

SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
					4	1		

O objetivo deste exercício é construir um resolveur de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesma para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num labirinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força

bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretendamos colocar nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna Verdadeiro sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna falso. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for(j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
```

```
if (grade[x][col]==k){return 0;}
}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
for (i=0;i<3;i++){
for(j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9}};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

📖 Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

	8			5				
				3	4	5	7	
				7		8		9
	6		4			9		3
		7		1		5		
4		8		7			2	
9		1		2				
8	4	2	3					
			1				8	

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
7,2	6,9	4,1

Responda aqui:

1	2	3
---	---	---



SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
					4	1		

O objetivo deste exercício é construir um resolvidor de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesma para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num laberinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretendidamente vai ser colocado nessa linha, coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna Verdadeiro sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna falso. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for (j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
if (grade[x][col]==k){return 0;}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
```

```
for (i=0;i<3;i++){
for (j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

3	6	1		2	5	9		
	8		9	6			1	
4							5	7
		8				4	7	1
			6		3			
2	5	9				8		
7	4							5
	2			1	8		6	
		5	4	7		3	2	9

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
5,8	4,4	6,8
Responda aqui:		
1	2	3



204-75615 - n lim

SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
					4	1		

O objetivo deste exercício é construir um resolveur de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesmo para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num labirinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força

bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretendamos colocar nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna Verdadeiro sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna falso. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for (j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
```

```
if (grade[x][col]==k){return 0;}
}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
for (i=0;i<3;i++){
for (j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9}};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

🔗 Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

3	6			2			8	9
			3	6	1			
8		3				6		2
4			6		3			7
6		7				1		8
			4	1	8			
9	7			3			1	4

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
7,2	9,4	7,9
Responda aqui:		
1	2	3



204-75639 - n lim

SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
					4	1		

O objetivo deste exercício é construir um resolvidor de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesma para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num labirinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força

bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretendamos colocar nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna **Verdadeiro** sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna **falso**. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for (j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
```

```
if (grade[x][col]==k){return 0;}
}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
for (i=0;i<3;i++){
for (j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9}};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

		3		2		6		
9			3		5			1
		1	8		6	4		
		8	1		2	9		
7								8
		6	7		8	2		
		2	6		9	5		
8			2		3			9
		5		1		3		

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
4,1	5,8	9,4
Responda aqui:		
1	2	3



204-75646 - n lim

SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
				4	1			

O objetivo deste exercício é construir um resolvidor de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesma para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num laberinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretensamente vai ser colocado nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna Verdadeiro sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna falso. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for (j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
if (grade[x][col]==k){return 0;}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
```

```
for (i=0;i<3;i++){
for (j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

		6		8		3		
	4	9		7		2	5	
			4		5			
6			3	1	7			4
		7				8		
1			8	2	6			9
			7		2			
	7	5		4		1	9	
		3		9		6		

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
7,3	9,6	6,3
Responda aqui:		
1	2	3



204-75691 - n lim

SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
				4	1			

O objetivo deste exercício é construir um resolveur de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesmo para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num labirinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força

bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretendamos colocar nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna Verdadeiro sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna falso. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for (j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
```

```
if (grade[x][col]==k){return 0;}
}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
for (i=0;i<3;i++){
for (j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9}};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

📖 Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

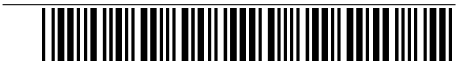
		3		2		6		
9			3		5			1
		1	8		6	4		
		8	1		2	9		
7								8
		6	7		8	2		
		2	6		9	5		
8			2		3			9
		5		1		3		

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
9,6	6,5	9,8

Responda aqui:

1	2	3
---	---	---



204-75703 - n lim

SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
					4	1		

O objetivo deste exercício é construir um resolveur de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesmo para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num labirinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força

bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretendamos colocar nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna Verdadeiro sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna falso. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for(j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
```

```
if (grade[x][col]==k){return 0;}
}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
for (i=0;i<3;i++){
for(j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9}};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

	5	3				7	9	
		9	7	5	3	4		
1								2
	9			8			1	
			9		7			
	8			3			7	
5								3
		7	6	4	1	2		
	6	1				9	4	

