 課題 2-1 実行結果

2.1.5 プログラムの解説

- このプログラムでは課題1で定義した、URIAGE 型、SHOHIN 型と printUriage 関数を使っている。

線形リストを構成する、一つひとつのノードは、データと次のノードを指すポインタとでできていて、このプログラムでは複数個のものが一つにまとまったものを構成するため構造体 (: URIAGELIST) をつかっている。URIAGELIST 構造体は次のノードのポインタと URIAGE 型のポインタ (uriage) から構成されている。URIAGELIST は uriagelist.h ヘッダーファイルで定義している。

uriagelist.c の printUriageList は URIAGELIST 型のポインタ (*l) を引数とする関数であり、*l から始まる線形リストを最後のノード (NULL) まで順番に辿り、各ノードのデータの部分である uriage に格納されている URIAGE 型ポインタを printUriage 関数に渡している、また、渡されたポインタが NULL の場合はなにもせずに return する。printUriage 関数はこのポインタが指すデータ (商品) を出力する。

2.2 課題 2-2

- URIAGELIST のノードを動的に確保する関数 URIAGELIST* newlist(void) を作成しなさい。

2.2.1 uriagelist.h

```

#ifndef C_ALGO_REPORT2_URIAGELIST_H
#define C_ALGO_REPORT2_URIAGELIST_H

#endif //C_ALGO_REPORT2_URIAGELIST_H

typedef struct uriagelist {
    struct uriagelist* next;

```

```

    URIAGE *uriage;
} URIAGELIST;

void printUriageList (URIAGELIST* l);
URIAGELIST* newlist (void);

```

2.2.2 uriagelist.c

```

#include <stdio.h>
#include <stdlib.h>
#include "uriage.h"
#include "shohin.h"
#include "uriagelist.h"

void printUriageList (URIAGELIST *l) {
    if (l->uriage == NULL) {
        return;
    }
    while (l->next != NULL) {
        // 課題 1 の uriage.c で定義されている printUriage 関数を使っている。
        printUriage (l->uriage);
        l = l->next;
    }
}

URIAGELIST *newlist (void) {
    URIAGELIST *newNode = malloc (sizeof (URIAGELIST));
    if (newNode == NULL) {
        return NULL;
    } else {
        newNode->next = NULL;
        newNode->uriage = NULL;
        return newNode;
    }
}

```

2.2.3 テストプログラム 2-2

```

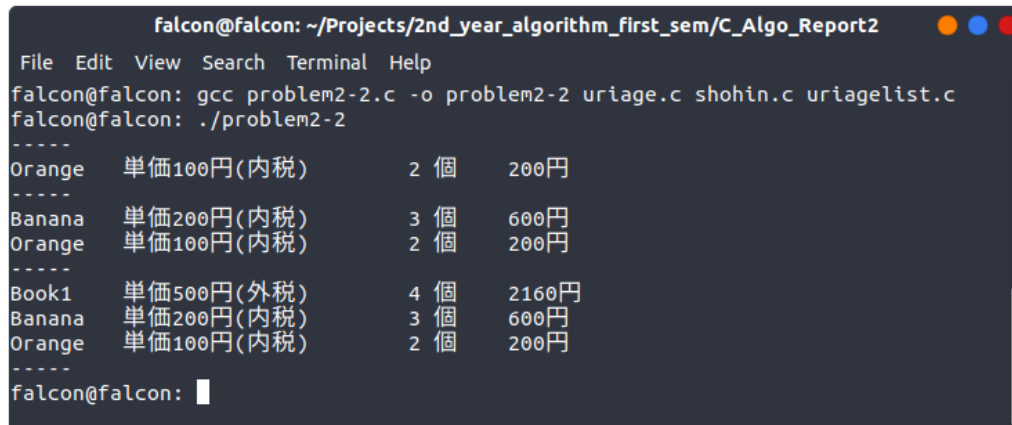
#include <stdio.h>
#include <stdlib.h>
#include "shohin.h"
#include "uriage.h"
#include "uriagelist.h"

int main (void) {
    URIAGE u1 = {1, 2};
    URIAGE u2 = {2, 3};
    URIAGE u3 = {3, 4};
    URIAGELIST *l0;
    URIAGELIST *l1;
    URIAGELIST *l2;
    URIAGELIST *l3;
    if ((l0 = newlist ()) == NULL) {
        fprintf (stderr, "領域を確保できませんでした");
    }
}

