Fundamentos da Programação

Gestão de Registos

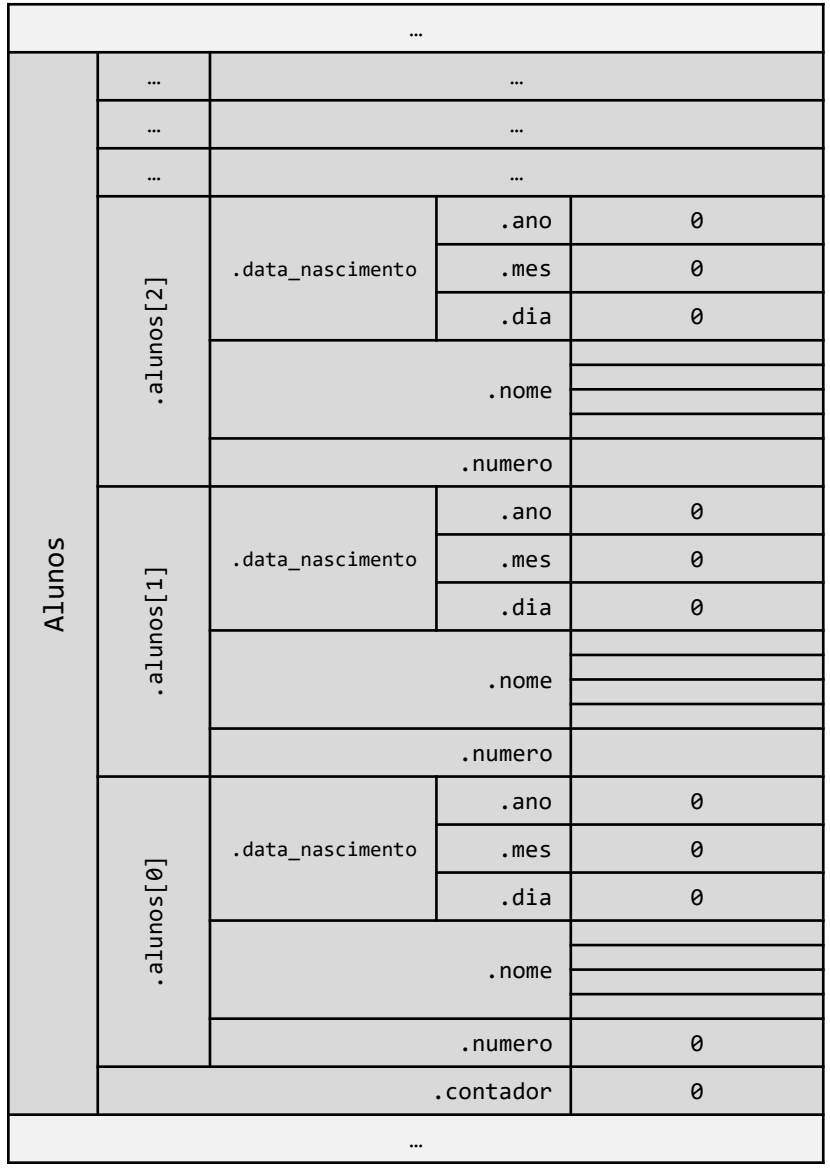
P.PORTO
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Conteúdo

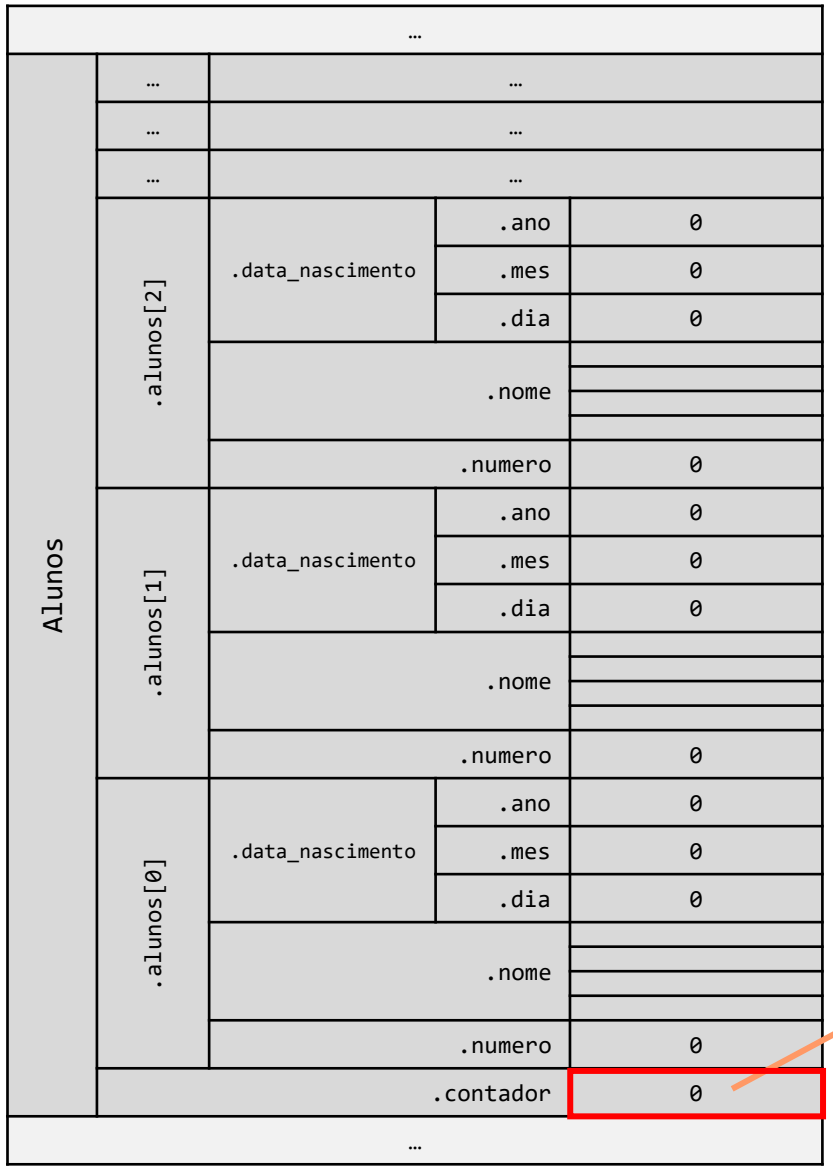
- Gestão de Registos
- Operações *Create, Read, Update, and Delete* (CRUD)

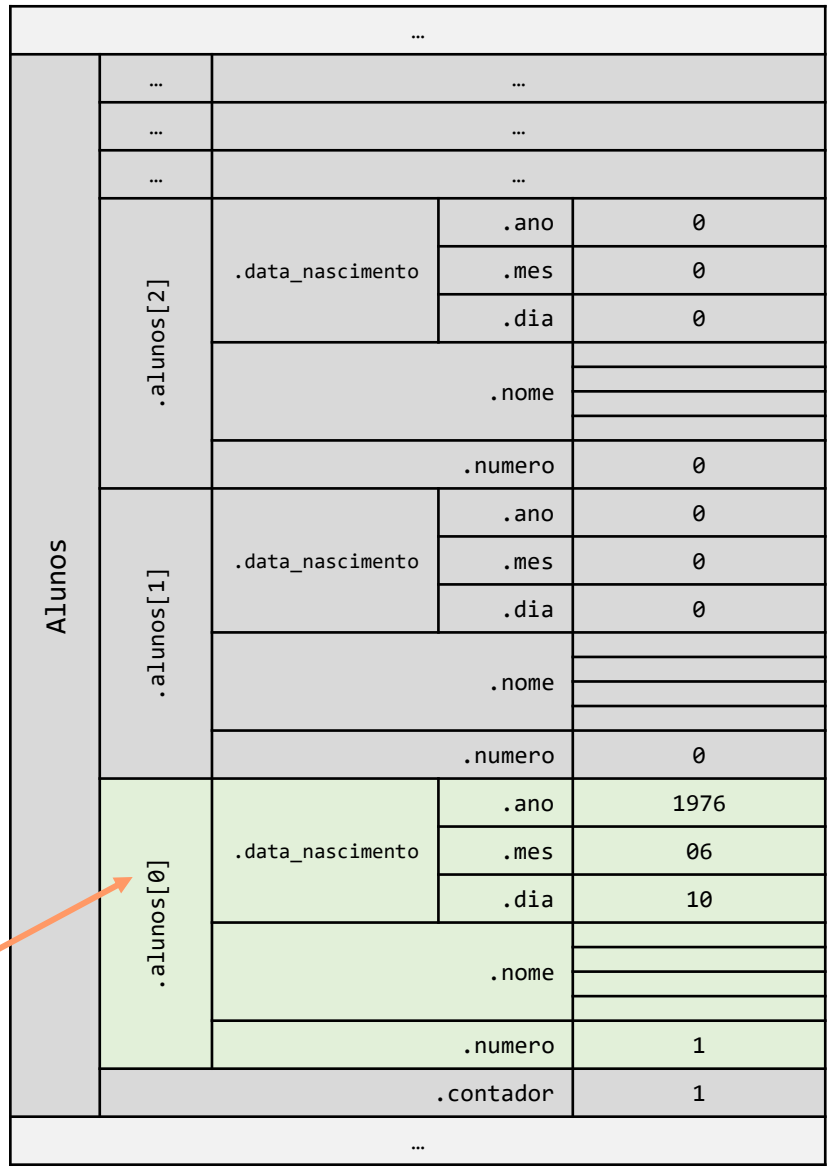
Gestão de alunos

```
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;
```