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
7,4	6,4	4,3
Responda aqui:		
1	2	3



SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
					4	1		

O objetivo deste exercício é construir um resolveur de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesmo para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num labirinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força

bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretendamos colocar nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna Verdadeiro sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna falso. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for(j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
```

```
if (grade[x][col]==k){return 0;}
}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
for (i=0;i<3;i++){
for(j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9}};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

📖 Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

	2		8	1		7	4	
7					3	1		
	9				2	8		5
		9		4			8	7
4			2		8			3
1	6			3		2		
3		2	7				6	
		5	6					8
	7	6		5	1		9	

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
5,8	7,7	6,6
Responda aqui:		
1	2	3



204-75653 - n lim

vivxj39a V: 1.02
75765 LUCAS ANTOSZCZYSZEN (75765);
JOAO BONFIN LINO (75622);
24JOG204 - 9 27/jun limite 1. avaliacao ____ /
/ ____

SUDOKU para computadores

O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould - um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
					4	1		

O objetivo deste exercício é construir um resolvidor de SUDOKUS (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesmo para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num labirinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretensamente vai ser colocado nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna **Verdadeiro** sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna **falso**. Veja

```

função resolve(grade, linha, coluna, k)
    para x de 0 a 8
        se grade[linha][x] = k
            retorne Falso
        fim{se}
        se grade[x][coluna] = k
            retorne Falso
        fim{se}
    fim{para}
    comeccolin = lin - lin % 3
    comeccocol = col - col % 3
    para i de 0 a 2
        para j de 0 a 2
            se grade[comeccolin+i][comeccocol+j]=k
                retorne Falso
            fim{se}
        fim{para}
    fim{para}
    retorne Verdadeiro
fim{função}

```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
    se (linha=8) E (coluna=9)
        retorne Verdadeiro
    fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```

se coluna = 9
    linha = linha + 1
    coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
    retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
    se resolve(grade, linha, coluna, k)
        grade[linha][coluna] = k
        se sudoku (grade, linha, coluna+1)
            retorne Verdadeiro
        fim{se}
    fim{se}
    grade[linha][coluna]=0
fim{para}
fim{funcão}

```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
    imprima(grade)
senão
    imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
    int i,j;
    for (i=0;i<9;i++){
        for(j=0;j<9;j++){
            cout<<a[i][j]<<" ";
        }
        cout<<endl;
    }
}

int resolve(int grade[9][9], int lin, int col, int k){
    int i,j,x,comecolin,comecocol;
    for (x=0;x<9;x++){
        if (grade[lin][x]==k){return 0;}
    }
    for (x=0;x<9;x++){
        if (grade[x][col]==k){return 0;}
    }
    comecolin=lin-(lin%3);
    comecocol=col-(col%3);
}
```

```

for (i=0; i<3; i++){
    for (j=0; j<3; j++){
        if (grade[comecolin+i][comecocol+j]==k){
            return 0;
        }
    }
}

return 1;
}

int sudoku(int grade[9][9], int lin, int col){
    int k;
    if ((lin==8)&&(col==9)){return 1;}
    if (col==9){lin++; col=0;}
    if (grade[lin][col]>0){
        return sudoku(grade, lin, col+1);
    }
    for (k=1; k<10; k++){
        if (resolve(grade, lin, col, k)){
            grade[lin][col]=k;
            if (sudoku(grade, lin, col+1)){return 1;}
        }
        grade[lin][col]=0;
    }
    return 0;
}

int M[9][9]={1,0,0, 0,0,0, 2,0,0},
              {0,0,0, 1,0,3, 0,0,4},
              {0,0,5, 0,6,7, 8,0,0},
              {6,0,8, 0,9,1, 0,0,5},
              {0,5,0, 0,0,0, 0,1,0},
              {2,0,0, 6,4,0, 7,0,3},
              {0,0,7, 8,1,0, 6,0,0},
              {4,0,0, 5,0,2, 0,0,0},
              {0,0,2, 0,0,0, 0,0,9}};

int main(){
    if (sudoku(M, 0, 0)){puzzle(M);}
    else {cout<<" solucao impossivel";}
}

```

👉 Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

	7	9		5		1	8	
8								7
		7	3		6	8		
4	5		7		8		9	6
		3	5		2	7		
7								5
	1	6		3		4	2	