```

```
        return 2;
    }
    if ((l1 = newlist()) == NULL) {
        fprintf(stderr, "領域を確保できませんでした");
        free(l0);
        return 2;
    }
    if ((l2 = newlist()) == NULL) {
        fprintf(stderr, "領域を確保できませんでした");
        free(l1);
        free(l0);
        return 2;
    }
    if ((l3 = newlist()) == NULL) {
        fprintf(stderr, "領域を確保できませんでした");
        free(l2);
        free(l1);
        free(l0);
        return 2;
    }
    printUriageList(l0);
    printf("-----\n");
    l1->next = l0;
    l1->uriage = &u1;
    printUriageList(l1);
    printf("-----\n");
    l2->next = l1;
    l2->uriage = &u2;
    printUriageList(l2);
    printf("-----\n");
    l3->next = l2;
    l3->uriage = &u3;
    printUriageList(l3);
    printf("-----\n");
    free(l3);
    free(l2);
    free(l1);
    free(l0);
    return 0;
}
```


2.2.4 実行結果



```
falcon@falcon: ~/Projects/2nd_year_algorithm_first_sem/C_Algo_Report2
File Edit View Search Terminal Help
falcon@falcon: gcc problem2-2.c -o problem2-2 uriage.c shohin.c uriagelist.c
falcon@falcon: ./problem2-2
-----
Orange   単価100円(内税)      2 個    200円
-----
Banana   単価200円(内税)      3 個    600円
Orange   単価100円(内税)      2 個    200円
-----
Book1    単価500円(外税)      4 個   2160円
Banana   単価200円(内税)      3 個    600円
Orange   単価100円(内税)      2 個    200円
-----
falcon@falcon: 
```

図 2: 課題 2-2 実行結果

2.2.5 プログラムの解説

- このプログラムでは課題1で定義した、URIAGE 型、SHOHIN 型を使っている。

関数 newList は引数がなく、返り値が URIAGELIST 型の関数である。この関数を呼び出すと malloc を使ってヒープ領域に URIAGELIST の大きさ分のメモリを確保し、メモリアロケーションが成功すると newNode 変数にメモリ上に確保した領域への参照ポインタが格納される。もしアロケーションに失敗すると malloc 関数から NULL が返され、newNode に格納される。newNode が NULL の場合は NULL を返す。

2.3 課題 2-3

- 線形リストにおいて、動的に確保したすべてのノードを free 関数により解放する関数 void freeUriageList(URIAGELIST* l, int purge) を作成しなさい。

2.3.1 uriagelist.h

```
#ifndef C_ALGO_REPORT2_URIAGELIST_H
#define C_ALGO_REPORT2_URIAGELIST_H

#endif //C_ALGO_REPORT2_URIAGELIST_H

typedef struct uriagelist {
    struct uriagelist* next;
    URIAGE *uriage;
} URIAGELIST;

void printUriageList(URIAGELIST* l);
URIAGELIST* newlist(void);
void freeUriageList(URIAGELIST *l, int purge);
```