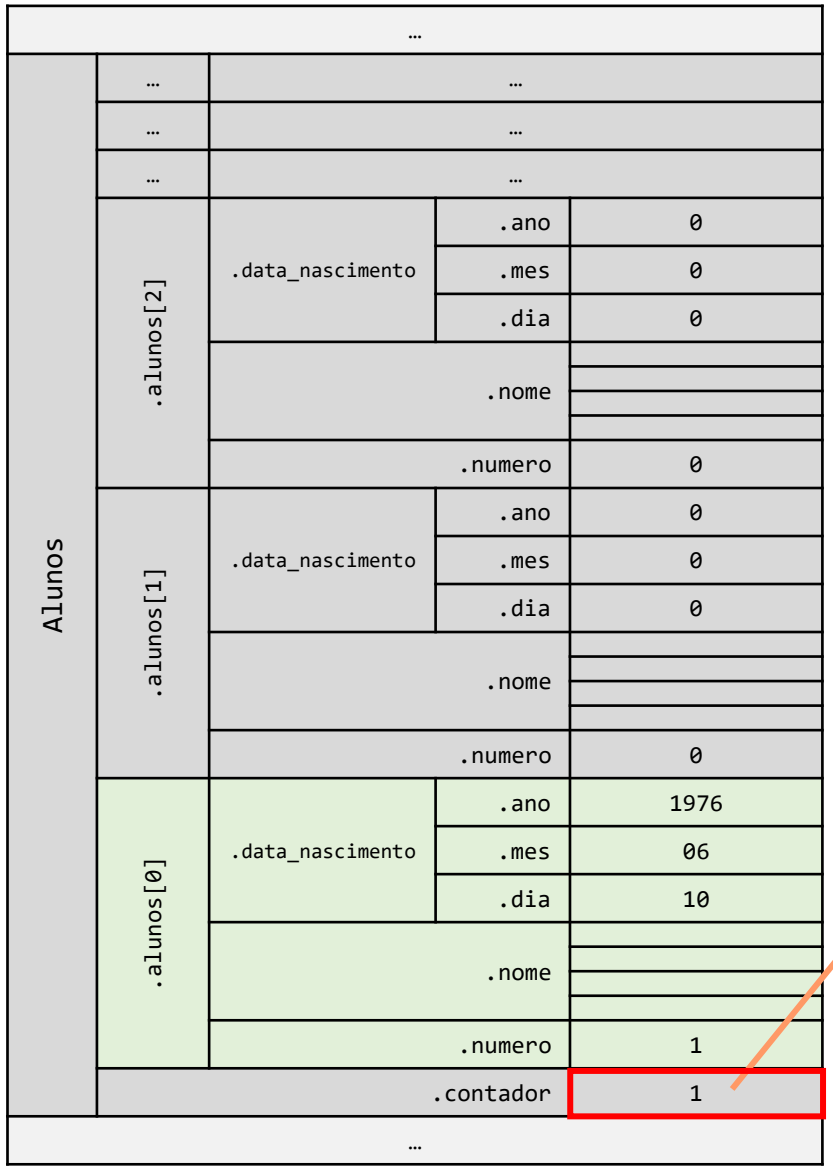


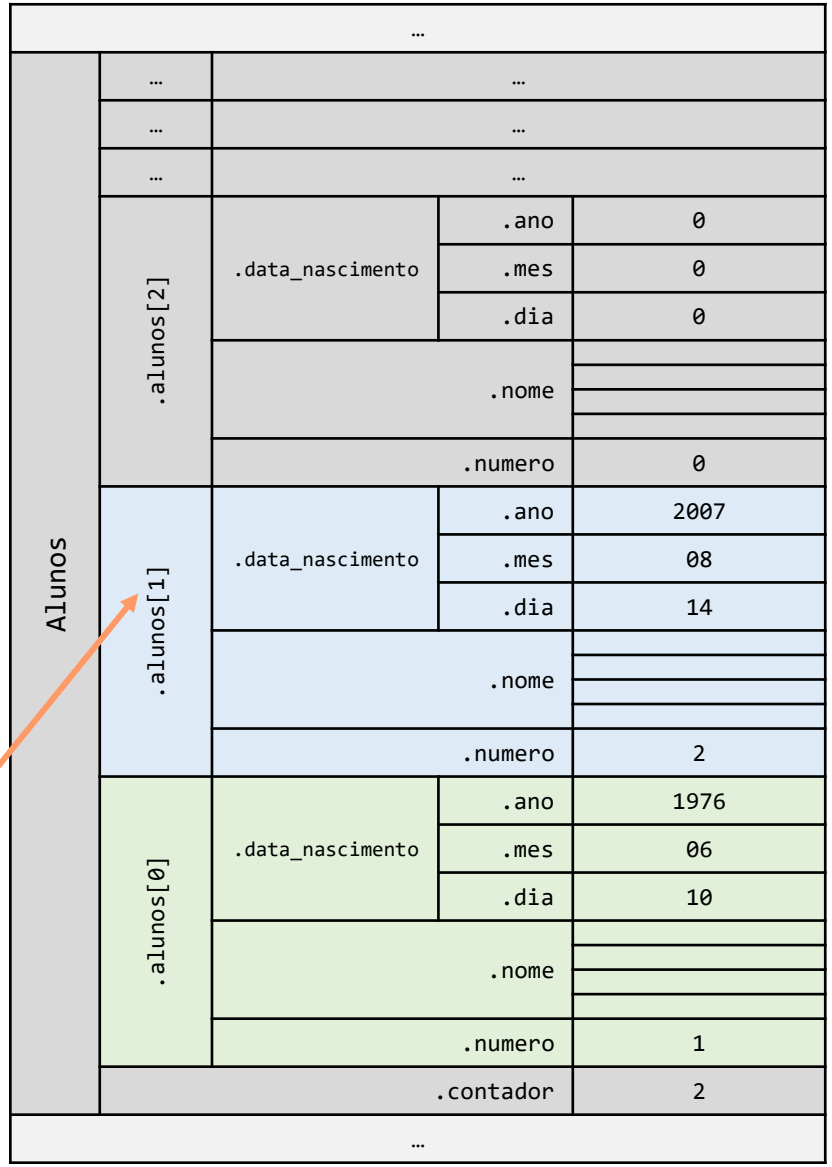
Inserir


...
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
...

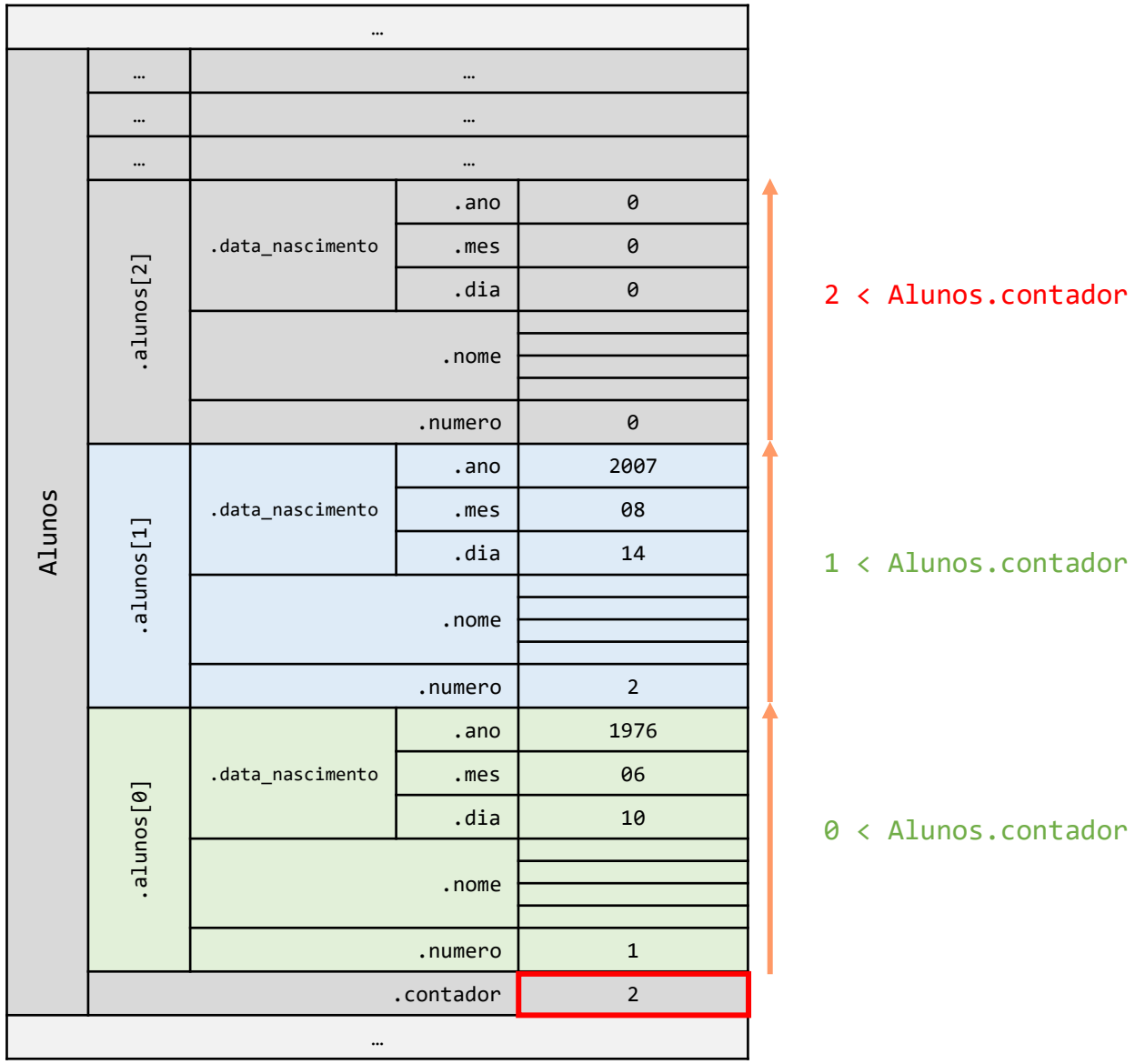

...
Alunos
...
...
...
.alunos[2]
.data_nascimento
.ano 0
.mes 0
.dia 0
.nome
.numero 0
.alunos[1]
.data_nascimento
.ano 0
.mes 0
.dia 0
.nome
.numero 0
.alunos[0]
.data_nascimento
.ano 1976
.mes 06
.dia 10
.nome
.numero 1
.contador 1
...

