

Fundamentos da Programação

Gestão de Registos

Conteúdo

- Gestão de Registos
- Operações *Create*, *Read*, *Update*, and *Delete* (CRUD)

Gestão de alunos

```
typedef struct {
    int ano, mes, dia;
} Data;

typedef struct {
    int numero;
    char nome[MAX_NOME_ALUNO];
    Data data_nascimento;
} Aluno;

typedef struct {
    int contador;
    Aluno alunos[MAX_ALUNOS];
} Alunos;
```

...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		
	.alunos[1]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		
	.alunos[0]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
.nome				
.numero		0		
.contador		0		
...				

Inserir

...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[1]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[0]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.contador		0	
...				

...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[1]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[0]	.data_nascimento	.ano	1976
			.mes	06
			.dia	10
		.nome		
		.numero		1
	.contador		1	
...				

Inserir

...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[1]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[0]	.data_nascimento	.ano	1976
			.mes	06
			.dia	10
		.nome		
		.numero		1
.contador			1	
...				

...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[1]	.data_nascimento	.ano	2007
			.mes	08
			.dia	14
		.nome		
		.numero		2
	.alunos[0]	.data_nascimento	.ano	1976
			.mes	06
			.dia	10
		.nome		
		.numero		1
.contador		2		
...				

Consultar

...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[1]	.data_nascimento	.ano	2007
			.mes	08
			.dia	14
		.nome		
		.numero		2
	.alunos[0]	.data_nascimento	.ano	1976
			.mes	06
			.dia	10
		.nome		
		.numero		1
	.contador		2	
...				

2 < Alunos.contador

1 < Alunos.contador

0 < Alunos.contador

Atualizar

...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[1]	.data_nascimento	.ano	2007
			.mes	08
			.dia	14
		.nome		
		.numero		2
	.alunos[0]	.data_nascimento	.ano	1976
			.mes	06
			.dia	10
		.nome		
		.numero		1
	.contador		2	
...				



...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[1]	.data_nascimento	.ano	2012
			.mes	02
			.dia	22
		.nome		
		.numero		2
	.alunos[0]	.data_nascimento	.ano	1976
			.mes	06
			.dia	10
		.nome		
		.numero		1
	.contador		2	
...				

Eliminar

...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
		.numero		0
	.alunos[1]	.data_nascimento	.ano	2007
			.mes	08
			.dia	14
		.nome		
		.numero		2
	.alunos[0]	.data_nascimento	.ano	1976
			.mes	06
			.dia	10
		.nome		
		.numero		1
	.contador		2	
...				



...				
Alunos		
		
		
	.alunos[2]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
	.numero		0	
	.alunos[1]	.data_nascimento	.ano	0
			.mes	0
			.dia	0
		.nome		
	.numero		0	
	.alunos[0]	.data_nascimento	.ano	2007
			.mes	08
.dia			14	
.nome				
.numero		2		
.contador		1		
...				

Gestão de alunos



input.c



alunos.c



main.c



input.h



alunos.h

input.h

```
#ifndef INPUT_H
#define INPUT_H

int enterInt(int minVal, int maxVal, char *msg);

float enterFloat(float minVal, float maxVal, char *msg);

double enterDouble(double minVal, double maxVal, char *msg);

char enterChar(char *msg);

void enterString(char *string, unsigned int maxSize, char *msg);

#endif // INPUT_H
```

input.h

```
#ifndef INPUT_H
#define INPUT_H

...

#endif /* INPUT_H */
```

```
#ifndef INPUT_H
#define INPUT_H

int strToInt(int number, int number, char *msg);

float strToFloat(float number, float number, char *msg);

double strToDouble(double number, double number, char *msg);

char strToChar(char *msg);

void strToString(char *string, unsigned int number, char *msg);

#endif /* INPUT_H */
```

input.h

```
int obterInt(int minValor, int maxValor, char *msg);

float obterFloat(float minValor, float maxValor, char *msg);

double obterDouble(double minValor, double maxValor, char *msg);

char obterChar(char *msg);

void lerString(char *string, unsigned int tamanho, char *msg);
```

```
/* Função obterInt */
/* Função obterFloat */
/* Função obterDouble */
/* Função obterChar */
/* Função lerString */
```

input.c

```

#include <stdio.h>
#include <string.h>

#define MAX_SIZE 1024
#define MAX_BUFFER 1024

void cleanupBuffer() {
    char ch;
    while ((ch = getchar()) != '\n' && ch != EOF);
}

int isInteger(int min, int max, char *msg) {
    int value;
    printf(msg);
    while (scanf("%d", &value) != 1 || value < min || value > max) {
        puts(MAX_BUFFER);
        cleanupBuffer();
        printf(msg);
    }
    cleanupBuffer();
    return value;
}

float getFloat(float min, float max, char *msg) {
    float value;
    printf(msg);
    while (scanf("%f", &value) != 1 || value < min || value > max) {
        puts(MAX_BUFFER);
        cleanupBuffer();
        printf(msg);
    }
    cleanupBuffer();
    return value;
}

double getDouble(double min, double max, char *msg) {
    double value;
    printf(msg);
    while (scanf("%lf", &value) != 1 || value < min || value > max) {
        puts(MAX_BUFFER);
        cleanupBuffer();
        printf(msg);
    }
    cleanupBuffer();
    return value;
}

char *getString(char *msg) {
    char value;
    printf(msg);
    value = getchar();
    cleanupBuffer();
    return value;
}

void testString(char *msg, unsigned int times, char *msg) {
    printf(msg);
    if (fgets(msg, sizeof msg, stdin) != NULL) {
        unsigned int len = strlen(msg) - 1;
        if (msg[len] == '\n') {
            msg[len] = '\0';
        } else {
            cleanupBuffer();
        }
    }
}

```

input.c

```
#include <stdio.h>
#include <string.h>

#define VALOR_INVALIDO "O valor inserido é inválido."
```

```
#include <stdio.h>
#include <string.h>

#define VALOR_INVALIDO "O valor inserido é inválido."

void limpaBuffer() {
    char ch;
    while ((ch = getchar()) != '\n' && ch != EOF);
}

int obterInt(int minimo, int maximo, char msg) {
    int valor;
    printf(msg);
    while (scanf("%d", &valor) != 1 || valor < minimo || valor > maximo) {
        puts(VALOR_INVALIDO);
        limpaBuffer();
        printf(msg);
    }
    limpaBuffer();
    return valor;
}

float obterFloat(float minimo, float maximo, char msg) {
    float valor;
    printf(msg);
    while (scanf("%f", &valor) != 1 || valor < minimo || valor > maximo) {
        puts(VALOR_INVALIDO);
        limpaBuffer();
        printf(msg);
    }
    limpaBuffer();
    return valor;
}

double obterDouble(double minimo, double maximo, char msg) {
    double valor;
    printf(msg);
    while (scanf("%lf", &valor) != 1 || valor < minimo || valor > maximo) {
        puts(VALOR_INVALIDO);
        limpaBuffer();
        printf(msg);
    }
    limpaBuffer();
    return valor;
}

char obterChar(char msg) {
    char valor;
    printf(msg);
    valor = getchar();
    limpaBuffer();
    return valor;
}

void lerString(char *string, unsigned int tamanho, char msg) {
    printf(msg);
    if (fgets(string, tamanho, stdin) != NULL) {
        unsigned int len = strlen(string) - 1;
        if (string[len] == '\n') {
            string[len] = '\0';
        } else {
            limpaBuffer();
        }
    }
}
```

input.c

```
void cleanInputBuffer() {
    char ch;
    while ((ch = getchar()) != '\n' && ch != EOF);
}
```

```

Minimize cmin10;
Minimize cstrng10;

Minimize VARIM_0001000 "0 value Minimize < 10value10;"

void cmin10(cmin10ffer**) {
    char *str;
    while (ch = getchar()) != '\n' || ch != '\0';
}

int min10(int min10r, int max10r, char *msg) {
    int value;
    printf(msg);
    while (scanf("%d", &value) != 1) value = min10r || value > max10r ||
        pos(MIN_0001000);
    cmin10chbffer();
    printf(msg);
}
cmin10chbffer();
return value;
}

float min10(float min10r, float max10r, char *msg) {
    float value;
    printf(msg);
    while (scanf("%f", &value) != 1) value = min10r || value > max10r ||
        pos(MIN_0001000);
    cmin10chbffer();
    printf(msg);
}
cmin10chbffer();
return value;
}

double min10(double min10r, double max10r, char *msg) {
    double value;
    printf(msg);
    while (scanf("%lf", &value) != 1) value = min10r || value > max10r ||
        pos(MIN_0001000);
    cmin10chbffer();
    printf(msg);
}
cmin10chbffer();
return value;
}

char *str10(char *msg) {
    char *str;
    printf(msg);
    value = getchar();
    cmin10chbffer();
    return value;
}

void h10(char *str10, unsigned int tmax10, char *msg) {
    printf(msg);
    if (fgets(str10, tmax10, stdin) != NULL) {
        unsigned int len = strlen(str10);
        if (str10[len - 1] == '\n') {
            str10[len] = '\0';
        }
        cmin10chbffer();
    }
}

