3ª PROVA DE MTP

ALUNOS:
João Víctor Oliveira Mendes - 11711EBI005
Mariana Rigo Estevão - 11711EBI008
Felipe Ferreira Lobato - 11711EBI026

QUESTÃO 01

MAT0 = 11711EBI005;
MAT1 = 11711EBI008;
MAT2 = 11711EBI026
KANO0 = 3; KCUR0 = 2; KNUM0 = 6
KANO1 = 3; KCUR1 = 2; KNUM1 = 9
KANO2 = 3; KCUR2 = 2; KNUM2 = 9

QUESTÃO 02

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>

#define N 10
#define KANO0 3
#define KANO1 3
#define KANO2 3
#define KCUR0 2
#define KCUR1 2
#define KCUR2 2
#define KNUM0 6
#define KNUM1 9
#define KNUM2 9
float media_de_aleatorios(int ID) {
int * p = (int *) malloc(N*sizeof(int));
int i;
float media = 0;
for(i = 0; i < N; i++)
        {
                p[i] = rand()%9 + 1;
                media += p[i];
        }
        return media;
        free(media_de_aleatorios);
}
```

```
int main() {
int ID0 = (KANO0+KANO1+KANO2)%9 + 1,
ID1 = (KCUR0+KCUR1+KCUR2)%9 + 1,
ID2 = (KNUM0+KNUM1+KNUM2)%9 + 1;
srand(time(NULL));
printf("1o: %f\n", media_de_aleatorios(ID0));
printf("2o: %f\n", media_de_aleatorios(ID1));
printf("3o: %f\n", media_de_aleatorios(ID2));
getche ();
return EXIT_SUCCESS;
}
```

O problema de memória estava na não liberação dos números gravados na variável "media", fazendo com que se acumulasse os resultados ao invés de limparem a memória.
Para isso, usamos a função free(media_de_aleatorios).
O problema de coerência no código estava na função srand, que para não repetir a sequência de números aleatórios, precisa ser colocada como srand(time(NULL)), sem esquecer da biblioteca #include <time.h>.

QUESTÃO 03

a)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define KANO0 3
#define KANO1 3
#define KANO2 3
#define KCUR0 2
#define KCUR1 2
#define KCUR2 2
#define KNUM0 6
#define KNUM1 9
#define KNUM2 9


double f(double x) {
   double y = 0.0;
   double PI = 4.0*atan(1.0);
   int ID2 = (KNUM0+KNUM1+KNUM2)%9 + 1;
```

```
    switch(ID2) {
        case 0: y = x*x-5*x+6; break;
        case 1: y = 2.0*PI*x; break;
        case 2: y = PI*x*x; break;
        case 3: y = 6*x/PI; break;
        case 4: y = x*x*2.0*PI; break;
        case 5: y = -x*x+5*x-6; break;
        case 6: y = 3.5*x-2.0; break;
        case 7: y = PI*x/2.0; break;
        case 8: y = -PI*x+0.1*x; break;
        case 9: y = 2.0*x-3.0*PI; break;
        default: y = 0.0;
    }
    return y;
}

int main() {
    int ID0 = (KANO0+KANO1+KANO2)%9 + 1,
        ID1 = (KCUR0+KCUR1+KCUR2)%9 + 1,
        ID2 = (KNUM0+KNUM1+KNUM2)%9 + 1;
    srand(ID0*100+ID1*10+ID2);
    double x, y; int i;
    FILE * arq;
    remove("dados.dat");
    arq = fopen("dados.dat","ab");
    for(i = 0; i < 100; i++) {
        x = (double) rand()/RAND_MAX;
        y = f(x);
        fwrite(&y,sizeof(double),1,arq);
    }
    printf("Codigo: %d%d%d\n",ID0,ID1,ID2);
    fclose(arq);
    return EXIT_SUCCESS;
}
```

Resposta obtida: código 177

b)

```
#include <stdio.h>
	#include <stdlib.h>
	#define KANO0 3
	#define KANO1 3
	#define KANO2 3
	#define KCUR0 2
	#define KCUR1 2
	#define KCUR2 2
	#define KNUM0 6
	#define KNUM1 9
	#define KNUM2 9

	double media(double a, double b, double c) {
	return (a+b+c)/3;
	}

	int main() {
	int ID0 = (KANO0+KANO1+KANO2)%9 + 1,
	ID1 = (KCUR0+KCUR1+KCUR2)%9 + 1,
	ID2 = (KNUM0+KNUM1+KNUM2)%9 + 1;
	FILE * arq;
	int idA, idB, idC;
	double nA, nB, nC;
	arq = fopen("dados.dat","rb");
	if(arq == NULL) {
	fprintf(stderr,"Arquivo inexistente!\n");
	return EXIT_FAILURE;
	}
	switch(ID2) {
	case 1: idA = 13; idB = 14; idC = 64; break;
	case 2: idA = 21; idB = 42; idC = 84; break;
	case 3: idA = 23; idB = 37; idC = 46; break;
	case 4: idA = 16; idB = 55; idC = 82; break;
	case 5: idA = 9; idB = 33; idC = 76; break;
	case 6: idA = 0; idB = 39; idC = 99; break;
	case 7: idA = 10; idB = 86; idC = 92; break;
	case 8: idA = 17; idB = 61; idC = 92; break;
	case 9: idA = 11; idB = 24; idC = 77; break;
	case 10: idA = 5; idB = 53; idC = 65; break;
	default: idA = idB = idC = 0;
	}
int i;
	for (i=0;i<idA;i++)
```

```
        {
                fseek(arq,i*sizeof(double),SEEK_SET);
                fread(&nA, sizeof(double),1,arq);
        }
        for (i=0;i<idB;i++)
        {
                fseek(arq,i*sizeof(double),SEEK_SET);
                fread(&nB, sizeof(double),1,arq);
        }
        for (i=0;i<idC;i++)
        {
                fseek(arq,i*sizeof(double),SEEK_SET);
                fread(&nC, sizeof(double),1,arq);
        }
        fclose(arq);
        printf("Matricula: %d%d%d\n",ID0,ID1,ID2);
        printf("Media [%lf %lf %lf] = %lf\n",nA,nB,nC,media(nA,nB,nC));
return EXIT_SUCCESS;
 }
```

