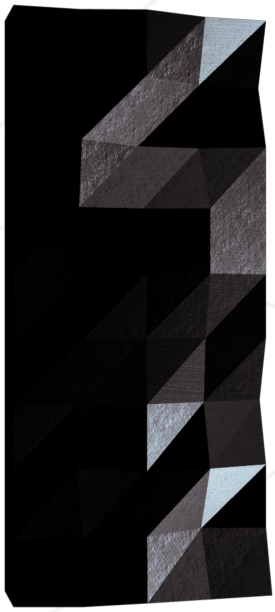

Push_swap

Çünkü Swap_push o kadar doğal değil

Özet:
Bu proje size mümkün olan en düşük sayıda hamleyi yaparak ve sınırlı bir talimat seti kullanarak bir yığındaki verileri sıralatacak. Başarabilmek için çeşitli tiplerdeki algoritmaları manipüle edip optimal bir veri sıralaması için en uygun olanını(birçoğu arasından) seçmeniz gerekecek.

Versiyon: 6

İçindekiler

Bölüm I

Foreword

- C

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

- ASM

```
cseg segment
assume cs:cseg, ds:cseg
org 100h
main proc
jmp debut
mess db 'Hello world!$'
debut:
mov dx, offset mess
mov ah, 9
int 21h
ret
main endp
cseg ends
end main
```

- LOLCODE

```
HAI
CAN HAS STDIO?
VISIBLE "HELLO WORLD!"
KTHXBYE
```

- PHP

```
<?php
 echo "Hello world!";
?>
```

- BrainFuck

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]
>++.>+.+++++++..+++.>++.
<<++++++++++++++.>.+++.------.--------.>+.>.
```

- C#

```
using System;

public class HelloWorld {
   public static void Main () {
      Console.WriteLine("Hello world!");
   }
}
```

- HTML5

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Hello world !</title>
   </head>
   <body>
      <p>Hello World !</p>
   </body>
</html>
```

- YASL

```
"Hello world!"
print
```

- OCaml

```
let main () =
   print_endline "Hello world !"

let _ = main ()
```

Bölüm II

Giriş

`Push_swap` projesi çok basit ve etkili bir algoritma projesidir: verileri sıralamak. Elinizde bir dizi integer, 2 yığın ve bu yığınları manipüle etmek için bir talimat setiniz var.

Siz ne mi yapacaksınız? C dilinde argüman olarak alınan integerları sıralayan `Push_swap` talimatlarını kullanarak en kısa programı hesaplayan ve bastıran `push_swap` programını yazacaksınız..

Kolay mı?

Bunu göreceğiz...

Bölüm III

Hedefler

Bir coder'ın hayatında sıralama algoritması yazmak her zaman çok önemli bir adımdır, çünkü genellikle karmaşıklık kavramı ile ilk karşılaşılan yer burasıdır.

Sıralama algoritmaları ve karmaşıklıkları, iş görüşmeli sırasında sorulan klasik soruların bir parçasıdır. Şu an bu kavramlara bakmak için iyi bir zaman çünkü bir noktada onlarla yüzleşmeniz gerekecek.

Bu projenin öğrenme hedefleri `C` ve temel algoritmaların titizlikle kullanılmasıdır. Bu basit algoritmaların karmaşıklıklarını özellikle incelemeniz gerek.

Değerleri sıralamak kolaydır. Onları en hızlı şekilde sıralamak ise daha az kolaydır, En verimli algoritma integer grubundan grubuna farklılık gösterebilir.