```

input.c

```
int obterInt(int minValor, int maxValor, char *msg) {
    int valor;
    printf(msg);
    while (scanf("%d", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}
```

```
/*teste.c*/
#include <stdio.h>
#include <string.h>

#define VALOR_MINIMO 0
#define VALOR_MAXIMO 1000000000

void cleanInputBuffer() {
    char ch;
    while ((ch = getchar()) != '\n' && ch != EOF);
}

int obterInt(int minValor, int maxValor, char *msg) {
    int valor;
    printf(msg);
    while (scanf("%d", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}

float obterFloat(float minValor, float maxValor, char *msg) {
    float valor;
    printf(msg);
    while (scanf("%f", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}

double obterDouble(double minValor, double maxValor, char *msg) {
    double valor;
    printf(msg);
    while (scanf("%lf", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}

char obterChar(char *msg) {
    char valor;
    printf(msg);
    valor = getchar();
    cleanInputBuffer();
    return valor;
}

void lerString(char *string, unsigned int tamanho, char *msg) {
    printf(msg);
    if (fgets(string, tamanho, stdin) != NULL) {
        unsigned int len = strlen(string) - 1;
        if (string[len] == '\n') {
            string[len] = '\0';
        } else {
            cleanInputBuffer();
        }
    }
}
```


input.c

```
float obterFloat(float minValor, float maxValor, char *msg) {
    float valor;
    printf(msg);
    while (scanf("%f", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}
```

```
/*teste.c*/
#include <stdio.h>
#include <string.h>

#define VALOR_MINIMO 0.0
#define VALOR_MAXIMO 100.0

void cleanInputBuffer() {
    char ch;
    while ((ch = getchar()) != '\n' && ch != EOF);
}

int obterInt(int minValor, int maxValor, char *msg) {
    int valor;
    printf(msg);
    while (scanf("%d", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}

float obterFloat(float minValor, float maxValor, char *msg) {
    float valor;
    printf(msg);
    while (scanf("%f", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}

double obterDouble(double minValor, double maxValor, char *msg) {
    double valor;
    printf(msg);
    while (scanf("%lf", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}

char obterChar(char *msg) {
    char valor;
    printf(msg);
    valor = getchar();
    cleanInputBuffer();
    return valor;
}

void lerString(char *string, unsigned int tamanho, char *msg) {
    printf(msg);
    if (fgets(string, tamanho, stdin) != NULL) {
        unsigned int len = strlen(string) - 1;
        if (string[len] == '\n') {
            string[len] = '\0';
        } else {
            cleanInputBuffer();
        }
    }
}
```

input.c

```
double obterDouble(double minValor, double maxValor, char *msg) {
    double valor;
    printf(msg);
    while (scanf("%lf", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}
```

```
/*teste.c*/
#include <stdio.h>
#include <string.h>

#define VALOR_MINIMO 0.0
#define VALOR_MAXIMO 10.0

void cleanInputBuffer() {
    char ch;
    while ((ch = getchar()) != '\n' && ch != EOF);
}

int obterInt(int minValor, int maxValor, char *msg) {
    int valor;
    printf(msg);
    while (scanf("%i", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}

float obterFloat(float minValor, float maxValor, char *msg) {
    float valor;
    printf(msg);
    while (scanf("%f", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}

double obterDouble(double minValor, double maxValor, char *msg) {
    double valor;
    printf(msg);
    while (scanf("%lf", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}

char obterChar(char *msg) {
    char valor;
    printf(msg);
    valor = getchar();
    cleanInputBuffer();
    return valor;
}

void testString(char *string, unsigned int tamanho, char *msg) {
    printf(msg);
    if (fgets(string, tamanho, stdin) != NULL) {
        unsigned int len = strlen(string) - 1;
        if (string[len] == '\n') {
            string[len] = '\0';
        } else {
            cleanInputBuffer();
        }
    }
}
```

input.c

```
char obterChar(char *msg) {  
    char valor;  
    printf(msg);  
    valor = getchar();  
    cleanInputBuffer();  
    return valor;  
}
```

```
#include <stdio.h>  
#include <string.h>  
  
#define MAX_SIZE 100  
#define MAX_BUFFER 100  
  
void cleanInputBuffer() {  
    char ch;  
    while ((ch = getchar()) != '\n' && ch != EOF);  
}  
  
int obterInt(int min, int max, int *valor, char *msg) {  
    int valor;  
    printf(msg);  
    while (scanf("%d", &valor) != 1 || valor < min || valor > max) {  
        printf("Invalido!\n");  
        cleanInputBuffer();  
        printf(msg);  
    }  
    cleanInputBuffer();  
    return valor;  
}  
  
float obterFloat(float min, float max, float *valor, char *msg) {  
    float valor;  
    printf(msg);  
    while (scanf("%f", &valor) != 1 || valor < min || valor > max) {  
        printf("Invalido!\n");  
        cleanInputBuffer();  
        printf(msg);  
    }  
    cleanInputBuffer();  
    return valor;  
}  
  
double obterDouble(double min, double max, double *valor, char *msg) {  
    double valor;  
    printf(msg);  
    while (scanf("%lf", &valor) != 1 || valor < min || valor > max) {  
        printf("Invalido!\n");  
        cleanInputBuffer();  
        printf(msg);  
    }  
    cleanInputBuffer();  
    return valor;  
}  
  
char obterChar(char *msg) {  
    char valor;  
    printf(msg);  
    valor = getchar();  
    cleanInputBuffer();  
    return valor;  
}  
  
void lerString(char *string, unsigned int tamanho, char *msg) {  
    printf(msg);  
    if (fgets(string, tamanho, stdin) != NULL) {  
        unsigned int len = strlen(string) - 1;  
        if (string[len] == '\n') {  
            string[len] = '\0';  
        } else {  
            cleanInputBuffer();  
        }  
    }  
}
```

input.c

```
void lerString(char *string, unsigned int tamanho, char *msg) {  
    printf(msg);  
    if (fgets(string, tamanho, stdin) != NULL) {  
        unsigned int len = strlen(string) - 1;  
        if (string[len] == '\\n') {  
            string[len] = '\\0';  
        } else {  
            cleanInputBuffer();  
        }  
    }  
}
```

```
int main(void) {  
    char *string =  
        "Digite o valor de x e y para calcular a soma:";  
    char ch;  
    while ((ch = getchar()) != '\\n' && ch != EOF);  
    int valor1, valor2, soma, tamanho, msg;  
    int valor;  
    printf(msg);  
    while (scanf("%d", &valor) != 1 || valor < 0 || valor > 1000) {  
        printf("Valor inválido. Digite um valor entre 0 e 1000: ");  
        cleanInputBuffer();  
    }  
    soma = valor1 + valor2;  
    printf("A soma é: %d\\n", soma);  
    return 0;  
}  
  
float obterValor(float *valor1, float *valor2, char *msg) {  
    float valor;  
    printf(msg);  
    while (scanf("%f", &valor) != 1 || valor < 0 || valor > 1000) {  
        printf("Valor inválido. Digite um valor entre 0 e 1000: ");  
        cleanInputBuffer();  
    }  
    *valor1 = valor;  
    return *valor1;  
}  
  
double obterValor(double *valor1, double *valor2, char *msg) {  
    double valor;  
    printf(msg);  
    while (scanf("%lf", &valor) != 1 || valor < 0 || valor > 1000) {  
        printf("Valor inválido. Digite um valor entre 0 e 1000: ");  
        cleanInputBuffer();  
    }  
    *valor1 = valor;  
    return *valor1;  
}  
  
char obterChar(char *msg) {  
    char valor;  
    printf(msg);  
    valor = getchar();  
    cleanInputBuffer();  
    return valor;  
}  
  
void lerString(char *string, unsigned int tamanho, char *msg) {  
    printf(msg);  
    if (fgets(string, tamanho, stdin) != NULL) {  
        unsigned int len = strlen(string) - 1;  
        if (string[len] == '\\n') {  
            string[len] = '\\0';  
        } else {  
            cleanInputBuffer();  
        }  
    }  
}
```

alunos.h

```
#ifndef ALUNOS_H
#define ALUNOS_H

#define MAX_ALUNOS 10

#define ERRO_ALUNO_NAO_ENCONTRADO "O aluno não existe na lista."
#define ERRO_LISTA_VAZIA "A lista de alunos está vazia."
#define ERRO_LISTA_DUPLO "A lista de alunos está cheia."
#define ERRO_ALUNO_DUPLO "O número de aluno já se encontra atribuído."

#define MAX_NOME_ALUNO 40
#define MAX_ID_ALUNO 1000
#define MAX_ID_DATA_ALUNO "Insira um número de aluno (0-1000): "

#define MAX_NOME_ALUNO 40
#define MAX_ID_DATA_ALUNO "Insira o nome do aluno: "

#define MAX_ID_ALUNO 40
#define MAX_ID_DATA_ALUNO "Insira o dia de nascimento: "

#define MAX_ID_ALUNO 40
#define MAX_ID_DATA_ALUNO "Insira o mês de nascimento: "

#define MAX_ID_ALUNO 40
#define MAX_ID_DATA_ALUNO "Insira o ano de nascimento: "

typedef struct {
    int ano, mes, dia;
} Data;

typedef struct {
    int numero;
    char nome[MAX_NOME_ALUNO];
    Data data_nascimento;
} Aluno;

typedef struct {
    int contador;
    Aluno alunos[MAX_ALUNOS];
} Alunos;

void inserirAluno(Alunos *alunos);
void procurarAluno(Alunos *alunos);
void atualizarAluno(Alunos *alunos);
void removerAluno(Alunos *alunos);
void listarAlunos(Alunos *alunos);

#endif /* ALUNOS_H */
```

alunos.h

```
#ifndef ALUNOS_H
#define ALUNOS_H

...

#endif /* ALUNOS_H */
```

```
#ifndef ALUNOS_H
#define ALUNOS_H

#define MAX_ALUNOS 10

#define MAX_ALUNO_NOME_LEN 50
#define MAX_ALUNO_ID_LEN 10
#define MAX_ALUNO_DATA_LEN 10
#define MAX_ALUNO_DATA_YEAR 4

#define MAX_NOME_LEN 50
#define MAX_ID_LEN 10
#define MAX_DATA_LEN 10
#define MAX_DATA_YEAR 4

#define MAX_NOME_LEN 50
#define MAX_ID_LEN 10
#define MAX_DATA_LEN 10
#define MAX_DATA_YEAR 4

typedef struct {
    int ano, mes, dia;
} Data;

typedef struct {
    int numero;
    char nome[MAX_NOME_LEN];
    Data data_nascimento;
} Aluno;

typedef struct {
    int tamanho;
    Aluno alunos[MAX_ALUNOS];
} Alunos;

void inserirAluno(Alunos *alunos);
void procurarAluno(Alunos *alunos);
void atualizarAluno(Alunos *alunos);
void removerAluno(Alunos *alunos);
void listarAlunos(Alunos *alunos);

#endif /* ALUNOS_H */
```