Inserir

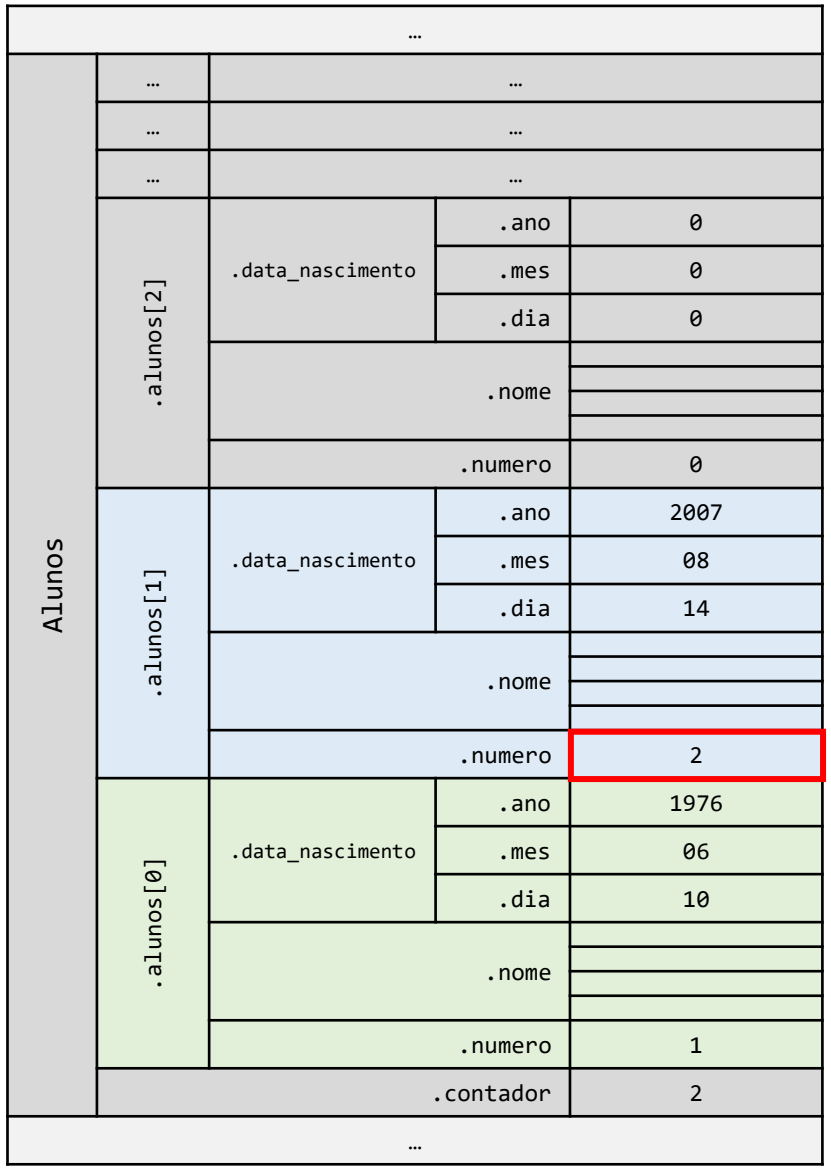

...
Alunos
...
...
...
.alunos[2]
.data_nascimento
.ano 0
.mes 0
.dia 0
.nome
.numero 0
.alunos[1]
.data_nascimento
.ano 0
.mes 0
.dia 0
.nome
.numero 0
.alunos[0]
.data_nascimento
.ano 1976
.mes 06
.dia 10
.nome
.numero 1
.contador 1
...

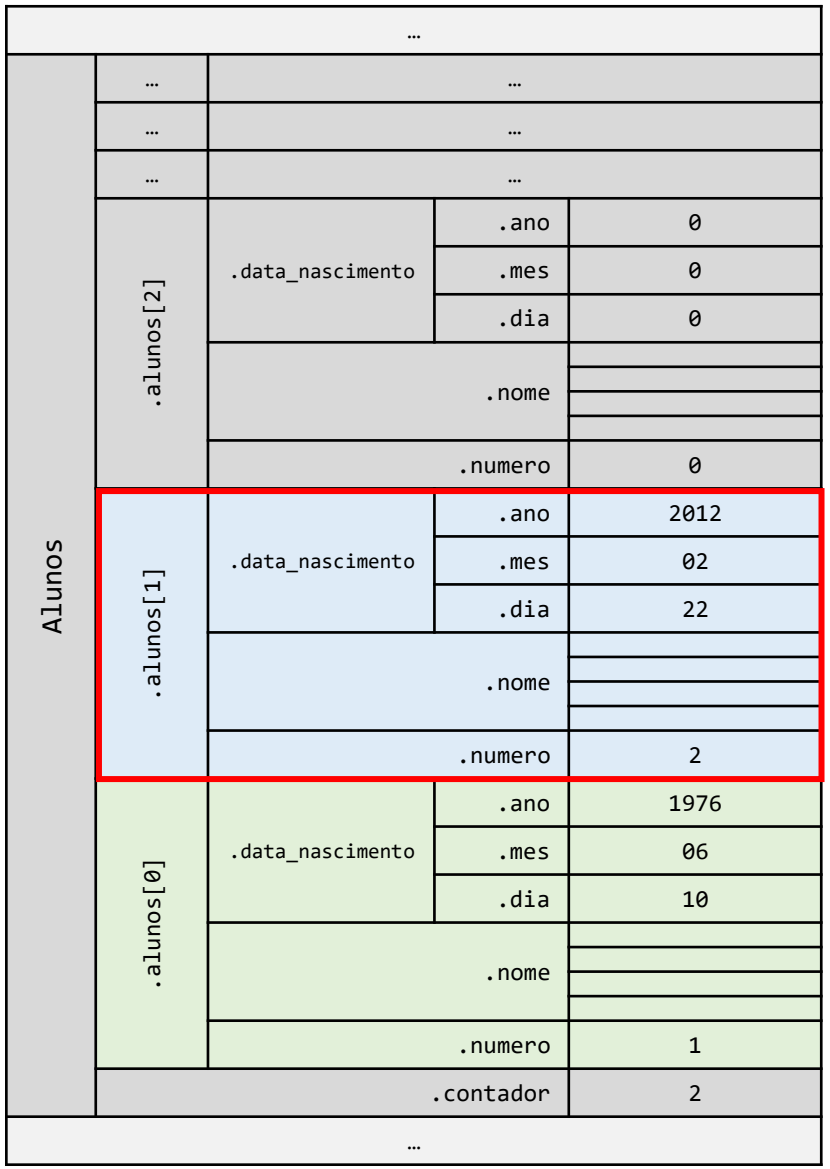

...
Alunos
...
...
...
.alunos[2]
.data_nascimento
.ano 0
.mes 0
.dia 0
.nome
.numero 0
.alunos[1]
.data_nascimento
.ano 2007
.mes 08
.dia 14
.nome
.numero 2
.alunos[0]
.data_nascimento
.ano 1976
.mes 06
.dia 10
.nome
.numero 1
.contador 2
...

Consultar

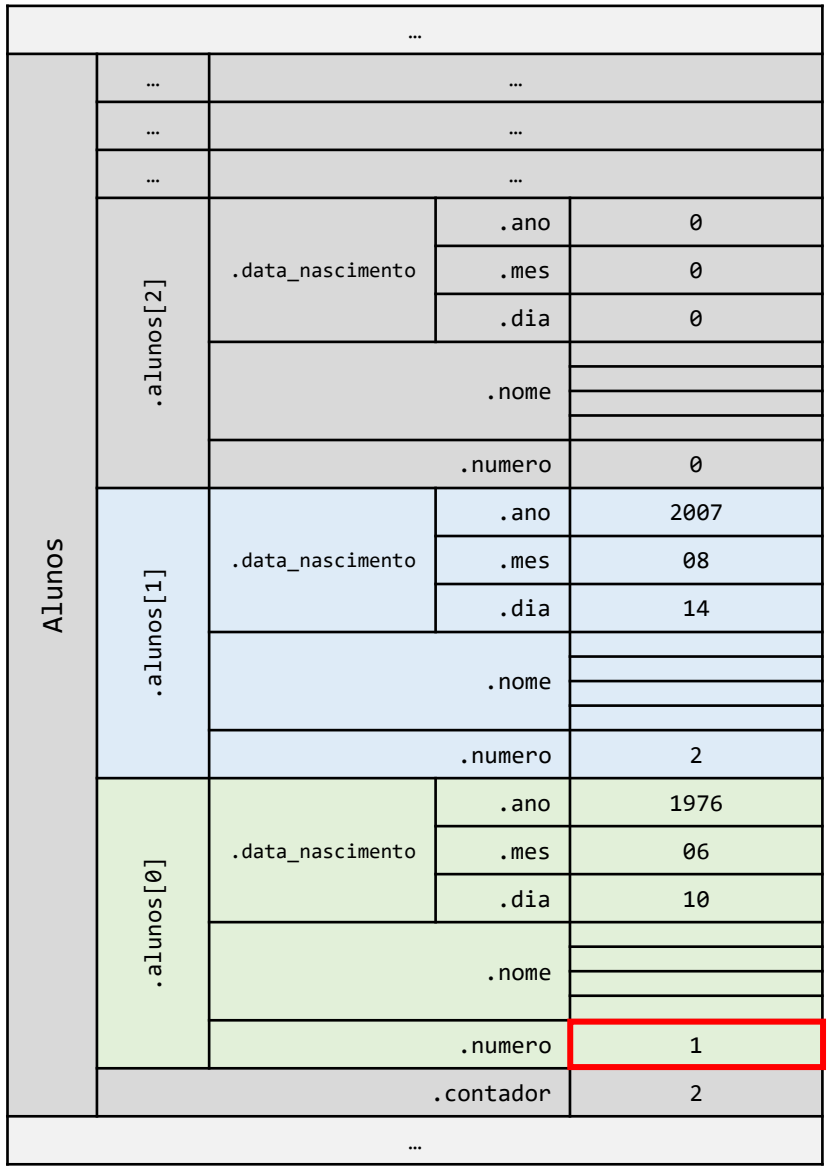

…
Alunos
…
…
…
.alunos[2]
.data_nascimento
.ano
0
.mes
0
.dia
0
.nome
.numero
0
.alunos[1]
.data_nascimento
.ano
2007
.mes
08
.dia
14
.nome
.numero
2
.alunos[0]
.data_nascimento
.ano
1976
.mes
06
.dia
10
.nome
.numero
1
.contador
2
…
2 < Alunos.contador
1 < Alunos.contador
0 < Alunos.contador