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
9,9	9,5	4,5

Responda aqui:

1	2	3
---	---	---



204-75765 - n lim

SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
				4	1			

O objetivo deste exercício é construir um resolveur de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesmo para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num labirinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força

bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretendamos colocar nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna Verdadeiro sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna falso. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for (j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
```

```
if (grade[x][col]==k){return 0;}
}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
for (i=0;i<3;i++){
for (j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9}};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

			7			8		
		6					3	1
	4				2			
	2	4		7				
	1			3			8	
				6		2	9	
			8				7	
8	6					5		
		2			6			

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
6,3	4,1	6,6
Responda aqui:		
1	2	3



SUDOKU para computadores O Sudoku é um quebra-cabeça moderno (parece ter sido inventado por um arquiteto aposentado de 74 anos, Howard Garns em 1979) e começou a se tornar popular no Japão, com o nome de *suuji wa dokushin ni kagiru*, que significa “os dígitos devem permanecer únicos”. Sua popularidade naquele país se explica pela dificuldade do idioma japonês para fazer palavras cruzadas. No Ocidente, o jogo só começou a se popularizar em 2004, quando Wayne Gould – um juiz aposentado de Hong Kong, que também era fã de puzzles e programador de computador – viajou a Londres para convencer os editores do The Times a publicar o sudoku. Gould havia criado um programa de computador que gerava jogos de sudoku com vários níveis de dificuldade e não estava cobrando nada por ele. O Times decidiu arriscar e no dia 12 de novembro de 2004 publicou seu primeiro sudoku. No Brasil o sudoku é publicado pela Coquetel (Ediouro) desde o início de 2005 e a Folha de São Paulo e a extinta Gazeta do Povo começaram a publicar um sudoku diariamente. O sudoku é muito popular hoje e parece ter as características de um bom e duradouro quebra cabeça:

- é disponível em diversos graus de dificuldade
- sua regra de solução é muito simples
- a habilidade de resolvê-lo cresce com a prática
- ao resolvê-lo a pessoa vê o progresso em direção à solução

O jogo é apresentado em uma grade 9×9 , que está dividida em 9 linhas, 9 colunas e 9 blocos (de 3×3 cada). Neste texto, estes três elementos (linhas, colunas e blocos) serão chamados genericamente de “grupos”. A regra única do SUDOKU é que em cada grupo deve haver um conjunto completo de números 1..9. Note-se que os números são usados apenas por comodidade, nenhuma propriedade aritmética deles é importante. Poder-se-iam usar 9 símbolos de times de futebol, cores, signos do zodíaco etc. Antes de continuar a leitura, tente resolver este sudoku

		4	8					
	9		4	6			7	
	5					6	1	4
2	1		6			5		
5	8		7		9		4	1
		7			8		6	9
3	4	5					9	
	6			3	7		2	
					4	1		

O objetivo deste exercício é construir um resolvidor de SUDOKUs (bem formados = que têm solução única). Este exercício vai ainda abordar os temas de **recursividade** (uma função chama a si mesmo para resolver uma nova instância (possivelmente mais simples), até que o problema todo esteja resolvido. Outro conceito importante é o de **backtracking** que é usado junto com a recursividade. Quando uma determinada linha de ação conduz a um beco sem saída, há que desfazer o último tramo e seguir por outro caminho. Pense num laberinto, cuja saída é desconhecida. É a mesma estratégia. Finalmente, o terceiro conceito deste exercício é o da **força bruta** que significa testar todas as alternativas possíveis em busca da certa. Claro está que esta estratégia só é possível a um computador, pela rapidez com que ele gera e testa as alternativas. Em compensação para usar a força bruta, não há necessidade de maior inteligência ou estratégia. É esta a troca que se propõe aqui.