alunos.h

```
#define MAX_ALUNOS 30
```

```
#ifndef ALUNOS_H
#define ALUNOS_H

#define MAX_ALUNOS 30

#define ERRO_ALUNO_NAO_ENCONTRADO "O aluno não existe na lista."
#define ERRO_LISTA_VAZIA "A lista de alunos está vazia."
#define ERRO_LISTA_DUPLO "A lista de alunos está cheia."
#define ERRO_ALUNO_DUPLO "O número de aluno já se encontra atribuído."

#define MAX_NUM_ALUNO 1
#define MAX_NUM_ALUNO 1000
#define MAX_DATA_NUM_ALUNO "Insira um número de aluno (0-1000): "

#define MAX_NUM_ALUNO 30
#define MAX_DATA_NUM "Insira o nome do aluno: "

#define MAX_DIA 1
#define MAX_DIA 31
#define DATA_DIA_MES "Insira o dia de nascimento: "

#define MAX_MES 1
#define MAX_MES 12
#define DATA_MES_ANO "Insira o mês de nascimento: "

#define MAX_ANO 1000
#define MAX_ANO 2021
#define DATA_ANO_MES "Insira o ano de nascimento: "

typedef struct {
    int ano, mes, dia;
} Data;

typedef struct {
    int numero;
    char nome[MAX_NUM_ALUNO];
    Data data_nascimento;
} Aluno;

typedef struct {
    int contador;
    Aluno alunos[MAX_ALUNOS];
} Alunos;

void inserirAluno(Alunos *alunos);
void procurarAluno(Alunos *alunos);
void atualizarAluno(Alunos *alunos);
void removerAluno(Alunos *alunos);
void listarAlunos(Alunos *alunos);

#endif /* ALUNOS_H */
```

alunos.h

```
#define ERRO_ALUNO_NAO_EXISTE "O aluno não existe na lista."  
#define ERRO_LISTA_VAZIA "A lista de alunos está vazia."  
#define ERRO_LISTA_CHEIA "A lista de alunos está cheia."  
#define ERRO_ALUNO_EXISTE "O número de aluno já se encontra atribuído."
```

```
#ifndef ALUNOS_H  
#define ALUNOS_H  
  
#define MAX_ALUNOS 30  
  
#define ERRO_ALUNO_NAO_EXISTE "O aluno não existe na lista."  
#define ERRO_LISTA_VAZIA "A lista de alunos está vazia."  
#define ERRO_LISTA_CHEIA "A lista de alunos está cheia."  
#define ERRO_ALUNO_EXISTE "O número de aluno já se encontra atribuído."  
  
#define MAX_NUM_ALUNO 3  
#define MAX_NUM_ALUNO 1000  
#define MAX_DATA_NUM_ALUNO "Indica um número de aluno [0-1000]."  
  
#define MAX_NUM_ALUNO 30  
#define MAX_DATA_NUM "Indica o nome do aluno."  
  
#define MAX_DIA 3  
#define MAX_DIA 30  
#define DATA_DIA_MES "Indica o dia de nascimento."  
  
#define MAX_MES 3  
#define MAX_MES 12  
#define DATA_MES_ANO "Indica o mês de nascimento."  
  
#define MAX_ANO 1000  
#define MAX_ANO 2021  
#define DATA_ANO_MES "Indica o ano de nascimento."  
  
typedef struct {  
    int ano, mes, dia;  
} Data;  
  
typedef struct {  
    int numero;  
    char nome[MAX_NUM_ALUNO];  
    Data data_nascimento;  
} Aluno;  
  
typedef struct {  
    int contador;  
    Aluno alunos[MAX_ALUNOS];  
} Alunos;  
  
void inserirAluno(Alunos *alunos);  
void procurarAluno(Alunos *alunos);  
void atualizarAluno(Alunos *alunos);  
void removerAluno(Alunos *alunos);  
void listarAlunos(Alunos *alunos);  
  
#endif /* ALUNOS_H */
```


alunos.h

```
#define MIN_NUM_ALUNO      0
#define MAX_NUM_ALUNO      1000
#define MSG_OBTER_NUM_ALUNO "Insira um número de aluno [0-1000]: "

#define MAX_NOME_ALUNO      31
#define MSG_OBTER_NOME      "Insira o nome do aluno: "

#define MIN_DIA              1
#define MAX_DIA              31
#define OBTER_DIA_NASC       "Insira o dia de nascimento: "

#define MIN_MES              1
#define MAX_MES              12
#define OBTER_MES_NASC       "Insira o mês de nascimento: "

#define MIN_ANO              1990
#define MAX_ANO              2021
#define OBTER_ANO_NASC       "Insira o ano de nascimento: "
```

```
#ifndef ALUNOS_H
#define ALUNOS_H

#define MAX_ALUNOS  30

#define ERRO_ALUNO_NAO_EXISTE  "O aluno não existe na lista."
#define ERRO_LISTA_VAZIA      "A lista de alunos está vazia."
#define ERRO_LISTA_DUPLO      "A lista de alunos está cheia."
#define ERRO_ALUNO_EXISTE      "O número de aluno já se encontra atribuído."

#define MIN_NUM_ALUNO  1
#define MAX_NUM_ALUNO  1000
#define MSG_OBTER_NUM_ALUNO "Insira um número de aluno [0-1000]: "

#define MAX_NOME_ALUNO  31
#define MSG_OBTER_NOME  "Insira o nome do aluno: "

#define MIN_DIA  1
#define MAX_DIA  31
#define OBTER_DIA_NASC "Insira o dia de nascimento: "

#define MIN_MES  1
#define MAX_MES  12
#define OBTER_MES_NASC "Insira o mês de nascimento: "

#define MIN_ANO  1990
#define MAX_ANO  2021
#define OBTER_ANO_NASC "Insira o ano de nascimento: "

typedef struct {
    int ano, mes, dia;
} Data;

typedef struct {
    int numero;
    char nome[MAX_NOME_ALUNO];
    Data data_nascimento;
} Aluno;

typedef struct {
    int contador;
    Aluno alunos[MAX_ALUNOS];
} Alunos;

void inserirAluno(Alunos *alunos);
void procurarAluno(Alunos *alunos);
void atualizarAluno(Alunos *alunos);
void removerAluno(Alunos *alunos);
void listarAlunos(Alunos *alunos);

#endif /* ALUNOS_H */
```

alunos.h

```
typedef struct {
    int ano, mes, dia;
} Data;

typedef struct {
    int numero;
    char nome[MAX_NOME_ALUNO];
    Data data_nascimento;
} Aluno;

typedef struct {
    int contador;
    Aluno alunos[MAX_ALUNOS];
} Alunos;
```

```
#ifndef ALUNOS_H
#define ALUNOS_H

#define MAX_ALUNOS 10

#define ERRO_ALUNO_NAO_ENCONTRADO "O aluno não existe na lista."
#define ERRO_LISTA_VAZIA "A lista de alunos está vazia."
#define ERRO_LISTA_DUPLO "A lista de alunos está cheia."
#define ERRO_ALUNO_DUPLO "O número de aluno já se encontra atribuído."

#define MAX_NUM_ALUNO 1
#define MAX_NUM_ALUNO 1000
#define MAX_DATA_NUM_ALUNO "Insira um número de aluno (0-1000): "

#define MAX_NUM_ALUNO 10
#define MAX_DATA_NUM_ALUNO "Insira o nome do aluno: "

#define MAX_DIA 31
#define MAX_MES 12
#define DATA_DIA_MES "Insira o dia de nascimento: "

#define MAX_MES 12
#define MAX_MES 12
#define DATA_MES_MES "Insira o mês de nascimento: "

#define MAX_ANO 2000
#define MAX_ANO 2021
#define DATA_ANO_MES "Insira o ano de nascimento: "

typedef struct {
    int ano, mes, dia;
} Data;

typedef struct {
    int numero;
    char nome[MAX_NOME_ALUNO];
    Data data_nascimento;
} Aluno;

typedef struct {
    int contador;
    Aluno alunos[MAX_ALUNOS];
} Alunos;

void inserirAluno(Alunos *alunos);
void procurarAluno(Alunos *alunos);
void atualizarAluno(Alunos *alunos);
void removerAluno(Alunos *alunos);
void listarAlunos(Alunos *alunos);

#endif /* ALUNOS_H */
```

alunos.h

```
void inserirAlunos(Alunos *alunos);  
void procurarAlunos(Alunos alunos);  
void atualizarAlunos(Alunos *alunos);  
void removerAlunos(Alunos *alunos);  
void listarAlunos(Alunos alunos);
```

```
#ifndef ALUNOS_H  
#define ALUNOS_H  
  
#define MAX_ALUNOS 10  
  
#define ERRO_ALUNO_NAO_ENCONTRADO "O aluno não existe na lista."  
#define ERRO_LISTA_VAZIA "A lista de alunos está vazia."  
#define ERRO_LISTA_DUPLO "A lista de alunos está cheia."  
#define ERRO_ALUNO_DUPLO "O número do aluno já se encontra atribuído."  
  
#define MAX_NOME_ALUNO 40  
#define MAX_NUM_ALUNO 1000  
#define MAX_DATA_ALUNO "Indica um número de aluno [0-1000]:"  
  
#define MAX_NUM_ALUNO 10  
#define MAX_DATA_ALUNO "Indica o nome do aluno:"  
  
#define MAX_DIA 31  
#define MAX_MES 12  
#define DATA_DIA_MES "Indica o dia de nascimento:"  
  
#define MAX_MES 12  
#define MAX_DIA 31  
#define DATA_MES_DIA "Indica o mês de nascimento:"  
  
#define MAX_ANO 2000  
#define MAX_MES 12  
#define DATA_ANO_MES "Indica o ano de nascimento:"  
  
typedef struct {  
    int ano, mes, dia;  
} Data;  
  
typedef struct {  
    int numero;  
    char nome[MAX_NOME_ALUNO];  
    Data data_nascimento;  
} Aluno;  
  
typedef struct {  
    int contador;  
    Aluno alunos[MAX_ALUNOS];  
} Alunos;  
  
void inserirAlunos(Alunos *alunos);  
void procurarAlunos(Alunos alunos);  
void atualizarAlunos(Alunos *alunos);  
void removerAlunos(Alunos *alunos);  
void listarAlunos(Alunos alunos);  
  
#endif /* ALUNOS_H */
```

