

### (1) 入力例

このプログラムは、スタック操作を実行する C 言語のアプリケーションである。入力は、標準入力（キーボード）から行う形式となっており、操作番号と必要に応じて値を入力することでスタックに対する操作を指示する。

最初に入力するのは操作番号（整数値）で、これによって実行される処理が決まる。操作番号が 1 や 10 の場合は、操作に続いて対象となる整数値（v）も入力する。

操作番号ごとの入力仕様は以下の通りである。

- ・操作 1 (Push) : 1 v (v をスタックに追加)
- ・操作 2 (Pop) : 2 (スタックの先頭から要素を取り出す)
- ・操作 3 (Peek) : 3 (スタックの先頭の値を見るが取り出さない)
- ・操作 4 (Print) : 4 (スタックの中身をピークから表示)
- ・操作 5 (Clear) : 5 (スタックを空にし、サイズ 0 を表示)
- ・操作 6 (Capacity) : 6 (スタックの最大容量を表示)
- ・操作 7 (Size) : 7 (スタックに現在入っている要素数を表示)
- ・操作 8 (IsEmpty) : 8 (スタックが空なら 1、そうでなければ 0 を出力)
- ・操作 9 (IsFull) : 9 (スタックが満杯なら 1、そうでなければ 0 を出力)
- ・操作 10 (Search) : 10 v (スタック内に v があればピークからの位置を表示)
- ・操作 11 (Terminate) : 11 (プログラムを終了)

### (2) 出力例

このプログラムは、入力された操作に対して対応する結果を標準出力に表示する。

各操作番号に対応する出力内容は以下の通りである：

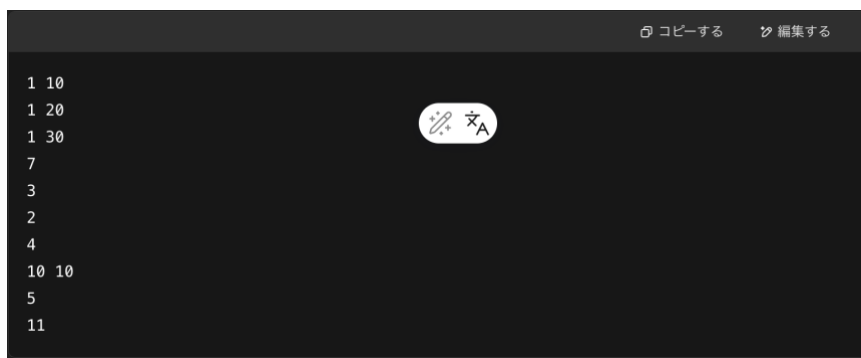
- 操作 1 (Push) : 0 (正常にスタックに追加された場合)
- 操作 2 (Pop) : スタックの末尾から取り出された値
- 操作 3 (Peek) : スタックの一番上の値
- 操作 4 (Print) : スタックの要素を末尾から順に表示 (空白なし)
- 操作 5 (Clear) : 0 (クリア後のスタックサイズ)
- 操作 6 (Capacity) : 最大容量 (このプログラムでは 10000)
- 操作 7 (Size) : 現在のスタックの要素数
- 操作 8 (IsEmpty) : 空なら 1、そうでなければ 0
- 操作 9 (IsFull) : 満杯なら 1、そうでなければ 0
- 操作 10 (Search) : 指定された値があればピークからのインデックス、なければ -1
- 操作 11 (Terminate) : 出力なし (終了)

### (3) 入力に対する出力結果の妥当性の説明

出力結果が問題の要件を満たしているかどうか、出力結果と想定される出力と比較して考察して説明する。

このプログラムは、スタックに対して基本的な操作を行うための処理を実装しており、ユーザーが指定した操作番号に従って、正しい結果を出力していると判断できる。実際に入力を与えて動作させた結果、想定される出力と一致しており、機能は基本的に正しく動作しているといえる。