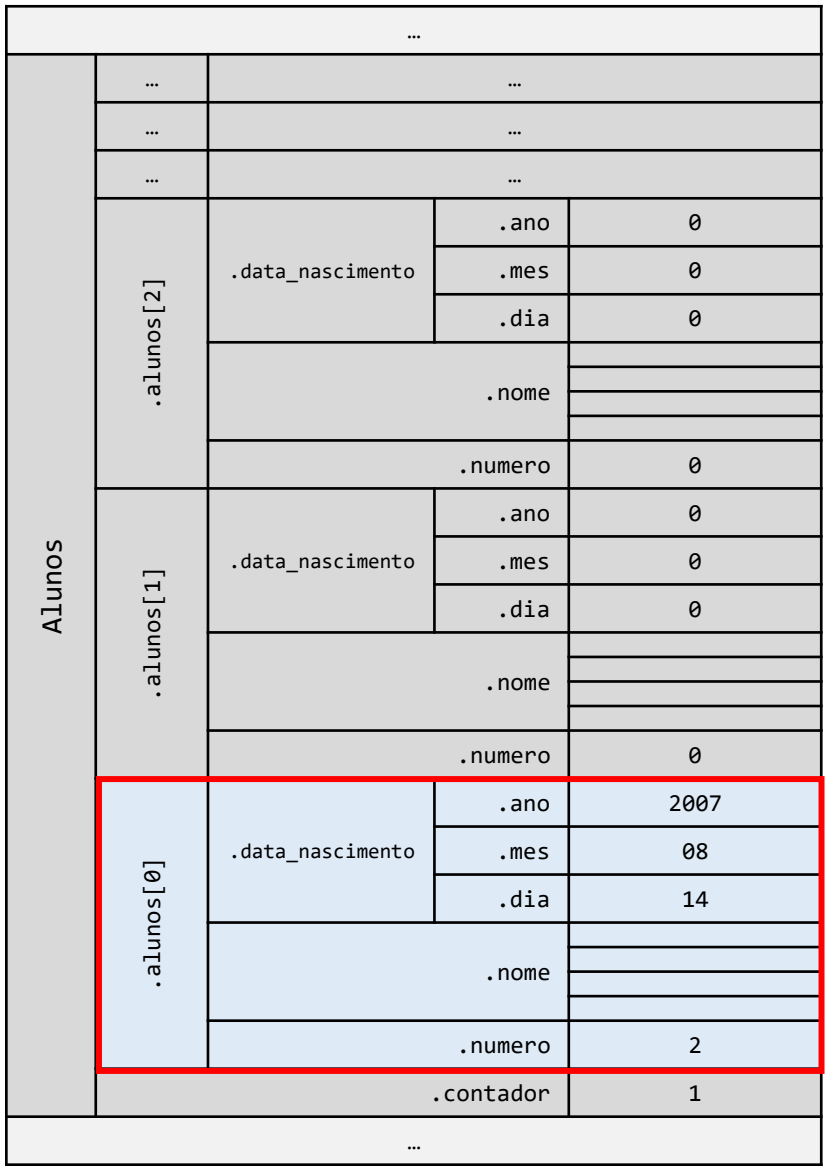
Atualizar


...
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

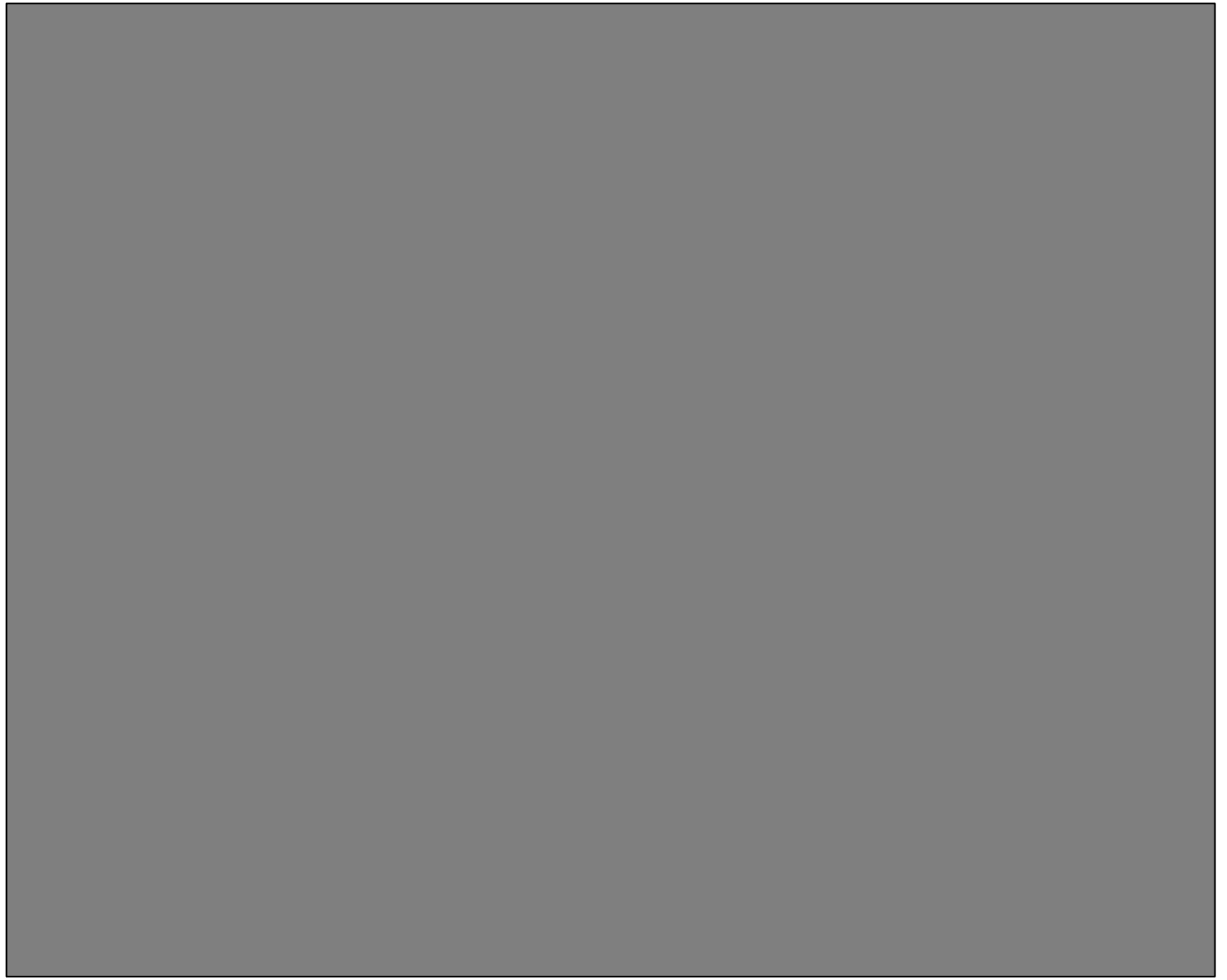
main.c

alunos.c

input.h

alunos.h

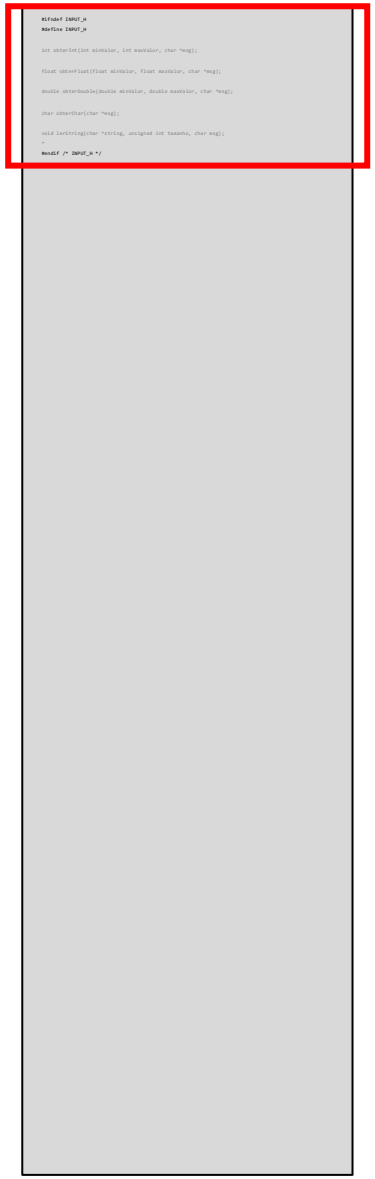
input.h



```
#ifndef INPUT_H
#define 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);

#endif /* INPUT_H */
```

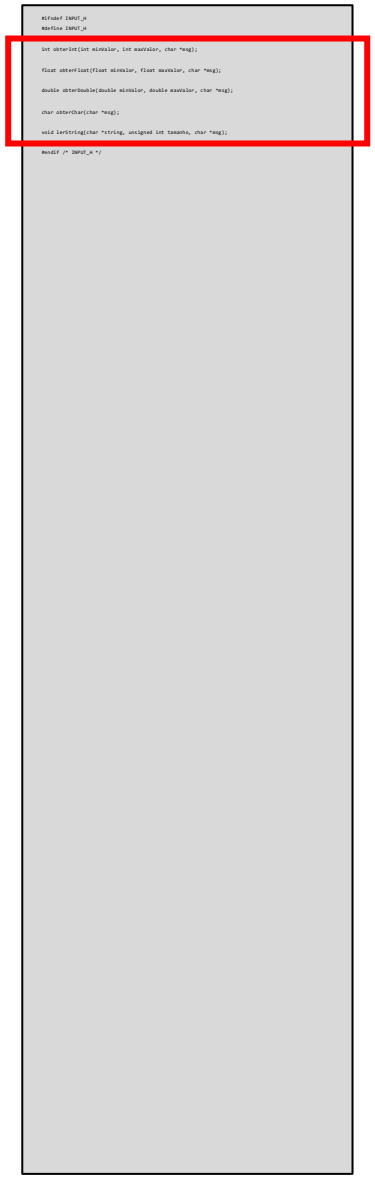
input.h

```
#ifndef INPUT_H
#define INPUT_H


…


#endif /* INPUT_H */
```