alunos.c

```
//alunos.c
#include <stdio.h>
#include <string.h>

#define "alunos.h"
#include "input.h"

void imprimeAluno(aluno aluno) {
    printf("Vetor 9: 900 92 94 96", aluno.numero, aluno.nome, aluno.data_nascimento.dia,
        aluno.data_nascimento.mes, aluno.data_nascimento.ano);
}

void registraAluno(aluno *aluno) {
    aluno.numero = 0;
    strcpy(aluno.nome, "");
    aluno.data_nascimento.dia = aluno.data_nascimento.mes = aluno.data_nascimento.ano = 0;
}

int procuraAluno(aluno aluno, int numero) {
    int i;
    for (i = 0; i < alunos.contador; i++) {
        if (aluno.aluno[i].numero == numero) {
            return i;
        }
    }
    return -1;
}

int insereAluno(aluno *aluno) {
    int numero = obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO, MSG_MEN_ALUNO);
    if (procuraAluno(*aluno, numero) == -1) {
        aluno.aluno[aluno.contador].numero = numero;
        strcpy(aluno.aluno[aluno.contador].nome, MEN_MEN_ALUNO, MSG_MEN_ALUNO);
        aluno.aluno[aluno.contador].data_nascimento.dia = obterInt(MEN_DIA, MEN_DIA,
            MSG_MEN_DIA);
        aluno.aluno[aluno.contador].data_nascimento.mes = obterInt(MEN_MES, MEN_MES,
            MSG_MEN_MES);
        aluno.aluno[aluno.contador].data_nascimento.ano = obterInt(MEN_ANO, MEN_ANO,
            MSG_MEN_ANO);
        return aluno.contador++;
    }
    return -1;
}

void atualizaAluno(aluno *aluno) {
    strcpy(*aluno, nome, MEN_MEN_ALUNO, MSG_MEN_ALUNO);
    (*aluno).data_nascimento.dia = obterInt(MEN_DIA, MEN_DIA, MSG_MEN_DIA);
    (*aluno).data_nascimento.mes = obterInt(MEN_MES, MEN_MES, MSG_MEN_MES);
    (*aluno).data_nascimento.ano = obterInt(MEN_ANO, MEN_ANO, MSG_MEN_ANO);
}

void listaAluno(aluno *aluno) {
    if (aluno.contador == MEN_MEN_ALUNO) {
        if (insereAluno(aluno) == -1) {
            puts(MEN_MEN_ALUNO, MSG_MEN);
        }
    } else {
        puts(MEN_MEN_ALUNO, MSG_MEN);
    }
}

void procuraAluno(aluno aluno) {
    int numero = procuraAluno(aluno, obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO,
        MSG_MEN_MEN_ALUNO));
    if (numero != -1) {
        imprimeAluno(aluno.aluno[numero]);
    } else {
        puts(MEN_MEN_ALUNO, MSG_MEN);
    }
}

void atualizaAluno(aluno *aluno) {
    int numero = procuraAluno(*aluno, obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO,
        MSG_MEN_MEN_ALUNO));
    if (numero != -1) {
        atualizaAluno(&aluno.aluno[numero]);
    } else {
        puts(MEN_MEN_ALUNO, MSG_MEN);
    }
}

void removeAluno(aluno *aluno) {
    int i, numero = procuraAluno(*aluno, obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO,
        MSG_MEN_MEN_ALUNO));
    if (numero != -1) {
        for (i = numero; i < alunos.contador - 1; i++) {
            alunos.aluno[i] = alunos.aluno[i + 1];
        }
        alunos.aluno[alunos.contador-1].numero = 0;
        alunos.contador--;
        puts(MEN_MEN_ALUNO, MSG_MEN);
    }
}

void listaAluno(aluno aluno) {
    if (aluno.contador == 0) {
        int i;
        for (i = 0; i < alunos.contador; i++) {
            imprimeAluno(aluno.aluno[i]);
        }
    } else {
        puts(MEN_MEN_ALUNO, MSG_MEN);
    }
}
```

alunos.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "alunos.h"
#include "input.h"
```

```
...
#include <stdio.h>
#include <string.h>
#include "alunos.h"
#include "input.h"

void lerArquivo(aluno *aluno) {
    printf("Vetor 9: 90-99-99", aluno.numero, aluno.nome, aluno.data_nascimento.dia,
        aluno.data_nascimento.mes, aluno.data_nascimento.ano);
}

void salvarArquivo(aluno *aluno) {
    aluno.numero = 0;
    strcpy(aluno.nome, "");
    aluno.data_nascimento.dia = aluno.data_nascimento.mes = aluno.data_nascimento.ano = 0;
}

int procurarArquivo(aluno *aluno, int numero) {
    int i;
    for (i = 0; i < aluno.contador; i++) {
        if (aluno.aluno[i].numero == numero) {
            return i;
        }
    }
    return -1;
}

int inserirArquivo(aluno *aluno) {
    int numero = obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO, MEN_MEN_ALUNO);
    if (procurarArquivo(*aluno, numero) == -1) {
        aluno.aluno[aluno.contador].numero = numero;
        lerString(aluno.aluno[aluno.contador].nome, MEN_MEN_ALUNO, MEN_MEN_ALUNO);
        aluno.aluno[aluno.contador].data_nascimento.dia = obterInt(MEN_DIA, MEN_DIA,
            MEN_DIA_MAX);
        aluno.aluno[aluno.contador].data_nascimento.mes = obterInt(MEN_MES, MEN_MES,
            MEN_MES_MAX);
        aluno.aluno[aluno.contador].data_nascimento.ano = obterInt(MEN_ANO, MEN_ANO,
            MEN_ANO_MAX);
        return aluno.contador++;
    }
    return -1;
}

void atualizarArquivo(aluno *aluno) {
    lerString(*aluno).nome, MEN_MEN_ALUNO, MEN_MEN_ALUNO);
    (*aluno).data_nascimento.dia = obterInt(MEN_DIA, MEN_DIA, MEN_DIA_MAX);
    (*aluno).data_nascimento.mes = obterInt(MEN_MES, MEN_MES, MEN_MES_MAX);
    (*aluno).data_nascimento.ano = obterInt(MEN_ANO, MEN_ANO, MEN_ANO_MAX);
}

void listarArquivo(aluno *aluno) {
    if (aluno.contador == MEN_MEN_ALUNO) {
        if (listarArquivo(aluno) == -1) {
            puts(MEN_MEN_ALUNO);
        }
    } else {
        puts(MEN_MEN_ALUNO);
    }
}

void procurarArquivo(aluno *aluno) {
    int numero = procurarArquivo(*aluno, obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO,
        MEN_MEN_ALUNO));
    if (numero != -1) {
        imprimeArquivo(aluno.aluno[numero]);
    } else {
        puts(MEN_MEN_ALUNO);
    }
}

void atualizarArquivo(aluno *aluno) {
    int numero = procurarArquivo(*aluno, obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO,
        MEN_MEN_ALUNO));
    if (numero != -1) {
        atualizarArquivo(&aluno.aluno[numero]);
    } else {
        puts(MEN_MEN_ALUNO);
    }
}

void removerArquivo(aluno *aluno) {
    int i, numero = procurarArquivo(*aluno, obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO,
        MEN_MEN_ALUNO));
    if (numero != -1) {
        for (i = numero; i < aluno.contador - 1; i++) {
            aluno.aluno[i] = aluno.aluno[i + 1];
        }
        salvarArquivo(aluno.aluno[i]);
        aluno.contador--;
    } else {
        puts(MEN_MEN_ALUNO);
    }
}

void listarArquivo(aluno *aluno) {
    if (aluno.contador == 0) {
        int i;
        for (i = 0; i < aluno.contador; i++) {
            imprimeArquivo(aluno.aluno[i]);
        }
    } else {
        puts(MEN_MEN_ALUNO);
    }
}
}
```

alunos.c

```
void imprimirAluno(Aluno aluno) {  
    printf("\n%3d %-30s %-d-%d-%d", aluno.numero, aluno.nome, aluno.data_nascimento.dia,  
        aluno.data_nascimento.mes, aluno.data_nascimento.ano);  
}
```

```
...  
#include <stdio.h>  
#include <string.h>  
#include "aluno.h"  
#include "liga.h"  
  
void ligarAluno(Aluno aluno) {  
    printf("Vetor 9: %d-%d-%d", aluno.numero, aluno.nome, aluno.data_nascimento.dia,  
        aluno.data_nascimento.mes, aluno.data_nascimento.ano);  
}  
  
void apagarAluno(Aluno aluno) {  
    aluno.contador = 0;  
    strcpy(aluno.nome, "");  
    aluno.data_nascimento.dia = aluno.data_nascimento.mes = aluno.data_nascimento.ano = 0;  
}  
  
int procurarAluno(Aluno aluno, int numero) {  
    int i;  
    for (i = 0; i < aluno.contador; i++) {  
        if (aluno.aluno[i].numero == numero) {  
            return i;  
        }  
    }  
    return -1;  
}  
  
int inserirAluno(Aluno *aluno) {  
    int numero = obterInt(MEM_NOME_ALUNO, MEM_NUM_ALUNO, MEM_DATA_NOME_ALUNO);  
    if (procurarAluno(*aluno, numero) == -1) {  
        aluno->aluno[aluno->contador].numero = numero;  
        strcpy(aluno->aluno[aluno->contador].nome, MEM_NUM_ALUNO, MEM_DATA_NOME);  
        aluno->aluno[aluno->contador].data_nascimento.dia = obterInt(MEM_DATA_NOME_ALUNO,  
            MEM_DATA_NOME);  
        aluno->aluno[aluno->contador].data_nascimento.mes = obterInt(MEM_DATA_NOME,  
            MEM_DATA_NOME);  
        aluno->aluno[aluno->contador].data_nascimento.ano = obterInt(MEM_DATA_NOME,  
            MEM_DATA_NOME);  
        return aluno->contador++;  
    }  
    return -1;  
}  
  
void atualizarAluno(Aluno *aluno) {  
    strcpy(aluno->aluno[aluno->contador].nome, MEM_NUM_ALUNO, MEM_DATA_NOME);  
    (*aluno->aluno[aluno->contador].data_nascimento.dia = obterInt(MEM_DATA_NOME,  
        MEM_DATA_NOME);  
    (*aluno->aluno[aluno->contador].data_nascimento.mes = obterInt(MEM_DATA_NOME,  
        MEM_DATA_NOME);  
    (*aluno->aluno[aluno->contador].data_nascimento.ano = obterInt(MEM_DATA_NOME,  
        MEM_DATA_NOME);  
}  
  
void listarAluno(Aluno *aluno) {  
    if (aluno->contador == MEM_NUM_ALUNO) {  
        if (listarAluno(aluno) == -1) {  
            puts(MEM_NUM_ALUNO_EXCULSO);  
        }  
    } else {  
        puts(MEM_DATA_NOME);  
    }  
}  
  
