Este é o CS50

Introdução à Ciência da Computação (CS50)

OpenCourseWare

Doar (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Pluralidade

Problema a resolver

As eleições podem ter diversos formatos e tamanhos. No Reino Unido, o Primeiro-Ministro (https://www.parliament.uk/education/about-your-parliament/general-elections/) é oficialmente nomeado pelo monarca, que geralmente escolhe o líder do partido político que conquista o maior número de cadeiras na Câmara dos Comuns. Os Estados Unidos utilizam um sistema eleitoral com várias etapas, o Colégio Eleitoral (https://www.archives.gov/federal-register/electoral-college/about.html) , no qual os cidadãos votam em como cada estado deve alocar os eleitores que, por sua vez, elegem o Presidente.

Talvez a maneira mais simples de realizar uma eleição seja por meio de um método conhecido como "voto majoritário simples" (também chamado de "voto único" ou "voto de maioria simples"). No voto majoritário simples, cada eleitor vota em apenas um candidato. Ao final da eleição, o candidato que obtiver o maior número de votos é declarado o vencedor.

Para este problema, você deverá implementar um programa que execute uma eleição por maioria simples, conforme o exemplo abaixo.

Demonstração

```
Alice
$ ./plurality Alice Bob Charlie
Number of voters: 3
Vote: Alice
Vote: Bob
Vote: Charlie
Alice
Bob
Charlie
$ ./plurality
Usage: plurality [candidate ...]
$
```

Recorded with **asciinema**

Código de Distribuição

Para este problema, você deverá estender a funcionalidade do "código de distribuição" fornecido pela equipe do CS50.

▼ **Baixe o código de distribuição.**

Faça login em cs50.dev (https://cs50.dev/) , clique na janela do terminal e execute `cd` o comando. Você verá que o prompt do terminal ficará semelhante ao seguinte:

```
$
```

Em seguida, execute

```
wget https://cdn.cs50.net/2024/fall/psets/3/plurality.zip
```

para baixar um arquivo ZIP chamado `plurality.zip` para o seu espaço de código.

Em seguida, execute

```
unzip plurality.zip
```

para criar uma pasta chamada `plurality` . Você não precisa mais do arquivo ZIP, então pode executar

```
rm plurality.zip
```

e responda com "y" seguido de Enter quando solicitado para remover o arquivo ZIP que você baixou.

Agora digite

```
cd plurality
```

Em seguida, pressione Enter para entrar (ou seja, abrir) esse diretório. Seu prompt agora deve ser semelhante ao abaixo.

```
plurality/ $
```

Se tudo correr bem, você deverá executar

```
ls
```

e veja um arquivo chamado `plurality.c`. A execução `code plurality.c` deve abrir o arquivo onde você digitará o código para este conjunto de problemas. Caso contrário, refaça seus passos e veja se consegue determinar onde errou!

▼ **Entenda o código em `plurality.c`**

Sempre que você for ampliar a funcionalidade de um código existente, é melhor ter certeza de que primeiro o entende em seu estado atual.

Observe primeiro o início do arquivo. A linha `#define MAX 9` é uma sintaxe usada aqui para indicar que `MAX` é uma constante (igual a `9`) que pode ser usada em todo o programa. Aqui, representa o número máximo de candidatos que uma eleição pode ter.

```
// Max number of candidates
#define MAX 9
```

Observe que, `plurality.c` em seguida, essa constante é usada para definir um array global — ou seja, um array que qualquer função pode acessar.

```
// Array of candidates
candidate candidates[MAX];
```

Mas, neste caso, o que é um `a` `candidate`? Em `plurality.c`, um `a` `candidate` é um `a` `struct`. Cada `a` `candidate` possui dois campos: um `string` chamado `a` `name`, que representa o nome do candidato, e um `int` chamado `b`, `votes` que representa o número de votos que o candidato recebeu.

```
// Candidates have name and vote count
typedef struct
{
    string name;
    int votes;
}
candidate;
```

Agora, observe a `main` função em si. Veja se consegue encontrar onde o programa define uma variável global `candidate_count` que representa o número de candidatos na eleição.

```
// Number of candidates
int candidate_count;
```