O algoritmo

Primeiro vamos programar uma função simples que recebe uma linha e coluna e um valor k que pretendidamente vai ser colocado nessa linha,coluna. Esta função faz uma verificação na linha da grade, na coluna da grade e no bloco. Se o valor k não estiver em nenhuma dessas localizações, então a função retorna **Verdadeiro** sinalizando que k pode ser colocada aqui. Se o k aparecer em algum lugar dos pesquisados, esta função retorna falso. Veja

```
função resolve(grade, linha, coluna, k)
para x de 0 a 8
se grade[linha][x] = k
retorne Falso
fim{se}
se grade[x][coluna] = k
retorne Falso
fim{se}
fim{para}
comecolin = lin - lin % 3
comecocol = col - col % 3
para i de 0 a 2
para j de 0 a 2
se grade[comecolin+i][comecocol+j]=k
retorne Falso
fim{se}
fim{para}
fim{para}
retorne Verdadeiro
fim{função}
```

Agora vem a espinha dorsal do algoritmo. Primeiro o usuário deve montar a grade, preocupando-se em salvar a grade vazia, já que o algoritmo faz as modificações na grade passada, e que portanto fica diferente do que era (fica resolvida). Para esta função deve ser passado uma linha e coluna, que pode ser 0,0. Vamos por partes

```
função sudoku (grade, linha, coluna)
se (linha=8) E (coluna=9)
retorne Verdadeiro
fim{se}
```

Este teste no início impede que a linha extravase as dimensões da grade. Depois,

```
se coluna = 9
linha = linha + 1
coluna = 0
fim{se}
se grade[linha][coluna] > 0 // está preenchida
retorne sudoku (grade, linha, coluna + 1)
fim{se}
para k de 1 a 9
se resolve(grade, linha, coluna, k)
grade[linha][coluna] = k
se sudoku (grade, linha, coluna+1)
retorne Verdadeiro
fim{se}
fim{se}
grade[linha][coluna]=0
fim{para}
fim{função}
```

Agora, deve-se montar a grade e depois

```
se sudoku(grade,0,0)
imprima(grade)
senão
imprima ("sudoku sem solução")
fim{se}
```

veja a seguir uma implementação deste código em C++

```
#include<iostream>
using namespace std;
int puzzle(int a[9][9]){
int i,j;
for (i=0;i<9;i++){
for (j=0;j<9;j++){
cout<<a[i][j]<<" ";
cout<<endl;
}
}
int resolve(int grade[9][9], int lin, int col, int k){
int i,j,x,comecolin,comecocol;
for (x=0;x<9;x++){
if (grade[lin][x]==k){return 0;}
}
for (x=0;x<9;x++){
if (grade[x][col]==k){return 0;}
}
comecolin=lin-(lin%3);
comecocol=col-(col%3);
```

```
for (i=0;i<3;i++){
for (j=0;j<3;j++){
if (grade[comecolin+i][comecocol+j]==k){
return 0;
}
}
}
return 1;
}
int sudoku(int grade[9][9], int lin, int col){
int k;
if ((lin==8)&&(col==9)){return 1;}
if (col==9){lin++; col=0;}
if (grade[lin][col]>0){
return sudoku(grade, lin, col+1);
}
for (k=1; k<10; k++){
if (resolve(grade, lin, col, k)){
grade[lin][col]=k;
if (sudoku(grade, lin, col+1)){return 1;}
}
grade[lin][col]=0;
}
return 0;
}
int M[9][9]={1,0,0, 0,0,0, 2,0,0},
{0,0,0, 1,0,3, 0,0,4},
{0,0,5, 0,6,7, 8,0,0},
{6,0,8, 0,9,1, 0,0,5},
{0,5,0, 0,0,0, 0,1,0},
{2,0,0, 6,4,0, 7,0,3},
{0,0,7, 8,1,0, 6,0,0},
{4,0,0, 5,0,2, 0,0,0},
{0,0,2, 0,0,0, 0,0,9};
int main(){
if (sudoku(M, 0, 0)){puzzle(M);}
else {cout<<" solucao impossivel";}
}
```

Para você fazer

A seguir um sudoku para você resolver. Resolva a mão, ou usando este programa, à sua escolha.

		5		8		7		
7			2		4			5
3	2						8	4
	6		1		5		4	
		8				5		
	7		8		3		1	
4	5						9	1
6			5		8			7
		3		1		6		

Para efeitos de correção, localize os valores encontrados nas células calculadas

1) L,C	2) L,C	3) L,C
8,2	8,5	5,5
Responda aqui:		
1	2	3



204-75758 - n lim