void procurarAluno(Aluno aluno) {  
    int numero = procurarAluno(aluno, obterInt(MEM_NUM_ALUNO, MEM_NUM_ALUNO,  
        MEM_DATA_NOME_ALUNO));  
    if (numero != -1) {  
        imprimeAluno(aluno.aluno[numero]);  
    } else {  
        puts(MEM_NUM_ALUNO_EXCULSO);  
    }  
}  
  
void atualizarAluno(Aluno *aluno) {  
    int numero = procurarAluno(*aluno, obterInt(MEM_NUM_ALUNO, MEM_NUM_ALUNO,  
        MEM_DATA_NOME_ALUNO));  
    if (numero != -1) {  
        atualizarAluno(&aluno->aluno[numero]);  
    } else {  
        puts(MEM_NUM_ALUNO_EXCULSO);  
    }  
}  
  
void removerAluno(Aluno *aluno) {  
    int i, numero = procurarAluno(*aluno, obterInt(MEM_NUM_ALUNO, MEM_NUM_ALUNO,  
        MEM_DATA_NOME_ALUNO));  
    if (numero != -1) {  
        for (i = numero; i < aluno->contador - 1; i++) {  
            aluno->aluno[i] = aluno->aluno[i + 1];  
        }  
        apagarAluno(aluno->aluno[aluno->contador - 1]);  
        aluno->contador--;  
    } else {  
        puts(MEM_NUM_ALUNO_EXCULSO);  
    }  
}  
  
void listarAluno(Aluno aluno) {  
    if (aluno.contador == 0) {  
        int i;  
        for (i = 0; i < aluno.contador; i++) {  
            imprimeAluno(aluno.aluno[i]);  
        }  
    } else {  
        puts(MEM_DATA_NOME);  
    }  
}
```

alunos.c

```
void apagarDadosAluno(Aluno *aluno) {  
  
    aluno->numero = 0;  
  
    strcpy(aluno->nome, "");  
  
    aluno->data_nascimento.dia = aluno->data_nascimento.mes = aluno->data_nascimento.ano = 0;  
  
}
```

```
...  
#include <stdio.h>  
#include <string.h>  
  
#define "alunos.h"  
#include "input.h"  
  
void listarAlunos(Aluno alunos) {  
    printf("\nListar Alunos:");  
    printf("\nNome: %s", alunos.nome);  
    printf("\nData Nascimento: %d/%d/%d", alunos.data_nascimento.dia, alunos.data_nascimento.mes, alunos.data_nascimento.ano);  
}  
  
void apagarDadosAluno(Aluno *aluno) {  
    aluno->numero = 0;  
    strcpy(aluno->nome, "");  
    aluno->data_nascimento.dia = aluno->data_nascimento.mes = aluno->data_nascimento.ano = 0;  
}  
  
int procurarAluno(Aluno alunos, int numero) {  
    int i;  
    for (i = 0; i < alunos.contador; i++) {  
        if (alunos.alunos[i].numero == numero) {  
            return i;  
        }  
    }  
    return -1;  
}  
  
int inserirAluno(Aluno *alunos) {  
    int numero = obterInt("Digite o numero do aluno:");  
    if (procurarAluno(*alunos, numero) == -1) {  
        alunos->alunos[alunos->contador].numero = numero;  
        listarAlunos(alunos->alunos->contador);  
        alunos->alunos[alunos->contador].data_nascimento.dia = obterInt("Dia do Nascimento:");  
        alunos->alunos[alunos->contador].data_nascimento.mes = obterInt("Mes do Nascimento:");  
        alunos->alunos[alunos->contador].data_nascimento.ano = obterInt("Ano do Nascimento:");  
        return alunos->contador++;  
    }  
    return -1;  
}  
  
void atualizarAluno(Aluno *alunos) {  
    listarAlunos(*alunos);  
    int i = obterInt("Digite o numero do aluno a ser atualizado:");  
    if (i < 0 || i > alunos->contador) {  
        printf("Numero invalido!\n");  
        return;  
    }  
    alunos->alunos[i].numero = obterInt("Digite o novo numero:");  
    alunos->alunos[i].data_nascimento.dia = obterInt("Dia do Nascimento:");  
    alunos->alunos[i].data_nascimento.mes = obterInt("Mes do Nascimento:");  
    alunos->alunos[i].data_nascimento.ano = obterInt("Ano do Nascimento:");  
}  
  
void excluirAluno(Aluno *alunos) {  
    if (alunos->contador == 0) {  
        printf("Nenhum aluno cadastrado!\n");  
        return;  
    }  
    int i = obterInt("Digite o numero do aluno a ser excluido:");  
    if (i < 0 || i > alunos->contador) {  
        printf("Numero invalido!\n");  
        return;  
    }  
    for (i = i; i < alunos->contador - 1; i++) {  
        alunos->alunos[i] = alunos->alunos[i + 1];  
    }  
    alunos->contador--;  
}  
  
void menuPrincipal() {  
    int i, numero = 0;  
    for (i = 0; i < MAX_ALUNOS; i++) {  
        alunos->alunos[i].numero = 0;  
        alunos->alunos[i].data_nascimento.dia = 0;  
        alunos->alunos[i].data_nascimento.mes = 0;  
        alunos->alunos[i].data_nascimento.ano = 0;  
    }  
    printf("Menu Principal:\n");  
    printf("1 - Inserir Aluno\n");  
    printf("2 - Atualizar Aluno\n");  
    printf("3 - Excluir Aluno\n");  
    printf("4 - Listar Alunos\n");  
    printf("5 - Apagar Dados Aluno\n");  
    printf("6 - Sair\n");  
    printf("Digite o numero da opcao:");  
    numero = obterInt("Digite o numero da opcao:");  
    switch (numero) {  
        case 1:  
            inserirAluno(alunos);  
            break;  
        case 2:  
            atualizarAluno(alunos);  
            break;  
        case 3:  
            excluirAluno(alunos);  
            break;  
        case 4:  
            listarAlunos(alunos);  
            break;  
        case 5:  
            apagarDadosAluno(alunos);  
            break;  
        case 6:  
            printf("Programa encerrado!\n");  
            return;  
        default:  
            printf("Opcao invalida!\n");  
            break;  
    }  
    menuPrincipal();  
}
```

alunos.c

```
int procurarAluno(Alunos alunos, int numero) {
    int i;
    for (i = 0; i < alunos.contador; i++) {
        if (alunos.alunos[i].numero == numero) {
            return i;
        }
    }
    return -1;
}
```

[illegible]

alunos.c

```
int inserirAluno(Alunos *alunos) {
    int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);

    if (procurarAluno(*alunos, numero) == -1) {

        alunos->alunos[alunos->contador].numero = numero;

        lerString(alunos->alunos[alunos->contador].nome, MAX_NOME_ALUNO, MSG_OBTEN_NOME);

        alunos->alunos[alunos->contador].data_nascimento.dia =
            obterInt(MIN_DIA, MAX_DIA, OBTEN_DIA_NASC);
        alunos->alunos[alunos->contador].data_nascimento.mes =
            obterInt(MIN_MES, MAX_MES, OBTEN_MES_NASC);
        alunos->alunos[alunos->contador].data_nascimento.ano =
            obterInt(MIN_ANO, MAX_ANO, OBTEN_ANO_NASC);

        return alunos->contador++;
    }
    return -1;
}
```

```
...
#include <stdio.h>
#include <string.h>
#include "alunos.h"
#include "input.h"

void lerString(aluno *aluno) {
    printf("Digite o nome do aluno: ");
    fgets(aluno->nome, MAX_NOME_ALUNO, stdin);
}

void obterInt(aluno *aluno, int *numero) {
    int i = 0;
    while (i < MAX_NUM_ALUNO) {
        printf("Digite o numero do aluno: ");
        fgets(aluno->num_str, MAX_NUM_ALUNO, stdin);
        sscanf(aluno->num_str, "%d", numero);
        i++;
    }
}

int procurarAluno(Alunos *alunos, int numero) {
    int i;
    for (i = 0; i < alunos->contador; i++) {
        if (alunos->alunos[i].numero == numero) {
            return i;
        }
    }
    return -1;
}

int inserirAluno(Alunos *alunos) {
    int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);

    if (procurarAluno(*alunos, numero) == -1) {
        alunos->alunos[alunos->contador].numero = numero;

        lerString(alunos->alunos[alunos->contador].nome, MAX_NOME_ALUNO, MSG_OBTEN_NOME);

        alunos->alunos[alunos->contador].data_nascimento.dia = obterInt(MIN_DIA, MAX_DIA,
            OBTEN_DIA_NASC);
        alunos->alunos[alunos->contador].data_nascimento.mes = obterInt(MIN_MES, MAX_MES,
            OBTEN_MES_NASC);
        alunos->alunos[alunos->contador].data_nascimento.ano = obterInt(MIN_ANO, MAX_ANO,
            OBTEN_ANO_NASC);

        return alunos->contador++;
    }
    return -1;
}

void atualizarAluno(Alunos *alunos) {
    lerString(alunos->alunos[alunos->contador].nome, MAX_NOME_ALUNO, MSG_OBTEN_NOME);
    (*alunos->alunos[alunos->contador].data_nascimento.dia = obterInt(MIN_DIA, MAX_DIA,
        OBTEN_DIA_NASC);
    (*alunos->alunos[alunos->contador].data_nascimento.mes = obterInt(MIN_MES, MAX_MES,
        OBTEN_MES_NASC);
    (*alunos->alunos[alunos->contador].data_nascimento.ano = obterInt(MIN_ANO, MAX_ANO,
        OBTEN_ANO_NASC);
}

void removerAluno(Alunos *alunos) {
    if (alunos->contador == 0) {
        printf("Nenhum aluno encontrado.\n");
    } else {
        printf("Digite o numero do aluno a ser removido: ");
        fgets(alunos->num_str, MAX_NUM_ALUNO, stdin);
        sscanf(alunos->num_str, "%d", &numero);

        int i;
        for (i = 0; i < alunos->contador; i++) {
            if (alunos->alunos[i].numero == numero) {
                alunos->alunos[i] = alunos->alunos[alunos->contador - 1];
                alunos->contador--;
            }
        }
        printf("Aluno removido com sucesso.\n");
    }
}

void listarAlunos(Alunos *alunos) {
    if (alunos->contador == 0) {
        printf("Nenhum aluno cadastrado.\n");
    } else {
        printf("Lista de alunos cadastrados:\n");
        for (i = 0; i < alunos->contador; i++) {
            printf("Aluno %d: Nome: %s, Dia: %d, Mes: %d, Ano: %d\n", i+1,
                alunos->alunos[i].nome, alunos->alunos[i].data_nascimento.dia,
                alunos->alunos[i].data_nascimento.mes, alunos->alunos[i].data_nascimento.ano);
        }
    }
}
```

alunos.c