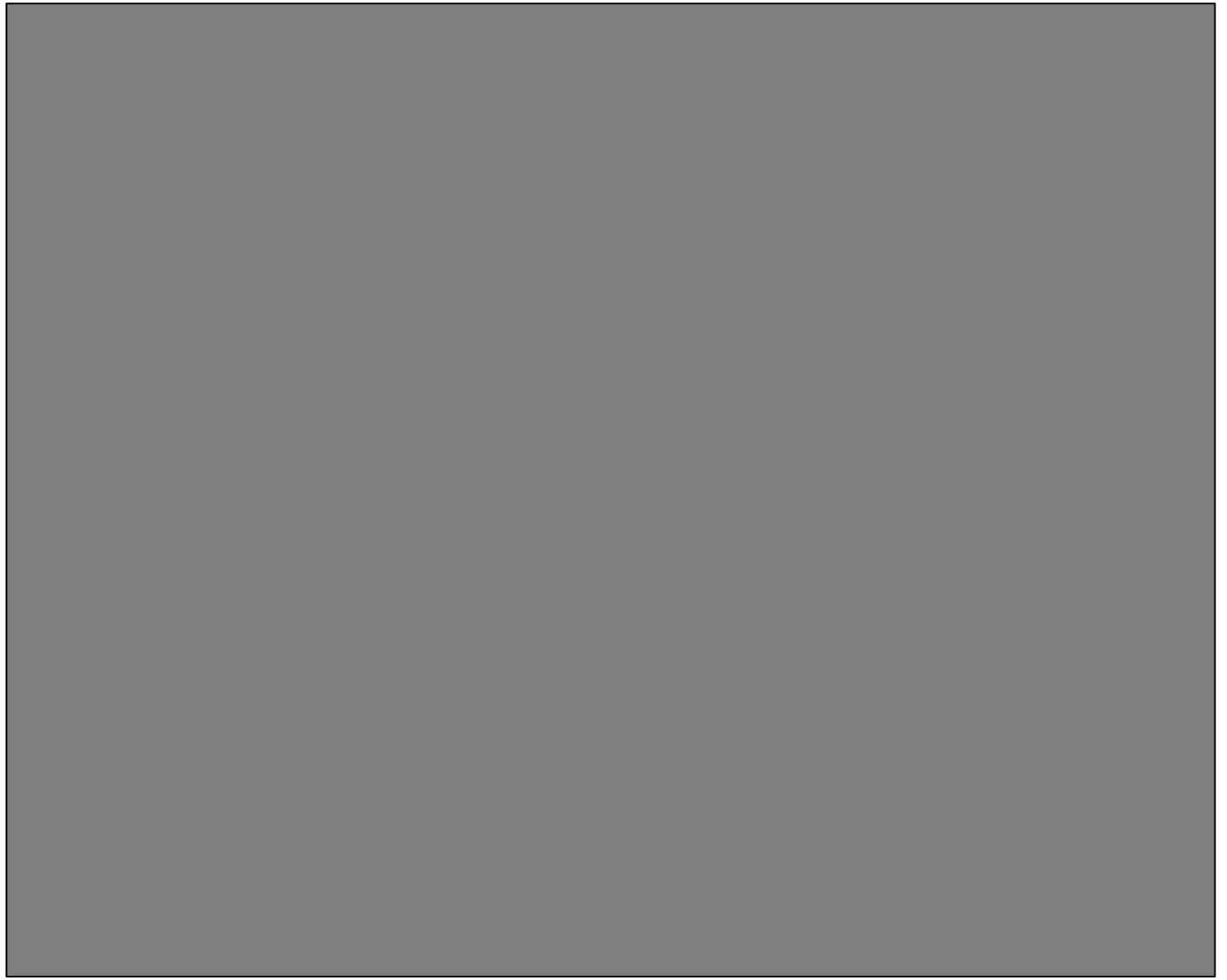


input.h

```
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);
```



input.c



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

#define VALOR_INVALIDO "O valor inserido é inválido."

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

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

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

input.c

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

input.c

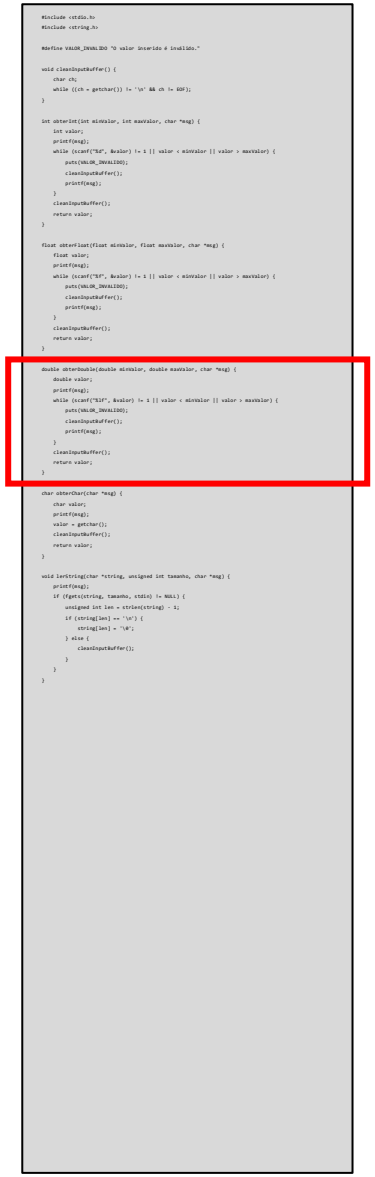
```c
int obterInt(int minValor, int maxValor, char *msg) {
    int valor;
    printf(msg);
    while (scanf("%d", &valor) != 1 || valor < minValor || valor > maxValor) {
        puts(VALOR_INVALIDO);
        cleanInputBuffer();
        printf(msg);
    }
    cleanInputBuffer();
    return valor;
}
```

input.c

```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;
}
```

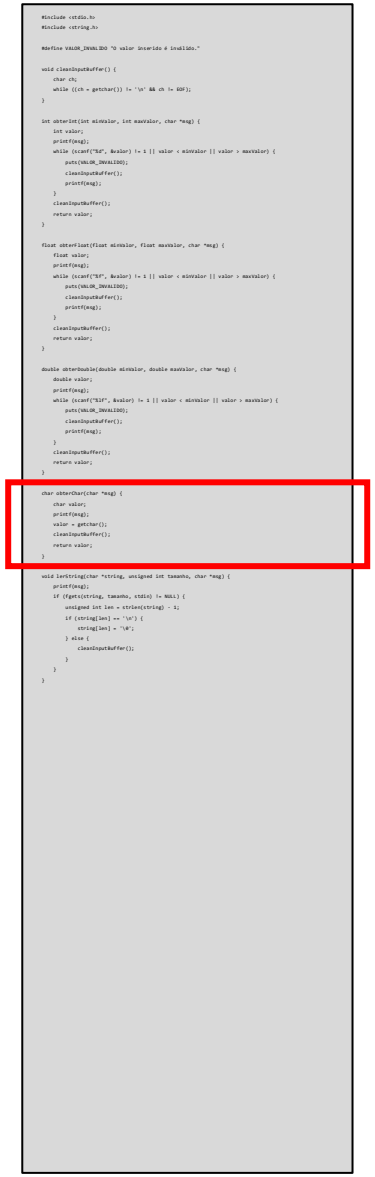
input.c

```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;
}
```