2.3.2 uriagelist.c

```
#include <stdio.h>
#include <stdlib.h>
#include "uriage.h"
#include "shohin.h"
```

```

#include "uriagelist.h"

void printUriageList (URIAGELIST *l) {
    if (l->uriage == NULL) {
        return;
    }
    while (l->next != NULL) {
        // 課題 1 の uriage.c で定義されている printUriage 関数を使っている。
        printUriage(l->uriage);
        l = l->next;
    }
}

URIAGELIST *newlist(void) {
    URIAGELIST *newNode = malloc(sizeof(URIAGELIST));
    if (newNode == NULL) {
        return NULL;
    } else {
        newNode->next = NULL;
        newNode->uriage = NULL;
        return newNode;
    }
}

void freeUriageList (URIAGELIST *l, int purge) {
    while (l->next != NULL) {
        if (purge == 1) {
            free(l->uriage);
        }
        URIAGELIST *previous_l = l;
        l = l->next;
        free(previous_l);
    }
}

```

2.3.3 テストプログラム 2-3

```

#include <stdio.h>
#include <stdlib.h>
#include "shohin.h"
#include "uriage.h"
#include "uriagelist.h"
int main(void){
    URIAGE u1={1,2};
    URIAGE u2={2,3};
    URIAGE u3={3,4};
    URIAGE *up1;
    URIAGE *up2;
    URIAGE *up3;
    URIAGELIST *l0;
    URIAGELIST *l1;
    URIAGELIST *l2;
    URIAGELIST *l3;
    if((l0=newlist())==NULL){
        fprintf(stderr, "領域を確保できませんでした");
        return 2;
    }
}

```

```

}
if((l1=newlist())==NULL){
    fprintf(stderr,"領域を確保できませんでした");
    free(l0);
    return 2;
}
if((l2=newlist())==NULL){
    fprintf(stderr,"領域を確保できませんでした");
    free(l1);
    free(l0);
    return 2;
}
if((l3=newlist())==NULL){
    fprintf(stderr,"領域を確保できませんでした");
    free(l2);
    free(l1);
    free(l0);
    return 2;
}
l1->next=l0;
l1->uriage=&u1;
l2->next=l1;
l2->uriage=&u2;
l3->next=l2;
l3->uriage=&u3;
printUriageList(l3);
printf("-----\n");
freeUriageList(l3,0);

if((l0=newlist())==NULL){
    fprintf(stderr,"領域を確保できませんでした");
    return 2;
}
if((l1=newlist())==NULL){
    fprintf(stderr,"領域を確保できませんでした");
    free(l0);
    return 2;
}
if((l2=newlist())==NULL){
    fprintf(stderr,"領域を確保できませんでした");
    free(l1);
    free(l0);
    return 2;
}
if((l3=newlist())==NULL){
    fprintf(stderr,"領域を確保できませんでした");
    return 2;
}
if((up1=malloc(sizeof(URIAGE)))==NULL){
    fprintf(stderr,"領域を確保できませんでした");
    free(l2);
    free(l1);
    free(l0);
    return 2;
}
if((up2=malloc(sizeof(URIAGE)))==NULL){

```

```

        fprintf(stderr, "領域を確保できませんでした");
        free(l2);
        free(l1);
        free(l0);
        free(up1);
        return 2;
    }
    if((up3=malloc(sizeof(URIAGE)))==NULL){
        fprintf(stderr, "領域を確保できませんでした");
        free(l2);
        free(l1);
        free(l0);
        free(up2);
        free(up1);
        return 2;
    }
    *up1=u1;
    *up2=u2;
    *up3=u3;
    l1->next=l0;
    l1->uriage=up1;
    l2->next=l1;
    l2->uriage=up2;
    l3->next=l2;
    l3->uriage=up3;
    printUriageList(l3);
    printf("-----\n");
    freeUriageList(l3,1);
    return 0;
}

```

2.3.4 実行結果

```

falcon@falcon: ~/Projects/2nd_year_algorithm_first_sem/C_Algo_Report2
File Edit View Search Terminal Help
falcon@falcon: gcc problem2-3.c -o problem2-3 uriage.c shohin.c uriagelist.c
falcon@falcon: ./problem2-3
Book1   単価500円(外税)      4 個    2160円
Banana  単価200円(内税)       3 個     600円
Orange  単価100円(内税)        2 個     200円
-----
Book1   単価500円(外税)      4 個    2160円
Banana  単価200円(内税)       3 個     600円
Orange  単価100円(内税)        2 個     200円
-----
falcon@falcon: 