```
void atualizarAluno(Aluno *aluno) {  
    lerString((*aluno).nome, MAX_NOME_ALUNO, MSG_OBTEN_NOME);  
    (*aluno).data_nascimento.dia = obterInt(MIN_DIA, MAX_DIA, OBTEN_DIA_NASC);  
    (*aluno).data_nascimento.mes = obterInt(MIN_MES, MAX_MES, OBTEN_MES_NASC);  
    (*aluno).data_nascimento.ano = obterInt(MIN_ANO, MAX_ANO, OBTEN_ANO_NASC);  
}
```

```
...  
#include <stdio.h>  
#include <string.h>  
#include "alunos.h"  
#include "input.h"  
  
void lerStringAluno(Aluno *aluno) {  
    printf("Digite o nome do aluno: ");  
    fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
    while (aluno->nome[0] == '\n')  
        fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
}  
  
void atualizarAluno(Aluno *aluno) {  
    int i = 0;  
    while (i < MAX_ALUNOS) {  
        printf("Atualizar aluno %d: ", i + 1);  
        if (i % 10 == 0) printf("\n");  
        lerStringAluno(aluno + i);  
        i++;  
    }  
}  
  
int inserirAluno(Aluno *alunos) {  
    int i = 0;  
    while (i < MAX_ALUNOS) {  
        printf("Inserir novo aluno: ");  
        if (i % 10 == 0) printf("\n");  
        lerStringAluno(aluno + i);  
        i++;  
    }  
}  
  
void atualizarAluno(Aluno *alunos) {  
    int i = 0;  
    while (i < MAX_ALUNOS) {  
        printf("Atualizar aluno %d: ", i + 1);  
        if (i % 10 == 0) printf("\n");  
        lerStringAluno(aluno + i);  
        i++;  
    }  
}  
  
void lerAluno(Aluno *aluno) {  
    printf("Digite o nome do aluno: ");  
    fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
    while (aluno->nome[0] == '\n')  
        fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
}  
  
void atualizarAluno(Aluno *alunos) {  
    int i = 0;  
    while (i < MAX_ALUNOS) {  
        printf("Atualizar aluno %d: ", i + 1);  
        if (i % 10 == 0) printf("\n");  
        lerStringAluno(aluno + i);  
        i++;  
    }  
}  
  
void lerAluno(Aluno *aluno) {  
    printf("Digite o nome do aluno: ");  
    fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
    while (aluno->nome[0] == '\n')  
        fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
}  
  
void atualizarAluno(Aluno *alunos) {  
    int i = 0;  
    while (i < MAX_ALUNOS) {  
        printf("Atualizar aluno %d: ", i + 1);  
        if (i % 10 == 0) printf("\n");  
        lerStringAluno(aluno + i);  
        i++;  
    }  
}  
  
void lerAluno(Aluno *aluno) {  
    printf("Digite o nome do aluno: ");  
    fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
    while (aluno->nome[0] == '\n')  
        fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
}  
  
void atualizarAluno(Aluno *alunos) {  
    int i = 0;  
    while (i < MAX_ALUNOS) {  
        printf("Atualizar aluno %d: ", i + 1);  
        if (i % 10 == 0) printf("\n");  
        lerStringAluno(aluno + i);  
        i++;  
    }  
}  
  
void lerAluno(Aluno *aluno) {  
    printf("Digite o nome do aluno: ");  
    fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
    while (aluno->nome[0] == '\n')  
        fgets(aluno->nome, MAX_NOME_ALUNO, stdin);  
}  
  
void atualizarAluno(Aluno *alunos) {  
    int i = 0;  
    while (i < MAX_ALUNOS) {  
        printf("Atualizar aluno %d: ", i + 1);  
        if (i % 10 == 0) printf("\n");  
        lerStringAluno(aluno + i);  
        i++;  
    }  
}
```

alunos.c

```
void inserirAlunos(Alunos *alunos) {
    if (alunos->contador < MAX_ALUNOS) {
        if (inserirAluno(alunos) == -1) {
            puts(ERRO_ALUNO_EXISTE);
        }
    } else {
        puts(ERRO_LISTA_CHEIA);
    }
}
```

```
...
#include <stdio.h>
#include <string.h>

#define "alunos.h"
#include "input.h"

void listarAlunos(Alunos *alunos) {
    printf("\nList 9:00-10:00", alunos->nome, alunos->nome, alunos->data_nascimento.dia,
        alunos->data_nascimento.mes, alunos->data_nascimento.ano);
}

void apagarAlunos(Alunos *alunos) {
    alunos->contador = 0;
    printf(alunos->nome, "\n");
    alunos->data_nascimento.dia = alunos->data_nascimento.mes = alunos->data_nascimento.ano = 0;
}

int procurarAlunos(Alunos *alunos, int numero) {
    int i;
    for (i = 0; i < alunos->contador; i++) {
        if (alunos->alunos[i].numero == numero) {
            return i;
        }
    }
    return -1;
}

int inserirAlunos(Alunos *alunos) {
    int numero = obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO, MEN_MEN_ALUNO);
    if (procurarAlunos(*alunos, numero) == -1) {
        alunos->alunos[alunos->contador].numero = numero;
        listarAlunos(alunos->alunos->contador);
        alunos->alunos[alunos->contador].data_nascimento.dia = obterInt(MEN_DIA, MEN_DIA,
            MEN_DIA);
        alunos->alunos[alunos->contador].data_nascimento.mes = obterInt(MEN_MES, MEN_MES,
            MEN_MES);
        alunos->alunos[alunos->contador].data_nascimento.ano = obterInt(MEN_ANO, MEN_ANO,
            MEN_ANO);
        return alunos->contador++;
    }
    return -1;
}

void atualizarAlunos(Alunos *alunos) {
    listarAlunos(alunos->nome, MEN_MEN_ALUNO, MEN_MEN_ALUNO);
    (*alunos->data_nascimento.dia = obterInt(MEN_DIA, MEN_DIA, MEN_DIA);
    (*alunos->data_nascimento.mes = obterInt(MEN_MES, MEN_MES, MEN_MES);
    (*alunos->data_nascimento.ano = obterInt(MEN_ANO, MEN_ANO, MEN_ANO);
}

void inserirAlunos(Alunos *alunos) {
    if (alunos->contador < MAX_ALUNOS) {
        if (inserirAluno(alunos) == -1) {
            puts(ERRO_ALUNO_EXISTE);
        }
    } else {
        puts(ERRO_LISTA_CHEIA);
    }
}

void procurarAlunos(Alunos *alunos) {
    int numero = procurarAlunos(alunos, obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO,
        MEN_MEN_ALUNO));
    if (numero != -1) {
        imprimirAlunos(alunos->alunos[numero]);
        puts(ERRO_ALUNO_NAO_ENCONTRADO);
    }
}

void atualizarAlunos(Alunos *alunos) {
    int numero = procurarAlunos(*alunos, obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO,
        MEN_MEN_ALUNO));
    if (numero != -1) {
        atualizarAlunos(&alunos->alunos[numero]);
        puts(ERRO_ALUNO_NAO_ENCONTRADO);
    }
}

void removerAlunos(Alunos *alunos) {
    int i, numero = procurarAlunos(*alunos, obterInt(MEN_MEN_ALUNO, MEN_MEN_ALUNO,
        MEN_MEN_ALUNO));
    if (numero != -1) {
        for (i = numero; i < alunos->contador - 1; i++) {
            alunos->alunos[i] = alunos->alunos[i + 1];
        }
        apagarAlunos(alunos->alunos->contador-1);
        alunos->contador--;
        puts(ERRO_ALUNO_NAO_ENCONTRADO);
    }
}

void listarAlunos(Alunos *alunos) {
    if (alunos->contador > 0) {
        int i;
        for (i = 0; i < alunos->contador; i++) {
            imprimirAlunos(alunos->alunos[i]);
        }
    } else {
        puts(ERRO_LISTA_VAZIA);
    }
}
}
```

alunos.c

```
void procurarAlunos(Alunos alunos) {
    int numero = procurarAluno(alunos,
                                obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO));

    if (numero != -1) {
        imprimirAluno(alunos.alunos[numero]);
    } else {
        puts(ERRO_ALUNO_NAO_EXISTE);
    }
}
```

```
...
#include <stdio.h>
#include <string.h>
#include "alunos.h"
#include "input.h"

void imprimirAluno(Aluno aluno) {
    printf("ID: %d Nome: %s\n", aluno.id, aluno.nome);
    printf("Data de Nascimento: %s\n", aluno.data_nascimento);
}

void procurarAluno(Alunos *alunos) {
    int numero = -1;
    if (alunos->alunos != NULL) {
        for (int i = 0; i < alunos->contador; i++) {
            if (alunos->alunos[i].numero == numero) {
                return i;
            }
        }
    }
    return -1;
}

int inserirAluno(Alunos *alunos) {
    int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
    if (procurarAluno(*alunos, numero) == -1) {
        alunos->alunos[alunos->contador].numero = numero;
        strcpy(alunos->alunos[alunos->contador].nome, MSG_OBTEN_NOME);
        alunos->alunos[alunos->contador].data_nascimento = obterData(MIN_DATA, MAX_DATA,
                                                                    MSG_OBTEN_DATA);
        alunos->alunos[alunos->contador].data_nascimento = obterData(MIN_DATA, MAX_DATA,
                                                                    MSG_OBTEN_DATA);
        alunos->alunos[alunos->contador].data_nascimento = obterData(MIN_DATA, MAX_DATA,
                                                                    MSG_OBTEN_DATA);
        return alunos->contador++;
    }
    return -1;
}

void atualizarAluno(Alunos *alunos) {
    int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
    if (procurarAluno(*alunos, numero) == -1) {
        puts(ERRO_ALUNO_NAO_EXISTE);
    } else {
        strcpy(alunos->alunos[numero].nome, MSG_OBTEN_NOME);
        alunos->alunos[numero].data_nascimento = obterData(MIN_DATA, MAX_DATA,
                                                            MSG_OBTEN_DATA);
        alunos->alunos[numero].data_nascimento = obterData(MIN_DATA, MAX_DATA,
                                                            MSG_OBTEN_DATA);
    }
}

void listarAlunos(Alunos *alunos) {
    if (alunos->contador == 0) {
        puts(ERRO_ALUNO_NAO_EXISTE);
    } else {
        printf("Lista de Alunos:\n");
        for (int i = 0; i < alunos->contador; i++) {
            imprimirAluno(alunos->alunos[i]);
        }
    }
}

void removerAluno(Alunos *alunos) {
    int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
    if (procurarAluno(*alunos, numero) == -1) {
        puts(ERRO_ALUNO_NAO_EXISTE);
    } else {
        for (int i = 0; i < alunos->contador - 1; i++) {
            alunos->alunos[i] = alunos->alunos[i + 1];
        }
        alunos->contador--;
    }
}

void listarAlunos(Alunos *alunos) {
    if (alunos->contador == 0) {
        puts(ERRO_ALUNO_NAO_EXISTE);
    } else {
        printf("Lista de Alunos:\n");
        for (int i = 0; i < alunos->contador; i++) {
            imprimirAluno(alunos->alunos[i]);
        }
    }
}
```

alunos.c