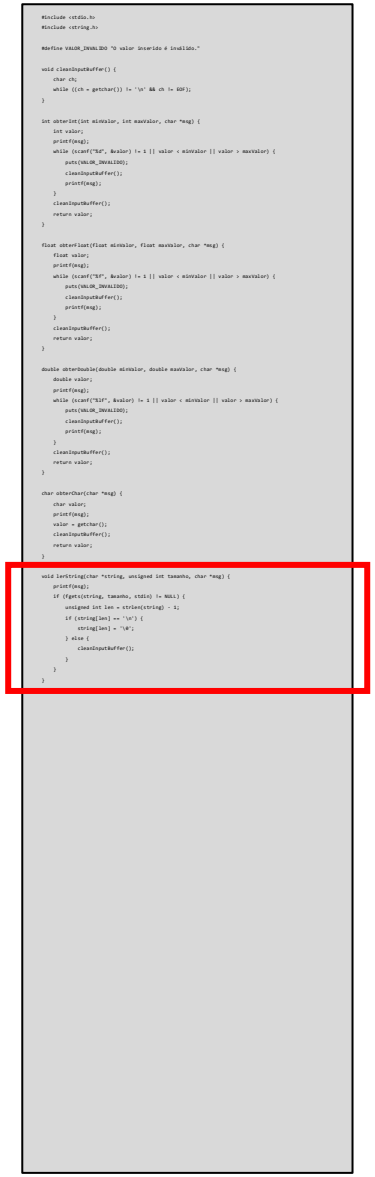
input.c

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

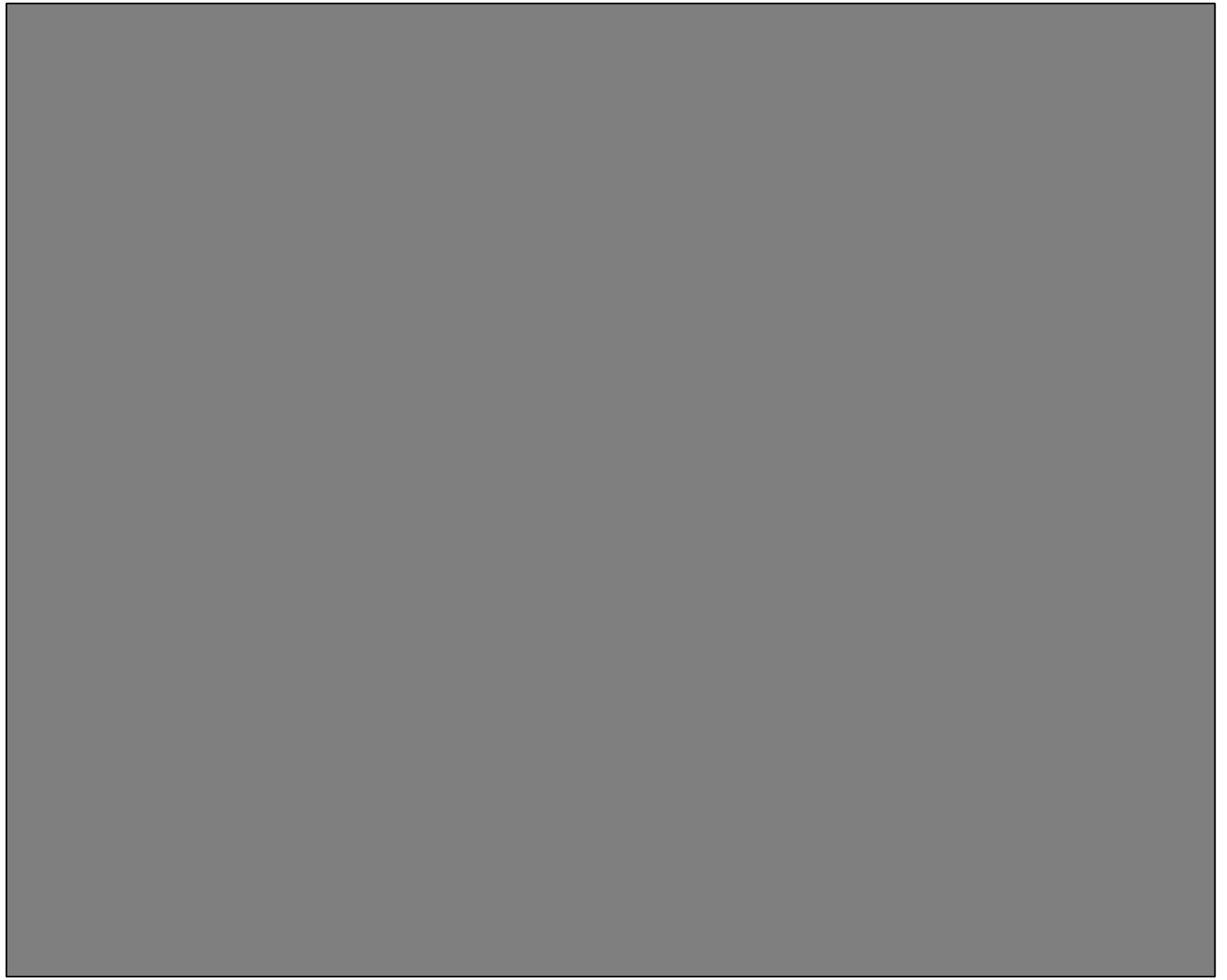


input.c

```
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();
        }
    }
}
```



alunos.h



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

#endif /* ALUNOS_H */
```

alunos.h

```
#ifndef ALUNOS_H
#define ALUNOS_H


…


#endif /* ALUNOS_H */
```

alunos.h

```
#define MAX_ALUNOS  30
```

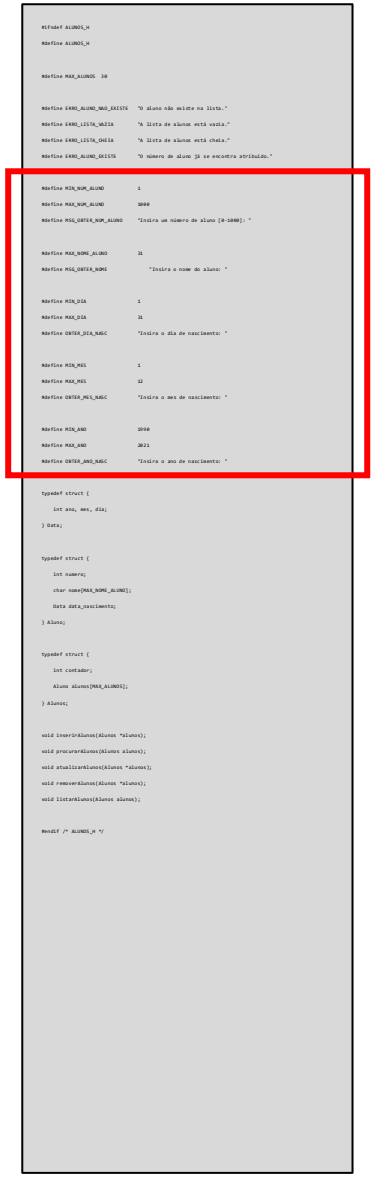
```
#ifndef ALUNOS_H
#define ALUNOS_H

#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 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 mes 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 inserirAlunos(Alunos *alunos);
void procurarAlunos(Alunos alunos);
void atualizarAlunos(Alunos *alunos);
void removerAlunos(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."
```

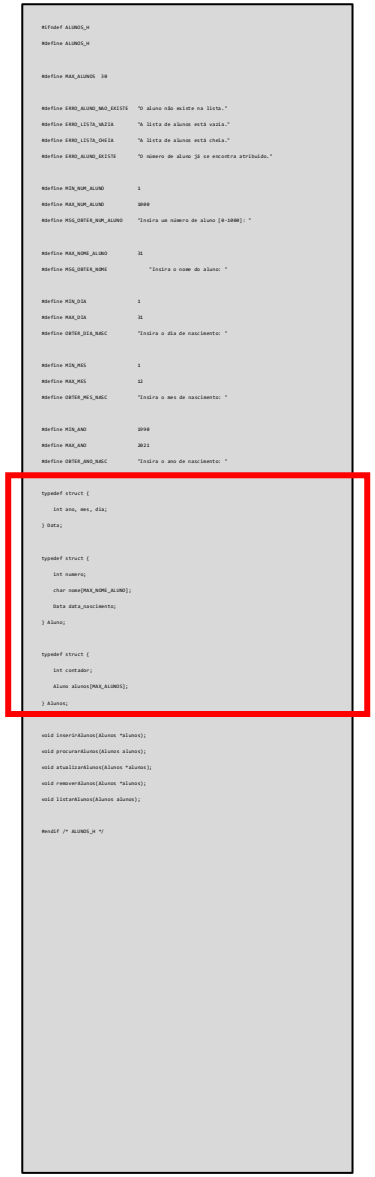
alunos.h

```
#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: "
```



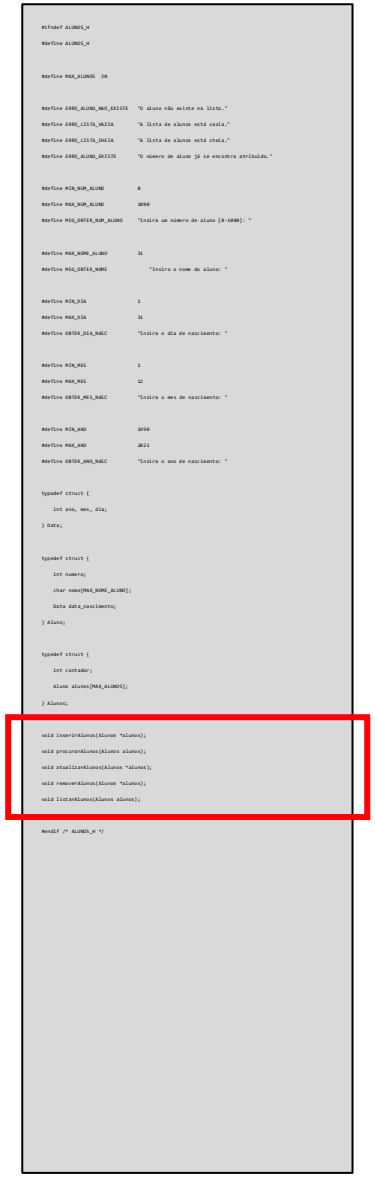
alunos.h

```
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;
```

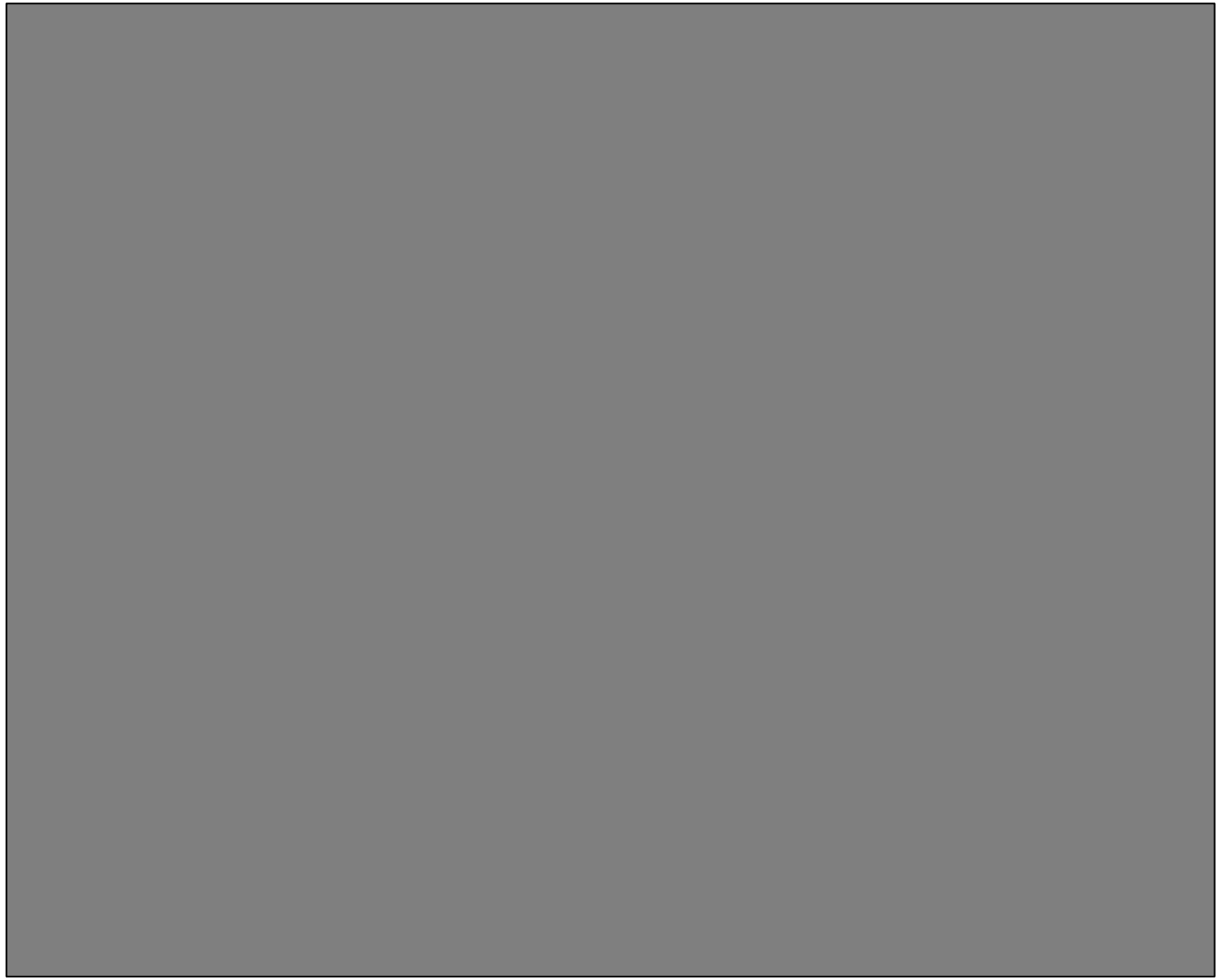


alunos.h

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



alunos.c