```

図 3: 課題 2-3 実行結果

2.3.5 プログラムの解説

- このプログラムでは課題1で定義した、URIAGE 型、SHOHIN 型を使っている。

freeUriageList は URIAGELIST 型のポインタ (*l) と int 型の purge を引数とし、戻り値が void の関数である。この関数で引数として受け取ったポインタ *l はある線形リストの先頭のノードを指していて、この関数を実行すると線形リストのすべてのノードが使っているメモリを開放する。また、この線形リストはデータへの参照ポインタを含む uriage と次のノードのポインタを含む next で構成されており、引数の purge が1の場合は各ノードの uriage ポイン

ターが参照しているデータが使ってメモリも開放する。purge が0の場合はすべてのノードにアロケートされている領域だけを開放する。

2.4 課題 2-4

- URIAGE のポインターを受け取ると、URIAGELIST のノードで作られた線形リストの先頭に、ノードを追加することで追加する URIAGELIST* add(URIAGELIST* l, URIAGE* u) 関数を作成しなさい。

2.4.1 uriagelist.h

```
#ifndef C_ALGO_REPORT2_URIAGELIST_H
#define C_ALGO_REPORT2_URIAGELIST_H

#endif //C_ALGO_REPORT2_URIAGELIST_H

typedef struct uriagelist {
    struct uriagelist* next;
    URIAGE *uriage;
} URIAGELIST;

void printUriageList(URIAGELIST* l);
URIAGELIST* newlist(void);
void freeUriageList (URIAGELIST *l , int purge);
URIAGELIST* add(URIAGELIST* l, URIAGE* u);
```

2.4.2 uriagelist.c

```
#include <stdio.h>
#include <stdlib.h>
#include "uriage.h"
#include "shohin.h"
#include "uriagelist.h"

void printUriageList(URIAGELIST *l) {
    if (l->uriage == NULL) {
        return;
    }
    while (l->next != NULL) {
        // 課題 1 の uriage.c で定義されている printUriage 関数を使っている。
        printUriage(l->uriage);
        l = l->next;
    }
}

URIAGELIST *newlist(void) {
    URIAGELIST *newNode = malloc(sizeof(URIAGELIST));
    if (newNode == NULL) {
        return NULL;
    } else {
        newNode->next = NULL;
        newNode->uriage = NULL;
        return newNode;
    }
}
```

```

void freeUriageList (URIAGELIST *l, int purge) {
    while (l->next != NULL) {
        if (purge == 1) {
            free(l->uriage);
        }
        URIAGELIST *previous_l = l;
        l = l->next;
        free(previous_l);
    }
}

URIAGELIST* add (URIAGELIST* l, URIAGE *u){
    URIAGELIST *newNode = newlist();
    newNode->uriage = l->uriage;
    newNode->next = l->next;
    l->uriage = u;
    l->next = newNode;
}