Na main, é definido ID0, ID1 e ID2 de acordo com os números gerados pelas matrículas. Já de acordo com ID2 é que são definidos idA, idB e idB, que serão utilizados para inicializar nA, nB e nC. Para fazer isso, utilizamos 3 laços, um para cada variável, onde i varia de 0 até a posição-1, e, utilizando a função fseek acessamos a posição desejada cujo número é utilizado na inicialização de nA, nB e nC através de fread. Esses três, por fim, são utilizados como parâmetro para a função “media”, que calcula a média dos 3 e retorna o resultado ao usuário.

-

QUESTÃO 04

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#define KANO0 3
#define KANO1 3
#define KANO2 3
#define KCUR0 2
#define KCUR1 2
#define KCUR2 2
#define KNUM0 6
#define KNUM1 9
#define KNUM2 9
```

```
typedef
struct Aluno {
        char nome[256];
        int matricula;
        unsigned int idade;
}
aluno;

void mostrar(Aluno aluno) {
printf("> %s: MAT %03d\n: %u anos;\n", aluno.nome, aluno.matricula,
aluno.idade);
}

void gravar(Aluno aluno) {
FILE * arq;
arq = fopen("registro.txt","ab");
fwrite(&(aluno.nome),256,1,arq);
fwrite(&(aluno.matricula),sizeof(int),1,arq);
fwrite(&(aluno.idade),sizeof(unsigned int),1,arq);
fclose(arq);
}

int ler(FILE * arq, Aluno * paluno, unsigned int id) {
arq = fopen("registro.txt","r");
fseek(arq,id*sizeof(Aluno),SEEK_SET);
int ok = fread(&(paluno->nome),256,1,arq);
fread(&(paluno->matricula),sizeof(int),1,arq);
fread(&(paluno->idade),sizeof(unsigned int),1,arq);
return ok;
}

void inicia() {
remove("registro.txt");
Aluno aluno;
strncpy(aluno.nome,"Oswald",256);
aluno.matricula = rand()%999 + 1;
aluno.idade = rand()%11 + 17;
gravar(aluno);
strncpy(aluno.nome,"João", 256);
aluno.matricula = 5;
aluno.idade = 19;
gravar(aluno);
```

```
strncpy(aluno.nome,"Mariana", 256);
aluno.matricula = 8;
aluno.idade = 20;
gravar (aluno);
strncpy(aluno.nome,"Felipe", 256);
aluno.matricula = 26;
aluno.idade = 18;
gravar(aluno);

strncpy(aluno.nome,"Ermengardo", 256);

aluno.matricula = 34;

aluno.idade = 101;

gravar(aluno);

strncpy(aluno.nome,"Juriemo", 256);

aluno.matricula = 75;

aluno.idade = 12;

gravar(aluno);

strncpy(aluno.nome,"Silvia", 256);

aluno.matricula = rand()%999 + 1;
aluno.idade = rand()%15 + 17;
gravar(aluno);
strncpy(aluno.nome,"Mickey", 256);
aluno.matricula = rand()%999 + 1;
aluno.idade = rand()%9 + 17;
gravar(aluno);
}

int main() {
int ID0 = (KANO0+KANO1+KANO2)%9 + 1,
ID1 = (KCUR0+KCUR1+KCUR2)%9 + 1,
ID2 = (KNUM0+KNUM1+KNUM2)%9 + 1;
srand(ID0*100+ID1*10+ID2);
Aluno aluno;
FILE * arq;
unsigned int i;
inicia();
arq = fopen("registro.txt","r");
```

```
i = 0;
while(!feof(arq)) {
if(ler(arq, &aluno, i))
mostrar(aluno);
i++;
}
fclose(arq);
getche();
return EXIT_SUCCESS;
}
```

Inicialmente, para ser possível rodar o programa foi necessário que não utilizássemos “00” na matrícula “008”, visto que, caso o fizéssemos, ele seria interpretado como um número octal e não seria possível rodar o programa.
Foi necessário chamar a função “gravar” para o aluno “Emengardo”, pois esta não estava sendo gravado no arquivo.
Já no código, é necessário trocar o w utilizado na função “gravar”, esse modo é usado para criar um arquivo ou substituir seu conteúdo anterior, que não é o ideal para esse caso. Portanto, utilizamos o modo “ab”, com o qual anexamos informações no final do arquivo.
Foi necessário, também, inverter as duas últimas linhas da função “ler” para que a leitura ocorresse na ordem em que é gravado no arquivo.