```
/*****
#include <stdlib.h>
#include <string.h>

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

void imprimirAluno(Aluno aluno) {
    printf("\tNúm %-30s %d-%d-%d", aluno.numero, aluno.nome, aluno.data_nascimento.dia,
        aluno.data_nascimento.mes, aluno.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(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_OBTER_NUM_ALUNO);

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

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

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

        alunos->alunos[alunos->contador].data_nascimento.dia = obterInt(MIN_DIA, MAX_DIA,
                                                                        OBTER_DIA_NASC);
        alunos->alunos[alunos->contador].data_nascimento.mes = obterInt(MIN_MES, MAX_MES,
                                                                        OBTER_MES_NASC);
        alunos->alunos[alunos->contador].data_nascimento.ano = obterInt(MIN_ANO, MAX_ANO,
                                                                        OBTER_ANO_NASC);

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

void atualizarAluno(Aluno *aluno) {
    lerString((*aluno).nome, MAX_NOME_ALUNO, MSG_OBTER_NOME);
    (*aluno).data_nascimento.dia = obterInt(MIN_DIA, MAX_DIA, OBTER_DIA_NASC);
    (*aluno).data_nascimento.mes = obterInt(MIN_MES, MAX_MES, OBTER_MES_NASC);
    (*aluno).data_nascimento.ano = obterInt(MIN_ANO, MAX_ANO, OBTER_ANO_NASC);
}

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 = procurarAluno(alunos, obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO,
MSG_OBTER_NUM_ALUNO));

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

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

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

void removerAluno(Alunos *alunos) {
    int i, numero = procurarAluno(*alunos, obterInt(MIN_NUM_ALUNO, MAX_NUM_ALUNO,
MSG_OBTER_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) {
        int i;
        for (i = 0; i < alunos.contador; i++) {
            imprimirAluno(alunos.alunos[i]);
        }
    } else {
        puts(ERRO_LISTA_VAZIA);
    }
}
```

alunos.c

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

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

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);
}
```

alunos.c

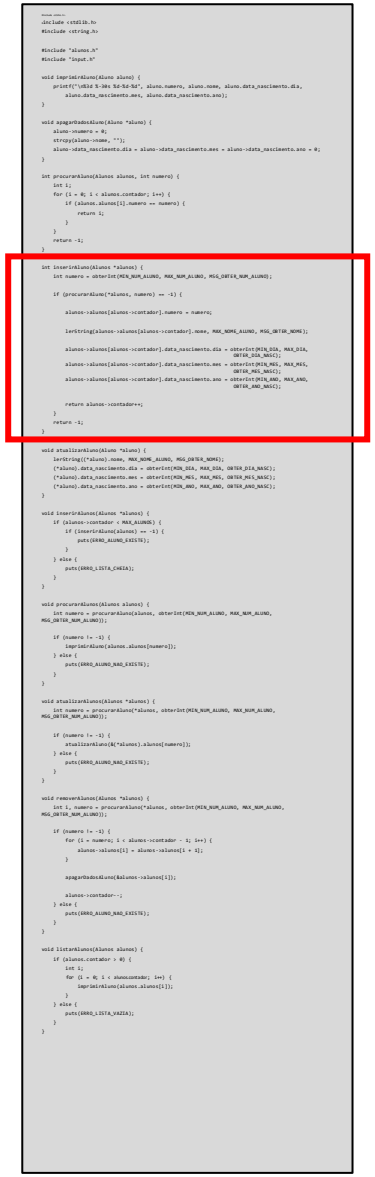
```
void apagarDadosAluno(Aluno *aluno) {
    aluno->numero = 0;
    strcpy(aluno->nome, "");
    aluno->data_nascimento.dia = aluno->data_nascimento.mes = aluno->data_nascimento.ano = 0;
}
```

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;
}
```

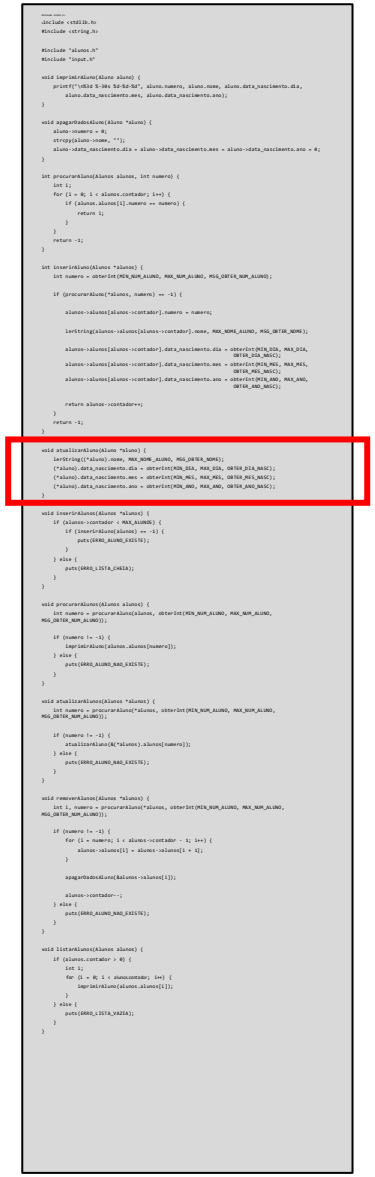
alunos.c

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

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

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

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

        alunos->alunos[alunos->contador].data_nascimento.dia =
             obterInt(MIN_DIA, MAX_DIA, OBTER_DIA_NASC);
        alunos->alunos[alunos->contador].data_nascimento.mes =
             obterInt(MIN_MES, MAX_MES, OBTER_MES_NASC);
        alunos->alunos[alunos->contador].data_nascimento.ano =
             obterInt(MIN_ANO, MAX_ANO, OBTER_ANO_NASC);

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



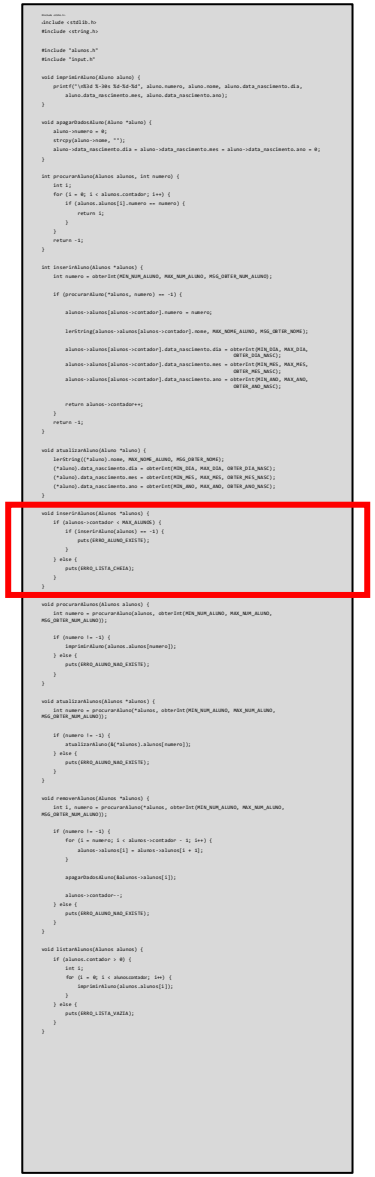
alunos.c

