2025年度 プログラミングIII 第13回 レポート

学籍番号 36714137
山本賢人

2026年1月14日

1 はじめに

本レポートでは課題のプログラムの中身と実行結果について報告する。

2 演習課題

2.1 演習 13-1

与えられたスタックのプログラムを関数やヘッダーごとに分割し、Makefile を作成してコンパイルが行えるようにする。また、スタックにプッシュされた全要素を表示する機能を追加する。

まず、プログラムの分割から行う。main 関数を main13.c に配置し、ヘッダファイルを include する形で実装する。

Listing 1: main13.c の実装

```c
#include <stdio.h>
#include "task13.h"

int main(void) {
    Stack stk;

    if (StackAlloc(&stk, 100) == -1) {
        printf("スタックの作成に失敗しました\n");
        return 1;
    }

    while (1) {
        int m, x;
        printf("現在のデータ
            数:␣%d␣/␣%d\n", StackNo(&stk), StackSize(&stk));
        printf("(1)プッシュ␣(2)ポップ␣(3)表示␣(0)終了:␣");
        scanf("%d", &m);
        if (m == 0) {
            break;
        }

        switch (m) {
            case 1:
                printf("データ:␣");
                scanf("%d", &x);
                if (StackPush(&stk, x) == -1) {
                    printf("プッシュに失敗しました\n");
                }
                break;
            case 2:
                if (StackPop(&stk, &x) == -1) {
                    printf("ポップに失敗しました\n");
                }
                break;
            case 3:
                printf("スタックの中身:␣");
                for (int i = 0; i < StackNo(&stk); i++) {
```

```
                    printf("%d␣", stk.stk[i]);
                }
                printf("\n");
                break;
            default:
                break;
        }
    }

    StackFree(&stk);

    return 0;
}
```

case 3 の部分がスタック全表示の新規実装部分である。StackNo 関数で現在のスタックポインタ位置を取得し、その数だけループを回して全要素を表示している。

次に、ヘッダファイルの実装を行う。Stack 構造体の定義と関数のプロトタイプ宣言を記述する。

Listing 2: task13.h の実装

```
#ifndef TASK13_H
#define TASK13_H

typedef struct {
    int max;
    int ptr;
    int *stk;
} Stack;

int StackAlloc(Stack *s, int max);
int StackFree(Stack *s);
int StackPush(Stack *s, int x);
int StackPop(Stack *s, int *x);
int StackPeek(const Stack *s, int *x);
int StackSize(const Stack *s);
int StackNo(const Stack *s);
int StackIsEmpty(const Stack *s);
int StackIsFull(const Stack *s);
int StackClear(Stack *s);

#endif
```

次に、スタック操作関数の実装を行う。これらは与えられたコードをそのまま分割したものである。

Listing 3: task13.c の実装

```
#include <stdio.h>
#include <stdlib.h>
#include "task13.h"

int StackAlloc(Stack *s, int max) {
    s->ptr = 0;
    if ((s->stk = calloc(max, sizeof(int))) == NULL) {
        s->max = 0;
```

```
9           return -1;
10      }
11      s->max = max;
12      return 0;
13  }
14
15  int StackFree(Stack *s) {
16      if (s->stk != NULL) {
17          free(s->stk);
18          s->max = s->ptr = 0;
19      }
20      return 0;
21  }
22
23  int StackPush(Stack *s, int x) {
24      if (s->ptr >= s->max) {
25          return -1;
26      }
27      s->stk[s->ptr++] = x;
28      return 0;
29  }
30
31  int StackPop(Stack *s, int *x) {
32      if (s->ptr <= 0) {
33          return -1;
34      }
35      *x = s->stk[--s->ptr];
36      return 0;
37  }
38
39  int StackPeek(const Stack *s, int *x) {
40      if (s->ptr <= 0) {
41          return -1;
42      }
43      *x = s->stk[s->ptr - 1];
44      return 0;
45  }
46
47  int StackSize(const Stack *s) {
48      return s->max;
49  }
50
51  int StackNo(const Stack *s) {
52      return s->ptr;
53  }
54
55  int StackIsEmpty(const Stack *s) {
56      return s->ptr <= 0;
57  }
58
59  int StackIsFull(const Stack *s) {
60      return s->ptr >= s->max;
61  }
62
63  int StackClear(Stack *s) {
```

```
64      s->ptr = 0;
65      return 0;
66  }
```

最後に、これらをコンパイルするための Makefile を実装する。

Listing 4: Makefile の実装

```
1   # Makefile
2   # サフィックスルールによる依存関係
3
4   CC = gcc
5   TARGET = task13
6   HEADER = task13.h
7
8   SRCS = main13.c task13.c
9   OBJS = main13.o task13.o
10
11  # サフィックスルール
12  .SUFFIXES: .c .o
13  .c.o:
14          $(CC) -c $<
15
16  # デフォルトターゲット
17  all: $(TARGET)
18
19  # 実行ファイル生成
20  $(TARGET): $(OBJS)
21          $(CC) -o $@ $(OBJS)
22
23  # 依存関係
24  main13.o: main13.c $(HEADER)
25  task13.o: task13.c $(HEADER)
26
27  # クリーン
28  clean:
29          rm -f $(OBJS) $(TARGET)
30
31  # 実行
32  run: $(TARGET)
33          ./$(TARGET)
34
35  .PHONY: all clean run
```

サフィックスルールを用いることで、.c ファイルから.o ファイルへの変換を簡潔に記述できる。

これは、過去のレポートで作成した Makefile を流用して作成している。

実行結果:

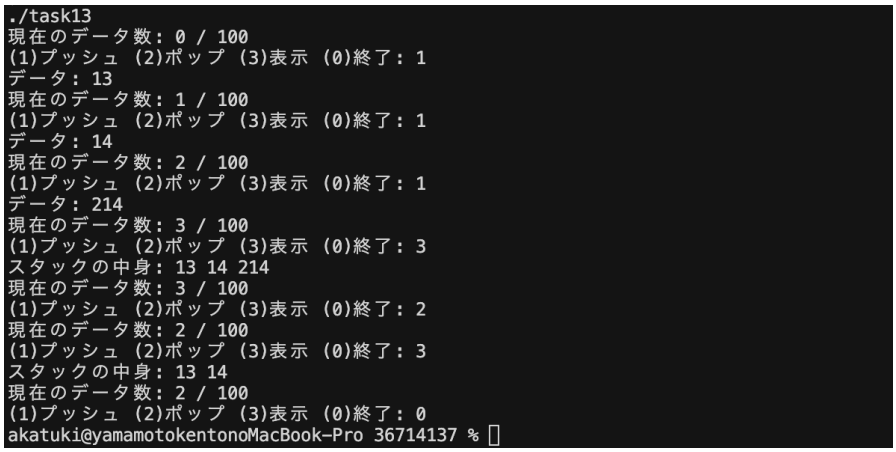

図 1: 演習 13-1 の実行結果

プッシュ、ポップ、全表示の各機能が正しく動作していることが確認できる。