Nome: Eduardo Tadahiro Nishizuka 7696393

Relatorio 2 - SSC0220 - Laboratório de Introdução à Ciência da Computação II
Todos os códigos juntos no diretório.

recursão_matriz

1. Ideia Geral: Demonstrar uso de recursao e comparacao com iteracao
2. Programa aloca uma matriz dado o numero de linhas e colunas do usuario e percorre ela usando recursão.
3. recursao_matriz-v2.c

```
void percorre_mat(int **mat, int l, int c, in

    static int num = 1;
    // 'visito' e marco a posicao l,c
    mat[l][c] = num++;

    // utiliza 4 vizinhos para percorrer a ma
    // direita, cima, esquerda, baixo

    // direita
    if (c+1 < m) {
```

4. recursao_matriz-v3.c

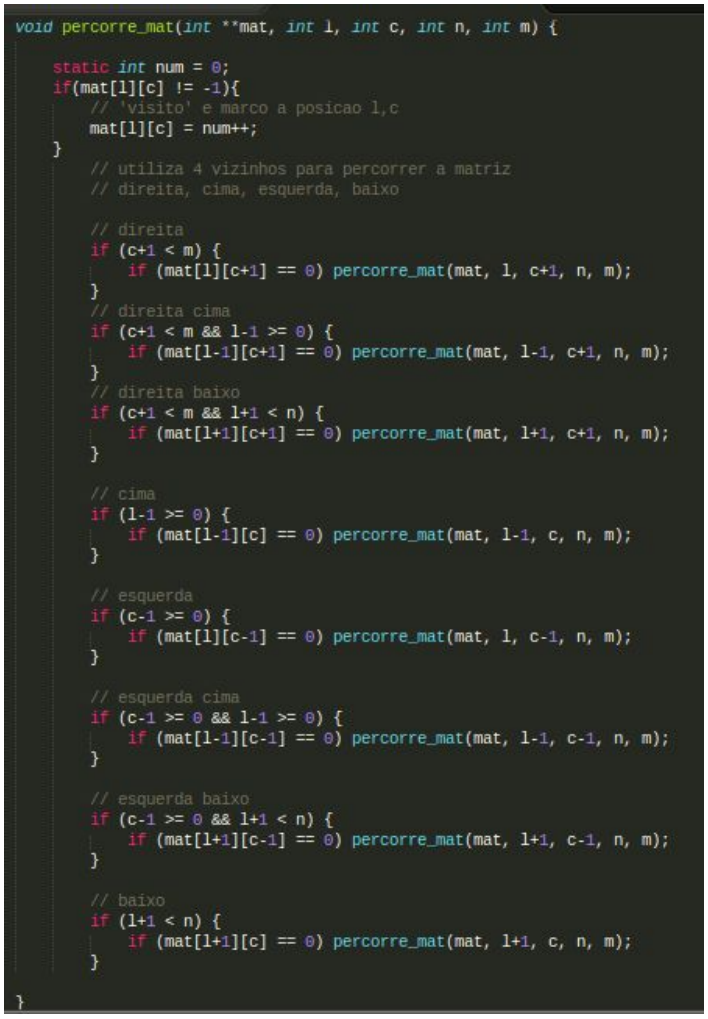
```
t main(int argc, char * argv[]) {

  if (argc < 5) {
      printf("Numero incorreto de parametros\n");
      printf("Uso: ./prog <linhas> <colunas> <linha inicial> <coluna inicial>\n");
      return 0;
  }
  int n = atoi(argv[1]);
  int m = atoi(argv[2]);
  int x = atoi(argv[3]);
  int y = atoi(argv[4]);
```

5. recursao_matriz-v4.c

```
void bloqueio(int **mat, int n, int m, int p){
    int nbloqueios = (p * n * m)/100;

    int i = 0;
    int x;
    int y;

    srand(time(NULL));

    while(i < nbloqueios){
        x = rand() % n;
        y = rand() % m;
        if(mat[x][y] != -1) mat[x][y] = -1;
        i++;
    }
}
```