Bölüm IV

Genel Talimatlar

- Bu proje sadece insanlar tarafından kontrol edilecektir. Bu yüzden dosyalarınızı istediğiniz gibi isimlendirmekte özgürsünüz, yine de aşağıda dikkat etmeniz gereken bazı noktalardan bahsettik.
- Çalıştırılacak dosya `push_swap` şeklinde isimlendirilmelidir.
- Bir `Makefile` hazırlamalısınız. Bu `Makefile`'ın projeyi derlemesi ve her zaman kullanılan kuralları içermesi gerekmektedir. Sadece gerekli olduğunda tekrar derleme yapmalıdır.
- Eğer zekiyseniz, projenizde kendi oluşturduğunuz kütüphaneyi kullanırsınız, ve aynı zamanda repository'nizin kök dizinindeki kendi `Makefile` 'ına sahip olan `libft` klasörünü de teslim edersiniz. `Makefile` dosyanız önce kütüphaneyi sonra da projenizi derleyecektir.
- Global değişken kullanımı yasaktır.
- Projeniz `C` dilinde Norm'a uygun olacak şekilde yazılmalıdır.
- Oluşabilecek hataları hassas bir şekilde ele almalısınız. Program beklenmedik bir hatayla kapanmamalıdır (Segmentation fault, bus error, double free, vb.).
- Program aynı zamanda `memory leaks`'e sahip olmamalıdır.
- Zorunlu kısımda aşağıdaki fonksiyonları kullanmanıza izin vardır:
  - `write`
  - `read`
  - `malloc`
  - `free`
  - `exit`
- Sorularınızı forum & Slack üzerinden sorabilirsiniz...

Bölüm V

Zorunlu kısım

V.1 Oyun kuralları

- Oyun `a` ve `b` isimli 2 yığın'dan oluşmaktadır.
- Başlangıç olarak:
  - `a` yığını rastgele sayıda birbirinin kopyası olmayan negative ve/veya pozitif sayıdan oluşmaktadır.
  - `b` yığını boştur.
- Amaç yığındaki sayıları artan şekilde `a` yığınında sıralamaktır.
- Bunu yapmak için aşağıdaki işlemleri kullanabilirsiniz:

sa : `swap a` -İlk 2 elementi `a` yığınının en üstüne çıkarır. Bir veya daha az eleman varsa hiçbir şey yapmaz.

sb : `swap b` -İlk 2 elementi `b` yığınının en üstüne çıkarır. Bir veya daha az eleman varsa hiçbir şey yapmaz.

ss : `sa` ve `sb` aynı anda kullanılır.

pa : `push a` - `b` yığınının en üstteki ilk elemanını `a` yığının en üstüne yerleştirir. `b` boşsa hiçbir şey yapmaz.

pb : `push b` -`a` yığınının en üstteki ilk elemanını `b` yığının en üstüne yerleştirir. `a` boşsa hiçbir şey yapmaz.

ra : `rotate a` - `a` yığınının tüm elemanlarını 1 üste taşır. İlk eleman son eleman haline gelir.

rb : `rotate b` - `b` yığınının tüm elemanlarını 1 üste taşır. İlk eleman son eleman haline gelir.

rr : `ra` ve `rb` aynı anda kullanılır.

rra : `reverse rotate a` - `a` yığınının tüm elemanlarını 1 alta taşır. Son eleman ilk eleman haline gelir.

rrb : `reverse rotate b` - `b` yığınının tüm elemanlarını 1 alta taşır. Son eleman ilk eleman haline gelir.

rrr : `rra` ve `rrb` aynı anda kullanılır.

V.2 Örnek

Bazı talimatların çalışmasını göstermek için rastgele bir integer listesini sıralayalım. Bu örnekte iki yığının da sağ taraftan büyüdüğünü düşüneceğiz..

```
---------------------------------------------------------------------------------------------------------
Init a and b:
2
1
3
6
5
8

- -
a b
---------------------------------------------------------------------------------------------------------
Exec sa:
1
2
3
6
5
8

- -
a b
---------------------------------------------------------------------------------------------------------
Exec pb pb pb:
6 3
5 2
8 1

- -
a b
---------------------------------------------------------------------------------------------------------
Exec ra rb (equiv. to rr):
5 2
8 1
6 3

- -
a b
---------------------------------------------------------------------------------------------------------
Exec rra rrb (equiv. to rrr):
6 3
5 2
8 1

- -
a b
---------------------------------------------------------------------------------------------------------
Exec sa:
5 3
6 2
8 1

- -
a b
---------------------------------------------------------------------------------------------------------
Exec pa pa pa:
1
2
3
5
6
8

- -
a b
---------------------------------------------------------------------------------------------------------
```

Bu örnekte `a` yığını 12 talimat kullanılarak sıralanmaktadır. Daha iyisini yapabilir misin?