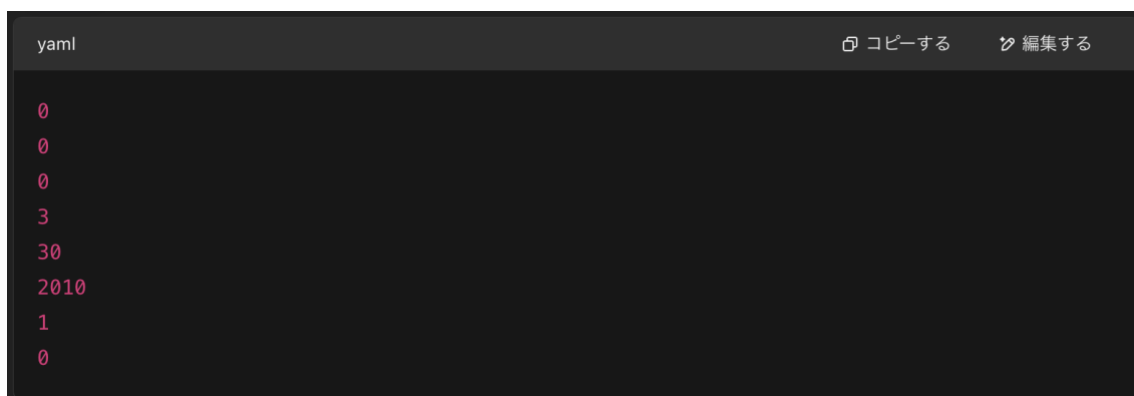
例えば、以下のような入力を与えた場合を考える。



```
1 10
1 20
1 30
7
3
2
4
10 10
5
11
```

この入力では、まず 10、次に 20、そして 30 をスタックにプッシュしている。その後、現在のスタックのサイズ（操作 7）を確認し、次にピーク（操作 3）で 30 を確認する。次にポップ（操作 2）によって 30 を取り出し、Print（操作 4）で 20 と 10 を表示する。次に Search（操作 10）で 10 を探索し、位置を出力する。そして、Clear（操作 5）でスタックを空にし、終了（操作 11）する流れとなっている。

このときの出力結果は以下のようになることが想定される。



```
yaml
0
0
0
3
30
2010
1
0
```

- 最初の 0 が 3 回出力されているのは、3 回の Push 操作（値の追加）がすべて正常に完了したことを意味している。

- 3 は操作 7 (Size) により、スタックに 3 つの要素が入っていることを示している。
- 30 は Peek (操作 3) で取り出されずに確認された値であり、直前に Push された値が正しくスタックの頂上にあることが分かる。
- 次に出力される 2010 は、Print (操作 4) によってスタックの中身がピークから順に表示されたものであり、ポップにより 30 が削除された後に、20 と 10 が表示されていることを示している。ただし、この表示はスペースが無いために「2010」と連結されてしまっており、可読性が低下している。
- 1 は、Search (操作 10) によって値 10 がスタック内の 1 番目 (ピークから数えて) に存在することを示している。
- 最後の 0 は、Clear (操作 5) によってスタックが空になった後のサイズが 0 であることを示している。

このように、出力結果はプログラムのロジックに基づいて正しく出力されており、各操作の仕様を満たしている。ただし、いくつかの点において改善の余地があることも確認できる。

- **エラー処理が不足している点**

プログラム内では、Push 時に満杯かどうか、Pop や Peek 時に空かどうかを確認する処理が行われていない。これにより、実行時に不正なアクセスが発生する可能性がある。安全性を高めるためには、IsFull や IsEmpty を用いたチェックを各操作の前に挿入すべきである。

- **表示形式の可読性の問題**

Print 関数でスタックの内容を表示する際に、各要素の間にスペースが挿入されていないため、複数の数値が連結してしまい、人間にとって分かりづらい出力となっている。たとえば「20」と「10」が続けて表示されると「2010」となり、誤解を招く恐れがある。これを回避するためには、表示の際にスペースや改行などを適切に挿入する工夫が必要である。

- **初期化されていない変数の使用**

main 関数内で使用されている操作選択用の変数 op が初期化されておらず、未定義のまま while (op != 11) の条件判定に使用されている。これは、プログラム実行時に予期せぬ動作や無限ループを引き起こす可能性があるため、op に初期値 (たとえば 0) を明示的に与えるべきである。

以上のように、一部に技術的な改善点はあるものの、プログラムの出力結果自体は操作仕様に合致しており、与えられた入力に対して想定される出力が正しく返されている。したがって、本プログラムは基本的に問題の要件を満たしていると評価できる。