6.recursao_matriz-v5.c

Dado a posição inicial, a recursão faz expandir a busca pelos vizinhos mais próximos(busca em profundidade)



```c
void percorre_mat(int **mat, int l, int c, int n, int m) {

    static int num = 0;
    if(mat[l][c] != -1){
        // 'visito' e marco a posicao l,c
        mat[l][c] = num++;
    }
        // utiliza 4 vizinhos para percorrer a matriz
        // direita, cima, esquerda, baixo

        // direita
        if (c+1 < m) {
            if (mat[l][c+1] == 0) percorre_mat(mat, l, c+1, n, m);
        }
        // direita cima
        if (c+1 < m && l-1 >= 0) {
            if (mat[l-1][c+1] == 0) percorre_mat(mat, l-1, c+1, n, m);
        }
        // direita baixo
        if (c+1 < m && l+1 < n) {
            if (mat[l+1][c+1] == 0) percorre_mat(mat, l+1, c+1, n, m);
        }

        // cima
        if (l-1 >= 0) {
            if (mat[l-1][c] == 0) percorre_mat(mat, l-1, c, n, m);
        }

        // esquerda
        if (c-1 >= 0) {
            if (mat[l][c-1] == 0) percorre_mat(mat, l, c-1, n, m);
        }

        // esquerda cima
        if (c-1 >= 0 && l-1 >= 0) {
            if (mat[l-1][c-1] == 0) percorre_mat(mat, l-1, c-1, n, m);
        }

        // esquerda baixo
        if (c-1 >= 0 && l+1 < n) {
            if (mat[l+1][c-1] == 0) percorre_mat(mat, l+1, c-1, n, m);
        }

        // baixo
        if (l+1 < n) {
            if (mat[l+1][c] == 0) percorre_mat(mat, l+1, c, n, m);
        }

}
```

criaarquivo.c

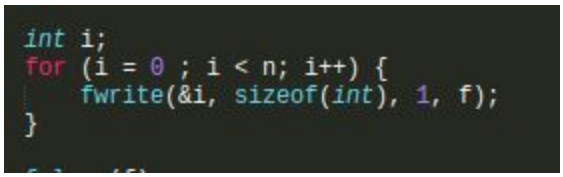
1. Programa que cria um arquivo binario contendo numeros inteiros
2. Programa pega o nome do arquivo a ser criado, a quantidade N den úmeros nesse arquivo e gera N números inteiros aleatórios. Por fim, ele escreve esses números num novo arquivo com o nome dado pelo parametro.

3-4. Tempos

n = 10	0.000254s
n = 100	0.000132s
n = 1000	0.000198s
n = 10000	0.002509s

n = 100000	0.016346s
n = 1000000	0.086980s
n = 10000000	0.752922s
n = 100000000	7.491380s

5. criaarquivo-v2.c



```
int i;
for (i = 0 ; i < n; i++) {
    fwrite(&i, sizeof(int), 1, f);
}
```

6. Tempos

n = 10	0.000104s
n = 100	0.000180s
n = 1000	0.000204s
n = 10000	0.001440s
n = 100000	0.011550s
n = 1000000	0.070844s
n = 10000000	0.658332s
n = 100000000	6.415646s

7. criaarquivo-v3.c

```c
srand(time(NULL));

int n = atoi(argv[2]); // total de elementos

int i;

t = clock();
for (i = 0 ; i < n; i++) {
    fwrite(&i, sizeof(int), 1, f);
}
printf("%fs\n", (float)(clock()-t)/CLOCKS_PER_SEC);

rewind(f);

t = clock();

for (i = 0 ; i < n; i++) {
    fread(&i, sizeof(int), 1, f);
}
printf("%fs\n", (float)(clock()-t)/CLOCKS_PER_SEC);
```

Tamanho	Tempo leitura	Tempo escrita
n = 10	0.000003s	0.000010s
n = 100	0.000005s	0.000009s
n = 1000	0.000043s	0.000034s
n = 10000	0.000561s	0.000424s
n = 100000	0.006876s	0.004076s
n = 1000000	0.053376s	0.022138s
n = 10000000	0.345089s	0.220467s
n = 100000000	3.278737s	2.182946s