```
void atualizarAlunos(Alunos *alunos) {
    int numero = procurarAluno(*alunos,
                                obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO));

    if (numero != -1) {
        atualizarAluno(&(*alunos).alunos[numero]);
    } else {
        puts(ERRO_ALUNO_NAO_EXISTE);
    }
}
```

```
...
#include <stdio.h>
#include <string.h>
#include "alunos.h"
#include "input.h"

void listarAlunos(Alunos alunos) {
    printf("Lista de alunos: %d", alunos.numero, alunos.nome, alunos.data_nascimento_dia,
        alunos.data_nascimento_mes, alunos.data_nascimento_ano);
}

void atualizarAlunos(Alunos *alunos) {
    int numero = 0;
    if (alunos.numero == 0) {
        alunos.data_nascimento_dia = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
    }

    int procurarAluno(Alunos alunos, int numero) {
        int i;
        for (i = 0; i < alunos.contador; i++) {
            if (alunos.alunos[i].numero == numero) {
                return i;
            }
        }
        return -1;
    }

    int inserirAlunos(Alunos *alunos) {
        int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
        if (procurarAluno(*alunos, numero) == -1) {
            alunos.alunos[alunos.contador].numero = numero;
            listarAlunos(alunos);
            alunos.alunos[alunos.contador].data_nascimento_dia = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
            alunos.alunos[alunos.contador].data_nascimento_mes = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
            alunos.alunos[alunos.contador].data_nascimento_ano = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
            return alunos.contador++;
        }
        return -1;
    }

    void atualizarAlunos(Alunos *alunos) {
        listarAlunos(*alunos);
        if (alunos.numero == 0) {
            printf("Atualizar aluno: ");
            (*alunos).data_nascimento_dia = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
            (*alunos).data_nascimento_mes = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
            (*alunos).data_nascimento_ano = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
        }
    }

    void inserirAlunos(Alunos *alunos) {
        if (alunos.contador == MAX_NUM_ALUNO) {
            printf("Erro: limite de alunos atingido.");
        } else {
            puts(ERRO_ALUNO_NAO_EXISTE);
        }
    }

    void procurarAlunos(Alunos alunos) {
        int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
        if (numero != -1) {
            atualizarAlunos(&alunos.alunos[numero]);
        } else {
            puts(ERRO_ALUNO_NAO_EXISTE);
        }
    }

    void atualizarAlunos(Alunos *alunos) {
        int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
        if (numero != -1) {
            atualizarAlunos(&alunos.alunos[numero]);
        } else {
            puts(ERRO_ALUNO_NAO_EXISTE);
        }
    }

    void removerAlunos(Alunos *alunos) {
        int i, numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
        if (numero != -1) {
            for (i = numero; i < alunos.contador - 1; i++) {
                alunos.alunos[i] = alunos.alunos[i + 1];
            }
            alunos.alunos[alunos.contador - 1].numero = 0;
            alunos.contador--;
            puts(ERRO_ALUNO_NAO_EXISTE);
        }
    }

    void listarAlunos(Alunos alunos) {
        if (alunos.contador == 0) {
            int i;
            for (i = 0; i < alunos.contador; i++) {
                printf("Aluno %d: ", i);
            }
        } else {
            puts(ERRO_ALUNO_NAO_EXISTE);
        }
    }
}
```

alunos.c

```
void removerAlunos(Alunos *alunos) {
    int i, numero = procurarAluno(*alunos,
                                obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO));

    if (numero != -1) {
        for (i = numero; i < alunos->contador - 1; i++) {
            alunos->alunos[i] = alunos->alunos[i + 1];
        }

        apagarDadosAluno(&alunos->alunos[i]);

        alunos->contador--;
    } else {
        puts(ERRO_ALUNO_NAO_EXISTE);
    }
}
```

```
...
#include <stdio.h>
#include <string.h>

#define 'alunos'
#define 'input'

void lerDadosAluno(Aluno aluno) {
    printf("Informe o nome do aluno: ");
    scanf("%s", aluno.nome);
    printf("Informe o número do aluno: ");
    scanf("%d", &aluno.numero);
}

void apagarDadosAluno(Aluno *aluno) {
    aluno->numero = 0;
    strcpy(aluno->nome, "");
    aluno->data_nascimento.dia = 0;
    aluno->data_nascimento.mes = 0;
    aluno->data_nascimento.ano = 0;
}

int procurarAluno(Alunos *alunos, int numero) {
    int i;
    for (i = 0; i < alunos->contador; i++) {
        if (alunos->alunos[i].numero == numero) {
            return i;
        }
    }
    return -1;
}

int inserirAluno(Alunos *alunos) {
    int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
    if (procurarAluno(*alunos, numero) == -1) {
        alunos->alunos[alunos->contador].numero = numero;
        lerDadosAluno(&alunos->alunos[alunos->contador]);
        alunos->alunos[alunos->contador].data_nascimento.dia = obterInt(DIA_MIN, DIA_MAX, MSG_OBTEN_DIA_MIN);
        alunos->alunos[alunos->contador].data_nascimento.mes = obterInt(MES_MIN, MES_MAX, MSG_OBTEN_MES_MIN);
        alunos->alunos[alunos->contador].data_nascimento.ano = obterInt(ANO_MIN, ANO_MAX, MSG_OBTEN_ANO_MIN);
        return alunos->contador++;
    }
    return -1;
}

void atualizarAluno(Alunos *alunos) {
    lerDadosAluno(*alunos);
    (*alunos).data_nascimento.dia = obterInt(DIA_MIN, DIA_MAX, MSG_OBTEN_DIA_MIN);
    (*alunos).data_nascimento.mes = obterInt(MES_MIN, MES_MAX, MSG_OBTEN_MES_MIN);
    (*alunos).data_nascimento.ano = obterInt(ANO_MIN, ANO_MAX, MSG_OBTEN_ANO_MIN);
}

void listarAlunos(Alunos *alunos) {
    if (alunos->contador == 0) {
        printf("Nenhum aluno cadastrado.\n");
    } else {
        printf("Lista de alunos cadastrados:\n");
    }
}

void procurarAluno(Alunos *alunos) {
    int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
    if (numero != -1) {
        lerDadosAluno(&alunos->alunos[numero]);
        printf("Dados do aluno %d:\n", numero);
    }
}

void atualizarAluno(Alunos *alunos) {
    int numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
    if (numero != -1) {
        atualizarAluno(&alunos->alunos[numero]);
    }
}

void removerAlunos(Alunos *alunos) {
    int i, numero = obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO, MSG_OBTEN_NUM_ALUNO);
    if (numero != -1) {
        for (i = numero; i < alunos->contador - 1; i++) {
            alunos->alunos[i] = alunos->alunos[i + 1];
        }
        apagarDadosAluno(&alunos->alunos[i]);
        alunos->contador--;
    } else {
        puts(ERRO_ALUNO_NAO_EXISTE);
    }
}

void listarAlunos(Alunos *alunos) {
    if (alunos->contador == 0) {
        printf("Nenhum aluno cadastrado.\n");
    } else {
        printf("Lista de alunos cadastrados:\n");
    }
}
}
```

alunos.c

```
void listarAlunos(Alunos alunos) {
    if (alunos.contador > 0) {
        int i;
        for (i = 0; i < alunos.contador; i++) {
            imprimirAluno(alunos.alunos[i]);
        }
    } else {
        puts(ERRO_LISTA_VAZIA);
    }
}
```

```
...
#include <stdio.h>
#include <string.h>

#define "alunos.h"
#include "input.h"

void listarAlunos(Alunos alunos) {
    printf("Vetor 0-999: ");
    printf("Nome: %s", alunos.nomes);
    printf("Idade: %d", alunos.idades);
    printf("Sexo: %s", alunos.sexos);
    printf("Data de Nascimento: %s", alunos.data_nascimento);
}

void imprimirAluno(Aluno *aluno) {
    printf("Nome: %s", aluno->nomes);
    printf("Idade: %d", aluno->idades);
    printf("Sexo: %s", aluno->sexos);
    printf("Data de Nascimento: %s", aluno->data_nascimento);
}

int procurarAluno(Alunos alunos, int numero) {
    int i;
    for (i = 0; i < alunos.contador; i++) {
        if (alunos.alunos[i].numero == numero) {
            return i;
        }
    }
    return -1;
}

int inserirAluno(Alunos *alunos) {
    int numero = obterInt("Insira o numero do aluno: ");
    if (procurarAluno(*alunos, numero) == -1) {
        alunos->alunos[alunos->contador].numero = numero;
        listarAlunos(*alunos);
        alunos->alunos[alunos->contador].data_nascimento = obterInt("Data de Nascimento: ");
        alunos->alunos[alunos->contador].idades = obterInt("Idade: ");
        alunos->alunos[alunos->contador].sexos = obterInt("Sexo: ");
        alunos->alunos[alunos->contador].data_nascimento = obterInt("Data de Nascimento: ");
        return alunos->contador++;
    }
    return -1;
}