V.3 'push_swap" programı

- Integer listesi olarak biçimlendirilmiş bir yığını argüman olarak alacak bir `push_swap` programı yazmalısınız. İlk eleman yığının en üstünde olmalıdır. (sıralamaya dikkat edin).
- Program `a` yığınını sıralayabilen en küçük talimat listesini en küçük sayı en üstte olacak şekilde bastırmalıdır.
- Talimatlar sadece ve sadece '\n' ile ayrılmalıdır.
- Amaç yığını mümkün olan en az sayıda hamleyle sıralamak. Değerlendirilme sırasında programınızın bulduğu talimat sayısıyla kabul edilen maksimum hamle sayısı karşılaştırılacak. Eğer programınız çok büyük bir liste veya doğru sıralanmamış bir liste basıyorsa, puan alamazsınız.
- Eğer parametre belirlenmemişse program hiçbir şey bastırmamalı ve bitmelidir.
- Hata alma durumunda terminale '\n' ile takip edilen bir `Error` yazısı bastırmalısınız. Bazı hatalar: tüm argümanlar integer değil, bazı argümanlar integerdan büyük, ve/veya bazı kopyalar mevcut.

```
$>./push_swap 2 1 3 6 5 8
sa
pb
pb
pb
sa
pa
pa
pa
$>./push_swap 0 one 2 3
Error
$>
```

Değerlendirilme sırasında programınızı düzgünce test etmek için bir binary sağlayacağız. Şu şekilde çalışacak:

```
$>ARG="4 67 3 87 23"; ./push_swap $ARG | wc -l
      6
$>ARG="4 67 3 87 23"; ./push_swap $ARG | ./checker_OS $ARG
OK
$>
```

`checker_OS KO` bastırırsa, `push_swap` integerları sıralamayan bir talimat seti kullanmıştır. `checker_OS` programı intranette proje sayfasında bulunabilir. Bonus kısımda nasıl çalıştığı hakkında bir açıklama bulabilirsiniz.

Bölüm VI

Bonus kısım

Bonus kısım, yalnızca zorunlu kısım MÜKEMMEL ise değerlendirilecektir. Bu zorunlu kısmın baştan sona tamamlanıp tüm kontrollerin kusursuzca yapılması anlamına gelir. IEğer bu koşul sağlanmamışsa bonuslarınız değerlendirilmeye alınmayacaktır.

`Push_swap` basitliğinden dolayı bir bonus kısmını hak ediyor gibi görünüyor. Kendi checker programınızı yazmaya ne dersiniz?

VI.1 “checker” programı

- Integer listesi şeklinde ayarlanmış `a` yığınını argüman olarak alan bir `checker` programı yazın. İlk eleman yığının en üstünde olmalıdır. (sıralamaya dikkat edin). Eğer hiç argüman yoksa `checker` durmalı ve hiçbir şey bastırmamalıdır.
- `checker` çıktıdaki ’\n’ le takip edilen talimatları okuyacak. Tüm talimatlar okunduğunda, `checker` argüman olarak alınan stack üzerinde işlemleri gerçekleştirecek.
- Talimatlar gerçekleştirildikten sonra `a` yığını düzgünce sıralanmış ve `b` yığını boşsa checker ’\n’ ile takip edilen `"OK"` yazısını çıktı olarak basmalıdır. Diğer tüm koşullarda `checker` ’\n’ ile takip edilen `"OK"` yazısını çıktı olarak basmalıdır..
- Hata alma durumunda terminale ’\n’ ile takip edilen bir `Error` yazısı bastırmalısınız. Bazı hatalar: tüm argümanlar integer değil, bazı argümanlar integerdan büyük, ve/veya bazı kopyalar mevcut, talimat mevcut değil ya da yanlış şekilde yazılmış.

`checker programı sayesinde programınızın sayıları doğru şekilde sıralayıp sıralamadığını öğrenebileceksiniz.`

```
$>./checker 3 2 1 0
rra
pb
sa
rra
pa
OK
$>./checker 3 2 1 0
sa
rra
pb
KO
$>./checker 3 2 one 0
Error
$>./checker "" 1
Error
$>
```

`Oluşacak durumları bizim verdiğimiz binary ile birebir olacak şekilde İNCELEMEYİN. Hataları halletmeniz zorunlu fakat argümanları ne şekilde ayrıştıracağınıza siz karar vereceksiniz.`

Bölüm VII

Teslim etme ve değerlendirme

Çalışmanızı her zaman olduğu gibi `GiTss` repository'nize yapın . Sadece repository'nizdeki çalışma notlandırılacak.

Herkese bol şans!