busca.c

1. Programa que recebe um arquivo e uma chave como entrada e tenta encontrar a chave no arquivo
2. O programa pega o tamanho do vetor e aloca dinamicamente um vetor com valores aleatorios, conforme a funcao geraVetorAleatorio(). Dado a chave por parametro no programa, ele realiza a busca conforme o tipo de operação dado também no programa(argv[3]). Se o tipo de op. for 1, ele faz a busca sequencial pela chave dada. Se o tipo de op. for 2, ele faz a ordenação usando o insertionsort (melhor caso O(n)) e em seguida faz a busca binária.
3.

```
140
141     if (op == 1) {
142         fprintf(stdout,"Realizando busca...\n");
143         fflush(stdout);
144         c1 = clock();
145         pos = buscaSequencial(vet, n, chave);
146         printf("busca: %.6fs\n", (float)(clock()-c1)/CLOCKS_PER_SEC);
147     } else if (op == 2) {
148         fprintf(stdout,"Ordenando...\n");
149         fflush(stdout);
150         c1 = clock();
151         //bubbleSort(vet, n); // nao usar - dica do Obama
152         insertionSort(vet, n);
153         printf("sort: %.6fs\n", (float)(clock()-c1)/CLOCKS_PER_SEC);
154         fprintf(stdout,"Realizando busca...\n");
155         fflush(stdout);
156         c1 = clock();
157         pos = buscaBinaria(vet, 0, n-1, chave);
158         printf("busca: %.6fs\n", (float)(clock()-c1)/CLOCKS_PER_SEC);
159     }
160
```

Tempo com n = 1000
sort: 0.000042s busca: 0.000002s

4. busca-v3.c

```
122
123     if (argc < 5) {
124         printf("Numero de argumentos insuficiente\n");
125         printf("Uso: ./prog <tam_vetor> <chave> <tipo_busca>\n");
126         printf("\t<tipo> = 1 sequencial, 2 binaria\n\n");
127         printf("\t<sort> = 1 bubble, 2 insertion\n");
128         return 0;
129     }
130
131     int n = atoi(argv[1]); // tamanho do vetor
132     int chave = atoi(argv[2]); // numero a buscar nos dados
133     int op = atoi(argv[3]);
134     int sort = atoi(argv[4]);
135
136     int * vet = geraVetorAleatorio(n);
137     printf("exemplos: %d, %d\n", vet[n-2], vet[n-1]);
138
139     long int pos;
140
141     clock_t c1;
142
143     if (op == 1) {
144         fprintf(stdout,"Realizando busca...\n");
145         fflush(stdout);
146         c1 = clock();
147         pos = buscaSequencial(vet, n, chave);
148         printf("busca: %.6fs\n", (float)(clock()-c1)/CLOCKS_PER_SEC);
149     } else if (op == 2) {
150         fprintf(stdout,"Ordenando...\n");
151         fflush(stdout);
152         c1 = clock();
153         if(sort == 1)
154             bubbleSort(vet, n); // nao usar - dica do Obama
155         else if(sort == 2)
156             insertionSort(vet, n);
157         printf("sort: %.6fs\n", (float)(clock()-c1)/CLOCKS_PER_SEC);
158         fprintf(stdout,"Realizando busca...\n");
159         fflush(stdout);
160         c1 = clock();
161         pos = buscaBinaria(vet, 0, n-1, chave);
162         printf("busca: %.6fs\n", (float)(clock()-c1)/CLOCKS_PER_SEC);
163     }
164
```

5. Tempos

N	Tempo seq.	busca binaria	bubblesort	insertionsort
10	0.000001s	0.000003s	0.000004s	0.000001s
100	0.000002s	0.000001s	0.000156s	0.000053s
1000	0.000007s	0.000001s	0.013638s	0.003679s
10000	0.000051s	0.000001s	0.714198s	0.142448s

6. busca-v4.c

```
131
132     if (argc < 6) {
133         printf("Numero de argumentos insuficiente\n");
134         printf("Uso: ./prog <tam_vetor> <chave> <tipo_busca>\n");
135         printf("\t<tipo> = 1 sequencial, 2 binaria\n\n");
136         printf("\t<sort> = 1 bubble, 2 insertion\n");
137         printf("\t<tipo_vetor> = 1 aleatorio, 2 sequencial\n");
138         return 0;
139     }
140
141     int n = atoi(argv[1]); // tamanho do vetor
142     int chave = atoi(argv[2]); // numero a buscar nos dados
143     int op = atoi(argv[3]);
144     int sort = atoi(argv[4]);
145     int tipovet = atoi(argv[5]);
146
147     int * vet;
148     if(tipovet == 1)
149         vet = (int *)geraVetorAleatorio(n);
150     else if (tipovet == 2)
151         vet = (int *)geraVetorSequencial(n);
152     printf("exemplos: %d, %d\n", vet[n-2], vet[n-1]);
153
154     long int pos;
155
```

7. Tempos
usando um vetor de tamanho n = 1000

Vetor:	Aleatorio	Sequencial
busca seq	0.000011s	0.000001s
bubblesort	0.014951s	0.006259s
insertionsort	0.003701s	0.000032s
busca bin.	0.000003s	0.000002s

TAD arquivo:

intArray.c