ルビス　ルスファン　アンシャー
15_15953

課題（Queue の実装）
プログラム名：queue.c

プログラムの中、node という構造体が宣言された。

```
struct node{
        int val;
        struct node *next;                    //連結リストになる
};
```

プログラムは 5 つの関数から成り立つ。
put 関数は先頭にデータを追加する。

```
int put(struct node *head, int val){
        //head->next に代入するための新しいアドレスを作成
        struct node *new;
        new = (struct node *)malloc(sizeof(struct node));
        if (new == NULL) return -1;

        if (head -> val == -1){
                //head に{-1}のある場合、-1 を追加したいデータと交換
                head -> val = val;
                head -> next = NULL;
        } else {
                //そうではない場合、新しいアドレスに head の val と next を代入し、head のアドレスに追加したいデータの val と next を入れる。そうすると、head のアドレスは不変。
                new -> val = head -> val;
                new -> next = head -> next;
                head -> val = val;
                head -> next = new;
        }
```

```
        return 0;
}
```

get 関数は最初に queue に入った val、つまり末尾にあるものを取り出して、そのアドレスを queue から削除する。

```
int get(struct node *head){
        struct node *t = head;

        if (t == NULL) return -1;

        //(t -> next) -> next は NULL になるまで、ループをする
        while((t -> next) -> next != NULL){
                t = t -> next;
        }

        int p = (t -> next) -> val;             // (t -> next) -> val が求めた値
        t -> next = NULL;
        free(t -> next);

        return p;
}
```

delete 関数はある整数を queue の中に探し、見つければ出力し、そのアドレスを queue から削除する。

```
int delete(struct node *head, int val){
        struct node *s;
        struct node *t = head;

        if (t == NULL) return -1;

        //val is in head
        //求めた値は head にある場合、head -> next を head にする。head を
        free し、head -> val を出力する.
        if (t -> val == val){
```

```
                head = t -> next;
                free(t);
                return val;
        }

        while(t -> next != NULL && t -> val != val){
                //s は t の 1 個まえのアドレスを指す
                s = t;
                t = t -> next;
        }

        if (t -> val == val){
                //求めた値が見つければ、s -> next に t -> next を入れて、t
                を free、求めた値を出力として出す
                s -> next = t -> next;
                free(t);
                return val;
        } else return -1;               //見つからなければ、-1 を return する
}
```

display 関数は queue にある要素を末尾から先頭までの順場に表示する。

```
void display(struct node *head){
        struct node *t = head;

        if(head -> next == NULL){
                printf("%d ", head -> val);
        } else {
                //再起的に表示する
                display(head -> next);
                printf("%d ", head -> val);             //先頭を後ろに表示する
        }
}
```

```
int main(){
	struct node head = {-1, NULL};
	int nums[] = {0, 1, 2, 3, 4, 5, 43, 7, 8, 9};
	int i, res;

	for(i = 0; i < 10; i++){
		printf("put %d¥n", nums[i]);
		res = put(&head, nums[i]);
		if(res != 0) return 1;
	}

	printf("queue: ");
	display(&head);
	printf("¥n");

	for (i = 0; i < 3; i++){
		res = get(&head);
		printf("get %d¥n", res);
	}

	printf("queue: ");
	display(&head);
	printf("¥n");

	res = delete(&head, 6);
	printf("delete %d¥n", res);

	printf("queue: ");
	display(&head);
	printf("¥n");
}
```

main 関数では、以下のような例を実行した：

ソースコード	実行結果

struct node head = {-1, NULL};
int nums[] = {0, 1, 2, 3, 4, 5, 43, 7, 8, 9};
int i, res;
 	
for(i = 0; i < 10; i++){
　　printf("put %d¥n", nums[i]);
　　res = put(&head, nums[i]);
　　if(res != 0) return 1;
}
	put 0
	put 1
	put 2
	put 3
	put 4
	put 5
	put 43
	put 7
	put 8
	put 9

printf("queue: ");
display(&head);
printf("¥n");
	queue: 0 1 2 3 4 5 43 7 8 9

for (i = 0; i < 3; i++){
　　res = get(&head);
　　printf("get %d¥n", res);
}
	get 0
	get 1
	get 2

printf("queue: ");
display(&head);
printf("¥n");
	queue: 3 4 5 43 7 8 9

res = delete(&head, 6); printf("delete %d¥n", res);	delete -1

printf("queue: ");
display(&head);
printf("¥n");
	queue: 3 4 5 43 7 8 9

ソースコード	実行結果

struct node head = {-1, NULL};
int nums[] = {0, 1, 2, 3, 4, 5, 43, 7, 8, 9};
int i, res;
 	
for(i = 0; i < 10; i++){
　　printf("put %d¥n", nums[i]);
　　res = put(&head, nums[i]);
　　if(res != 0) return 1;
}
	put 0
	put 1
	put 2
	put 3
	put 4
	put 5
	put 43
	put 7
	put 8
	put 9

printf("queue: ");
display(&head);
printf("¥n");
	queue: 0 1 2 3 4 5 43 7 8 9

for (i = 0; i < 3; i++){
　　res = get(&head);
　　printf("get %d¥n", res);
}
	get 0
	get 1
	get 2

printf("queue: ");
display(&head);
printf("¥n");
	queue: 3 4 5 43 7 8 9

res = delete(&head, 7); printf("delete %d¥n", res);	delete 7

printf("queue: ");
display(&head);
printf("¥n");
	queue: 3 4 5 43 8 9