```

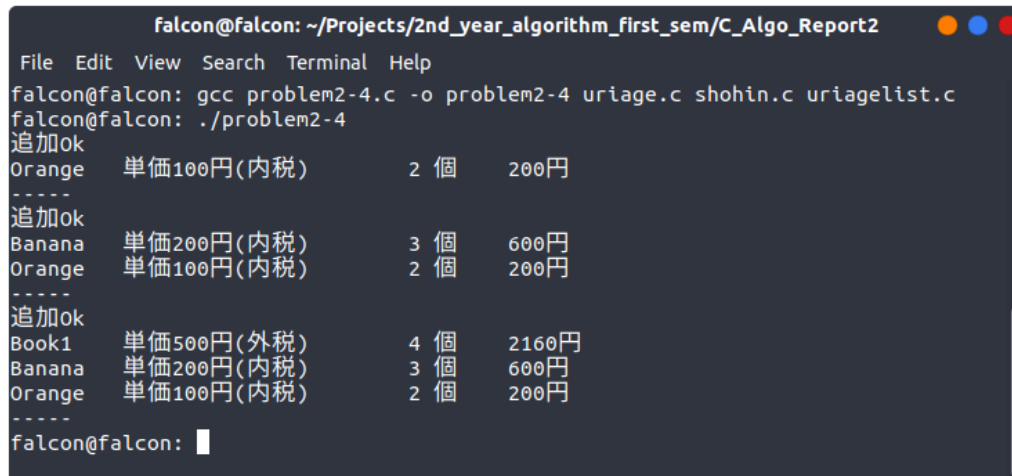
2.4.3 テストプログラム 2-4

```

#include <stdio.h>
#include <stdlib.h>
#include "shohin.h"
#include "uriage.h"
#include "uriagelist.h"
int main(void){
    URIAGE u1={1,2};
    URIAGE u2={2,3};
    URIAGE u3={3,4};
    URIAGE* u[]={&u1,&u2,&u3,NULL};
    URIAGE **p;
    URIAGELIST *l;
    URIAGELIST *m;
    if((l=newlist())==NULL){
        fprintf(stderr,"領域を確保できませんでした");
        return 2;
    }
    for(p=u;*p!=NULL;p++){
        if((m=add(l,*p))==NULL){
            fprintf(stderr,"領域を確保できませんでした");
            freeUriageList(l,0);
            return 2;
        }
        printf("追加%s\n", m->uriage==*p?"Ok":"NG");
        printUriageList(l);
        printf("-----\n");
    }
    freeUriageList(l,0);
    return 0;
}

```

2.4.4 実行結果



```
falcon@falcon: ~/Projects/2nd_year_algorithm_first_sem/C_Algo_Report2
File Edit View Search Terminal Help
falcon@falcon: gcc problem2-4.c -o problem2-4 uriage.c shohin.c uriagelist.c
falcon@falcon: ./problem2-4
追加ok
Orange  単価100円(内税)      2 個      200円
-----
追加ok
Banana  単価200円(内税)      3 個      600円
Orange  単価100円(内税)      2 個      200円
-----
追加ok
Book1   単価500円(外税)      4 個      2160円
Banana  単価200円(内税)      3 個      600円
Orange  単価100円(内税)      2 個      200円
-----
falcon@falcon: 
```

図 4: 課題 2-4 実行結果

2.4.5 プログラムの解説

- このプログラムでは課題1で定義した、URIAGE 型、SHOHIN 型と printUriage 関数を使っている。

関数 add は URIAGELIST 型のポインター l と URIAGE 型のポインター u を引数とし、返り値が URIAGELIST 型のポインターである。この関数では最初に、問題 2-2 で作った newlist 関数を使ってヒープ領域にメモリをアロケートし、URIAGELIST 型の空の線形リストを作る。そして、その線形リストへの参照ポインターを newNode 変数に格納する。次に、その空の線形リストの uriage メンバーに、引数として受け取ったポインター u を格納し、next には引数として受け取った線形リストの先頭ノードを指しているポインター (*l) を格納する。そして *l の next を newNode への参照ポインターで置き換える。こうすることで newNode が引数 *l の線形リストの先頭ノードになる。

2.5 考察

今回の課題を解いてレポート書くにあたって、構造体のメンバーに別の構造体のポインターを格納するパラダイムについて理解することができた。また、これを使って線形リストを構成する方法や線形リストの値を参照する方法について理解できた。問 2-4 では線形リストの削除や順序を変更する方法そして、線形リストの任意のポイントにノードを追加する方法について理解できた。

問 2-3 ではヒープ領域にメモリをアロケートしたり、アロケートしたメモリを使わなくなったら開放する方法について理解できた。