E quanto à parte em que ele copia os argumentos da linha de comando para a matriz `candidates`?

```
// Populate array of candidates
candidate_count = argc - 1;
if (candidate_count > MAX)
{
    printf("Maximum number of candidates is %i\n", MAX);
    return 2;
}
for (int i = 0; i < candidate_count; i++)
{
    candidates[i].name = argv[i + 1];
    candidates[i].votes = 0;
}
```

E onde pede ao usuário para digitar o número de eleitores?

```
int voter_count = get_int("Number of voters: ");
```

Em seguida, o programa permite que cada eleitor digite seu voto, chamando a `vote` função para cada candidato votado. Finalmente, `main` chama a `print_winner` função para imprimir o(s) vencedor(es) da eleição. Deixaremos para você identificar o código que implementa essa funcionalidade.

Se você olhar mais adiante no arquivo, porém, perceberá que as funções `vote` `and` `print_winner` foram deixadas em branco.

```
// Update vote totals given a new vote
bool vote(string name)
{
    // TODO
    return false;
}

// Print the winner (or winners) of the election
void print_winner(void)
{
    // TODO
    return;
}
```

Esta parte é de sua responsabilidade! **Você não deve modificar nada `plurality.c` além das implementações das `vote` funções `print_winner` (e a inclusão de arquivos de cabeçalho adicionais, se desejar).**

Dicas

Clique nos botões abaixo para ler algumas dicas!

▼ **Complete a `vote`função**

Em seguida, complete a `vote` função.

- Considere que `vote` a assinatura de 's, `bool vote(string name)` , mostra que ela recebe um único argumento, `string` chamado `name` , representando o nome do candidato que recebeu votos.
- `vote` deve retornar um `bool` , onde `true` indica que o voto foi computado com sucesso e `false` indica que não foi.

Uma forma de abordar esse problema é fazer o seguinte:

1. Itere sobre cada candidato.
    1. Verifique se o nome do candidato corresponde ao que foi digitado. `name`
        1. Em caso afirmativo, incremente os votos desse candidato e retorne. `true`
        2. Caso contrário, continue verificando.
2. Se não houver correspondências após verificar cada candidato, retorne. `false`

Vamos escrever um pseudocódigo para te lembrar de fazer exatamente isso:

```
// Update vote totals given a new vote
bool vote(string name)
{
    // Iterate over each candidate
        // Check if candidate's name matches given name
            // If yes, increment candidate's votes and return true

    // If no match, return false
}
```

No entanto, deixaremos a implementação em código por sua conta!

▼ **Complete a `print_winner`função**

Por fim, complete a `print_winner` função.

- A função deve imprimir o nome do candidato que recebeu o maior número de votos na eleição e, em seguida, imprimir uma nova linha.
- A eleição pode terminar em empate se vários candidatos obtiverem o número máximo de votos. Nesse caso, você deve exibir o nome de cada um dos candidatos vencedores, um em uma linha separada.

Você pode pensar que um algoritmo de ordenação seria a melhor solução para este problema: imagine ordenar os candidatos pelo total de votos e imprimir o(s) candidato(s) mais votado(s). Lembre-se, porém, que a ordenação pode ser custosa: mesmo o Merge Sort, um dos algoritmos de ordenação mais rápidos, tem complexidade computacional de O(n log n).$O(N\ log(N))$.

Considere que você precisa apenas de duas informações para resolver este problema:

1. O número máximo de votos
2. O(s) candidato(s) com esse número de votos

Assim sendo, uma boa solução pode exigir apenas duas buscas. Escreva um pseudocódigo para se lembrar de fazer exatamente isso:

```
// Print the winner (or winners) of the election
void print_winner(void)
{
    // Find the maximum number of votes

    // Print the candidate (or candidates) with maximum votes

}
```

Mas vamos deixar o código por sua conta!

Passo a passo

Como testar

Certifique-se de testar seu código para garantir que ele funcione corretamente...

- Uma eleição com qualquer número de candidatos (até o limite `MAX` de `9`)
- Votar em um candidato pelo nome.
- Votos inválidos para candidatos que não constam na cédula eleitoral.
- Imprimir o vencedor da eleição, caso haja apenas um.
- Imprimir o nome do vencedor da eleição, caso haja mais de um vencedor.

Correção

```
check50 cs50/problems/2025/x/plurality
```

Estilo

```
style50 plurality.c
```

Como enviar

```
submit50 cs50/problems/2025/x/plurality
```