<Yourein/string>

pinput

Pythonの標準入力にならった関数です。 getlineなどで入力した文字列をpinputに、spliterを指定して投げることで、分割された文字列がvector<string>型でreturnされます。 ただし、第一引数にistream型(`std::cin`等)を指定してはいけません。

```
std::vector<std::string> pinput(std::string s, char spliter)
```

例えば、以下のように使うことができます。

```
std::string sentence = "Hello, World!";
auto v = Yourein::strs::pinput(sentence, ',');

for (auto x : v){
    std::cout << x << endl;
}
```

このコードの出力結果は以下のようになります。

```
Hello
World!
```

この関数では先頭の空文字列(文字列の先頭に位置する空白文字のみの部分文字列)は削除されます。 そのため、上の出力結果では `World!` の前に本来存在する空白文字が削除されています。

strcount

Pythonの文字列型のメンバ関数 `.count(str)` にならった関数です。

文字列sの中に文字列counterが何度登場するかをカウントします。

```
int strcount(std::string s, std::string counter)
```

```
std::string s = "Hello, There, Hello, Hello";
std::cout << Youdain::strs::strcount(s, "Hello") << std::endl;
```

出力結果は以下の通りです。

```
3
```

両引数がstd::string型であるので、引数に変数を渡さない限りダブルクオーテーションを忘れないでください。

また、この関数は計算量 $O(N)$ となるので、$|S| \leq 5 \times 10^7$ 以下であることを想定しています。 加えて、std::wstring型には対応していません。(今後もしかしたらTemplateで作るかもしれません。)

sfind

```
string::find
```

の劣化関数です。 前者が文字列の発見位置をreturnするのに対し、この関数はそもそも存在するか否かを判定する関数です。

```
bool sfind(std::string s, std::string finder)
```

このように定義され、

```
std::string s1 = "I don't want to be just another";
std::cout << Yourein::strs::sfind(s1, "want") << std::endl;
std::cout << Yourein::strs::sfind(s1, "wanna") << std::endl;
```

以下のように動作します。

```
1
0
```

文字列sの中で文字列finderを発見したときは`true`を 見つからなかったときは`false`を返します。

また、この関数の計算量は最低$O(N)$、最悪$O(2N)$になります。 両者ともに線形時間ですが、`strcount`関数同様、$|S| \leq 5 \times 10^7$ の制約下での使用を想定しています。