void atualizarAluno(Alunos *alunos) {
    listarAlunos(*alunos);
    int i = obterInt("Insira o numero do aluno a ser atualizado: ");
    if (i == -1) {
        printf("Numero invalido!\n");
    } else {
        alunos->alunos[i].nomes = obterString("Novo nome: ");
        alunos->alunos[i].idades = obterInt("Nova idade: ");
        alunos->alunos[i].sexos = obterInt("Novo sexo: ");
        alunos->alunos[i].data_nascimento = obterInt("Nova data de nascimento: ");
    }
}

void excluirAluno(Alunos *alunos) {
    int i = obterInt("Insira o numero do aluno a ser excluido: ");
    if (i == -1) {
        printf("Numero invalido!\n");
    } else {
        alunos->alunos[i].numero = -1;
    }
}

void procurarAluno(Alunos alunos) {
    int numero = obterInt("Insira o numero do aluno a ser procurado: ");
    if (numero == -1) {
        printf("Numero invalido!\n");
    } else {
        int i = procurarAluno(alunos, numero);
        if (i != -1) {
            printf("Aluno encontrado!\n");
        } else {
            printf("Aluno nao encontrado!\n");
        }
    }
}

void atualizarAluno(Alunos *alunos) {
    int numero = obterInt("Insira o numero do aluno a ser atualizado: ");
    if (numero == -1) {
        printf("Numero invalido!\n");
    } else {
        alunos->alunos[numero].nomes = obterString("Novo nome: ");
        alunos->alunos[numero].idades = obterInt("Nova idade: ");
        alunos->alunos[numero].sexos = obterInt("Novo sexo: ");
        alunos->alunos[numero].data_nascimento = obterInt("Nova data de nascimento: ");
    }
}

void excluirAluno(Alunos *alunos) {
    int i = obterInt("Insira o numero do aluno a ser excluido: ");
    if (i == -1) {
        printf("Numero invalido!\n");
    } else {
        alunos->alunos[i].numero = -1;
    }
}

void listarAlunos(Alunos alunos) {
    if (alunos.contador == 0) {
        printf("Lista vazia!\n");
    } else {
        int i;
        for (i = 0; i < alunos.contador; i++) {
            imprimirAluno(alunos.alunos[i]);
        }
    }
}
```

main.c

```
#include <stdio.h>
#include "alunos.h"
#include "input.h"

int main() {
    int opcao;
    Aluno aluno = {contador = 0};

    do {
        printf("Alunos .....");
        printf("\n - Insere");
        printf("\n - Remove");
        printf("\n - Visualiza");
        printf("\n - Busca");
        printf("\n - Lista");
        printf("\n - Sair");
        printf("\n .....");
        printf("Alunos: %d/%d", aluno.contador, MAX_ALUNOS);

        printf("\nOpcao: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 0:
                break;
            case 1:
                insereAluno(&aluno);
                break;
            case 2:
                removeAluno(&aluno);
                break;
            case 3:
                visualizaAlunos(&aluno);
                break;
            case 4:
                buscaAluno(&aluno);
                break;
            case 5:
                listaAlunos(&aluno);
                break;
            default:
                printf("Opcao invalida");
        }
    } while (opcao != 0);

    return 0;
}
```


main.c

```
#include <stdio.h>
```

```
#include "alunos.h"
```

```
#include "input.h"
```

```
#include <stdio.h>
#include "alunos.h"
#include "input.h"

int main() {
    int opcao;
    Alunos alunos = {0, contador = 0};

    do {
        printf("Alunos .....");
        printf("\n - Insere");
        printf("\n - Remover");
        printf("\n - Visualizar");
        printf("\n - Salvar");
        printf("\n - Listar");
        printf("\n - Sair");
        printf("\n .....");

        printf("Alunos: %d/%d", alunos.contador, MAX_ALUNOS);

        printf("\nOpcao: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 0:
                break;
            case 1:
                inserirAlunos(&alunos);
                break;
            case 2:
                removerAlunos(&alunos);
                break;
            case 3:
                visualizarAlunos(&alunos);
                break;
            case 4:
                salvarAlunos(&alunos);
                break;
            case 5:
                listarAlunos(&alunos);
                break;
            default:
                printf("\nOpcao invalida");
        }
    } while (opcao != 0);

    return 0;
}
```

main.c

```
int main() {
    int opcao;
    Alunos alunos = {.contador = 0};

    ...

    return 0;
}
```

```
#include <stdio.h>
#include "alunos.h"
#include "opcao.h"

int main() {
    int opcao;
    Alunos alunos = {.contador = 0};

    do {
        printf("alunos: .....");
        printf("\n - inserir");
        printf("\n - remover");
        printf("\n - atualizar");
        printf("\n - buscar");
        printf("\n - listar");
        printf("\n - sair");
        printf("\n .....");
        printf("alunos: %d/%d, alunos.contador: %d, MAX_ALUNOS);

        printf("opcao: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 0:
                break;
            case 1:
                inserirAlunos(&alunos);
                break;
            case 2:
                removerAlunos(&alunos);
                break;
            case 3:
                atualizarAlunos(&alunos);
                break;
            case 4:
                buscarAlunos(&alunos);
                break;
            case 5:
                listarAlunos(&alunos);
                break;
            default:
                printf("opcao invalida");
        }
    } while (opcao != 0);

    return 0;
}
```

main.c

```
do {  
  
...  
  
} while (opcao != 0);
```

```
#include <stdio.h>  
#include "alunos.h"  
#include "input.h"  
  
int main() {  
    int opcao;  
    Alunos alunos = {contador = 0};  
  
    do {  
        printf("alunos.....");  
        printf("\n - inserir");  
        printf("\n - remover");  
        printf("\n - consultar");  
        printf("\n - listar");  
        printf("\n - sair");  
        printf("op.....");  
        printf("alunos: %d/%d, alunos: contador, max_alunos);  
  
        printf("opcao: ");  
        scanf("%d", &opcao);  
  
        switch (opcao) {  
            case 0:  
                break;  
            case 1:  
                inserirAlunos(&alunos);  
                break;  
            case 2:  
                removerAlunos(&alunos);  
                break;  
            case 3:  
                consultarAlunos(&alunos);  
                break;  
            case 4:  
                listarAlunos(&alunos);  
                break;  
            case 5:  
                sairAlunos(&alunos);  
                break;  
            default:  
                printf("opcao invalida");  
        }  
    } while (opcao != 0);  
  
    return 0;  
}
```

main.c

```
printf("\nAlunos-----");
printf("\n1 - Inserir");
printf("\n2 - Procurar");
printf("\n3 - Atualizar");
printf("\n4 - Remover");
printf("\n5 - Listar");
printf("\n0 - Sair");
printf("\n-----");
printf("\nAlunos: %d/%d", alunos.contador, MAX_ALUNOS);
```

```
#include <stdio.h>
#include "alunos.h"
#include "input.h"

int main() {
    int opcao;
    Alunos alunos = {contador = 0};

    printf("\nAlunos-----");
    printf("\n1 - Inserir");
    printf("\n2 - Procurar");
    printf("\n3 - Atualizar");
    printf("\n4 - Remover");
    printf("\n5 - Listar");
    printf("\n0 - Sair");
    printf("\n-----");
    printf("\nAlunos: %d/%d", alunos.contador, MAX_ALUNOS);

    printf("\nOpcao: ");
    scanf("%d", &opcao);

    switch (opcao) {
        case 0:
            break;
        case 1:
            inserirAlunos(&alunos);
            break;
        case 2:
            procurarAlunos(&alunos);
            break;
        case 3:
            atualizarAlunos(&alunos);
            break;
        case 4:
            removerAlunos(&alunos);
            break;
        case 5:
            listarAlunos(&alunos);
            break;
        default:
            printf("\nOpcao invalida");
    }

    } while (opcao != 0);

    return 0;
}
```

main.c

```
printf("\nOpção: ");  
scanf("%d", &opcao);
```

```
iniciate_vetores.h  
#include "alunos.h"  
#include "input.h"  
  
int main() {  
    int opcao;  
    Alunos alunos = {contador = 0};  
  
    do {  
        printf("Menu: .....");  
        printf("\n - Inserir");  
        printf("\n - Remover");  
        printf("\n - Visualizar");  
        printf("\n - Buscar");  
        printf("\n - Lista");  
        printf("\n - Sair");  
        printf("Opção: .....");  
        printf("Menu: %d", opcao, alunos.contador, MAX_ALUNOS);  
  
        printf("Opção: ");  
        scanf("%d", &opcao);  
  
        switch (opcao) {  
            case 0:  
                break;  
            case 1:  
                inserirAlunos(&alunos);  
                break;  
            case 2:  
                removerAlunos(&alunos);  
                break;  
            case 3:  
                visualizarAlunos(&alunos);  
                break;  
            case 4:  
                buscarAlunos(&alunos);  
                break;  
            case 5:  
                listarAlunos(&alunos);  
                break;  
            default:  
                printf("Opção inválida");  
        }  
    } while (opcao != 0);  
  
    return 0;  
}
```

main.c

```
switch (opcao) {
    case 0:
        break;

    case 1:
        inserirAlunos(&alunos);
        break;

    case 2:
        procurarAlunos(alunos);
        break;

    case 3:
        atualizarAlunos(&alunos);
        break;

    case 4:
        removerAlunos(&alunos);
        break;

    case 5:
        listarAlunos(alunos);
        break;

    default:
        printf("\nOpção invalida!");
}
```

```
#include <stdio.h>
#include "alunos.h"
#include "input.h"

int main() {
    int opcao;
    Alunos alunos = {0, 0};

    do {
        printf("alunos.....");
        printf("\n - inserir");
        printf("\n - procurar");
        printf("\n - atualizar");
        printf("\n - remover");
        printf("\n - listar");
        printf("\n - sair");
        printf("\n.....");
        printf("alunos: %d/%d, alunos: contador, max_alunos);
        printf("\nOpção: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 0:
                break;
            case 1:
                inserirAlunos(&alunos);
                break;
            case 2:
                procurarAlunos(alunos);
                break;
            case 3:
                atualizarAlunos(&alunos);
                break;
            case 4:
                removerAlunos(&alunos);
                break;
            case 5:
                listarAlunos(alunos);
                break;
            default:
                printf("\nOpção invalida!");
        }

    } while (opcao != 0);

    return 0;
}
```

Leitura recomendada

- (Capítulo 11) Damas, L. Linguagem C; FCA – Editora de Informática, Lda, 1999; ISBN 9789727221561.