```
void atualizarAluno(Aluno *aluno) {
    lerString((*aluno).nome, MAX_NOME_ALUNO, MSG_OBTER_NOME);
    (*aluno).data_nascimento.dia = obterInt(MIN_DIA, MAX_DIA, OBTER_DIA_NASC);
    (*aluno).data_nascimento.mes = obterInt(MIN_MES, MAX_MES, OBTER_MES_NASC);
    (*aluno).data_nascimento.ano = obterInt(MIN_ANO, MAX_ANO, OBTER_ANO_NASC);
}
```



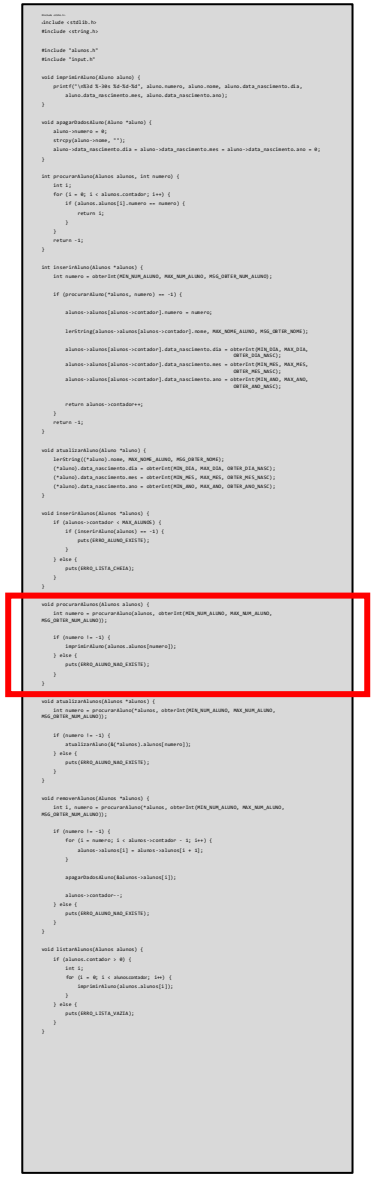
alunos.c

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



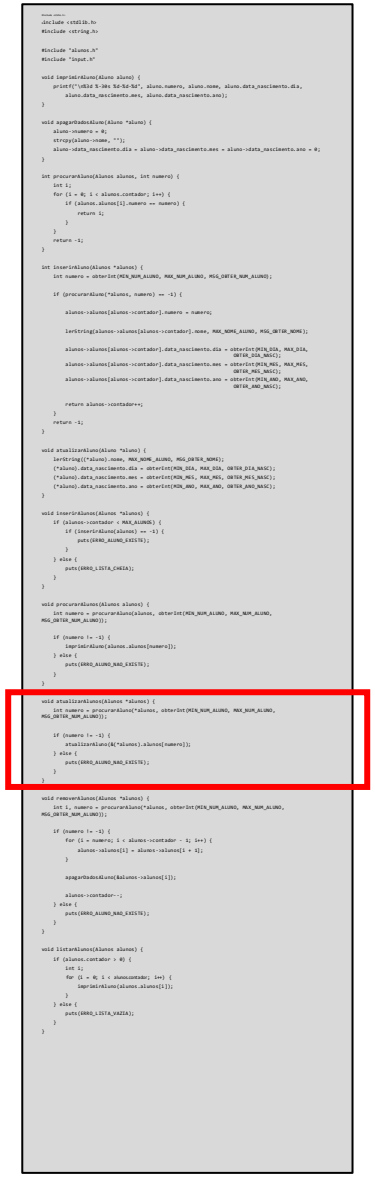
alunos.c

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


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



alunos.c

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


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



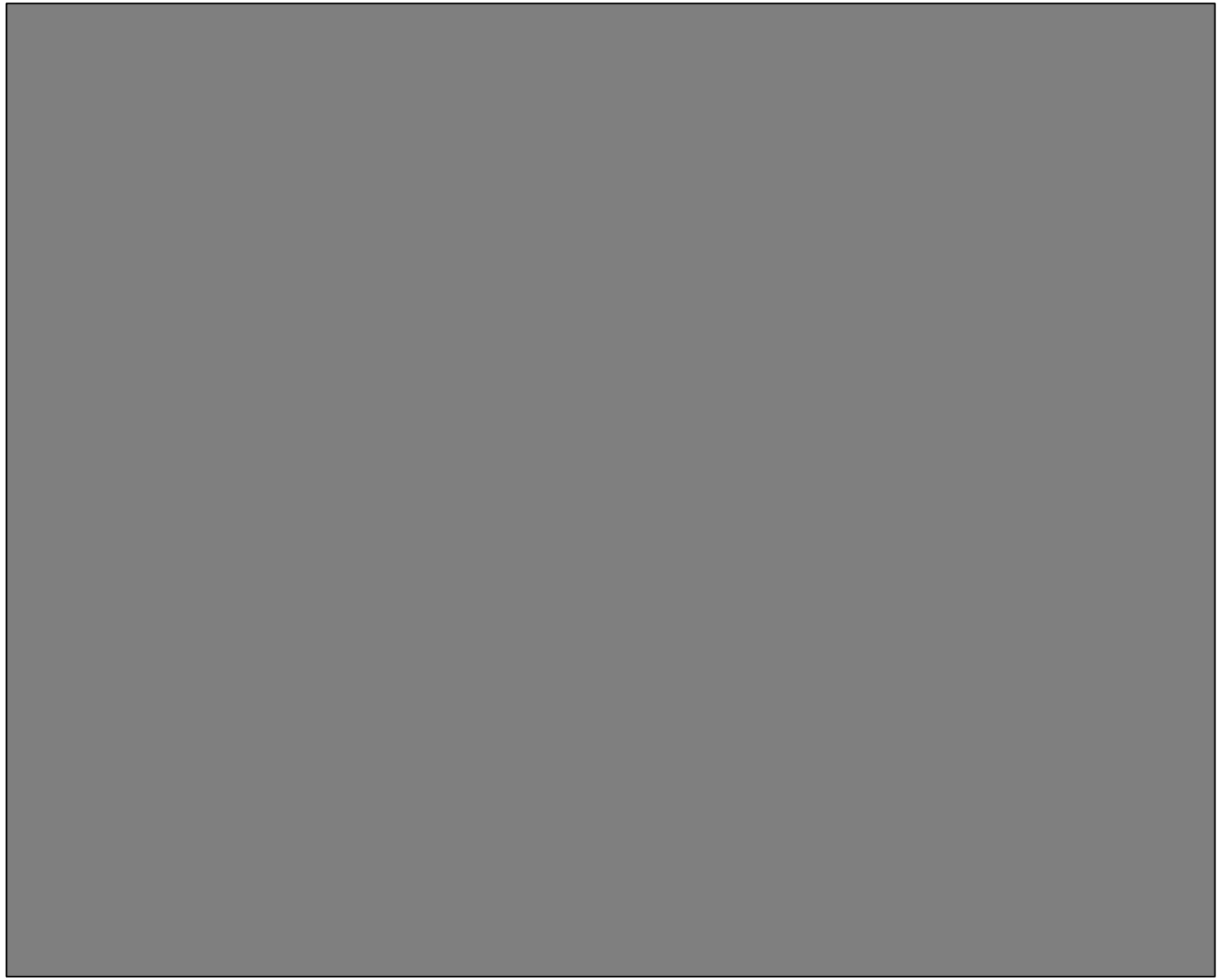
alunos.c

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

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);
    }
}
```

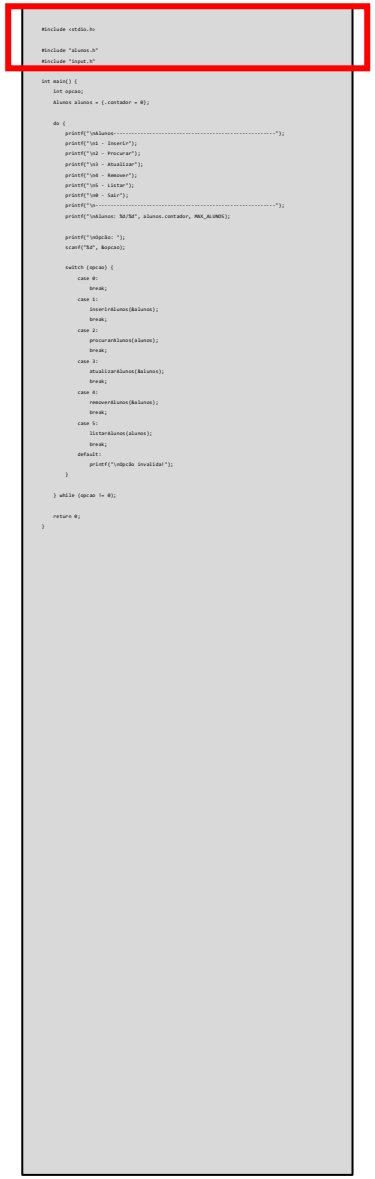
main.c



```c
#include <stdio.h>

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

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

    do {
        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("\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;
}
```

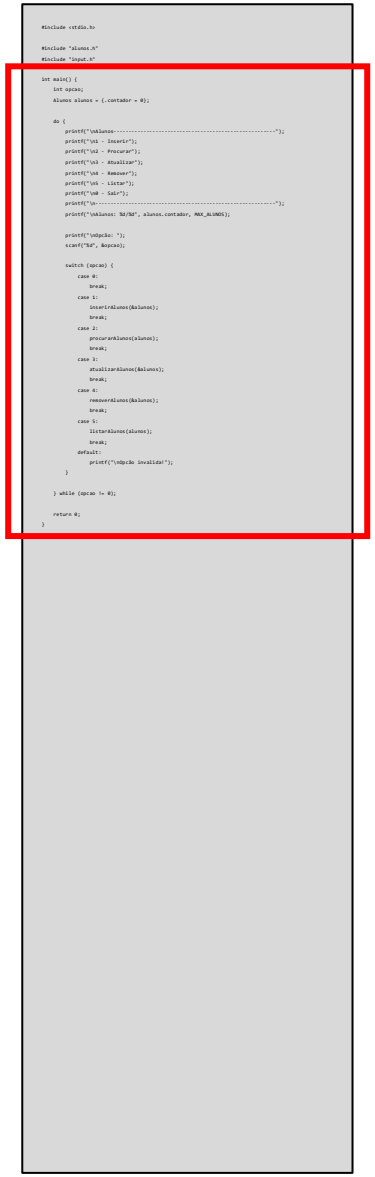
main.c

```c
#include <stdio.h>


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



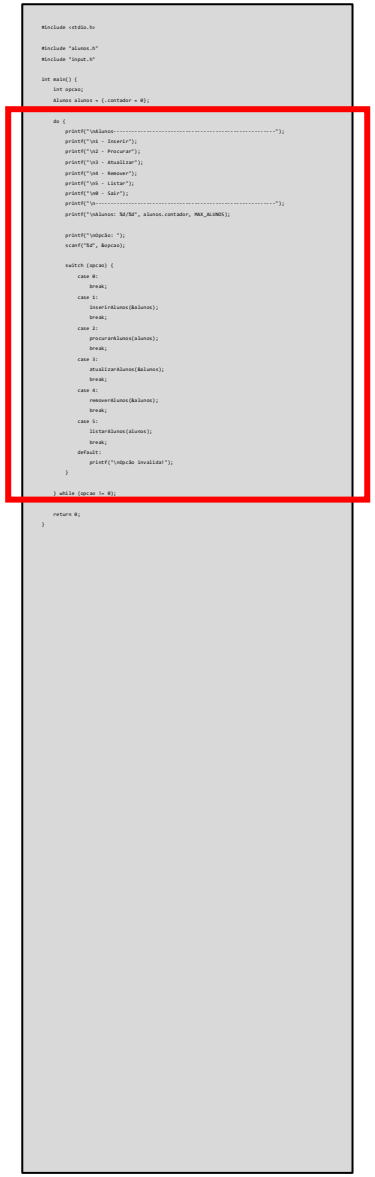
main.c

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


    …


    return 0;
}
```



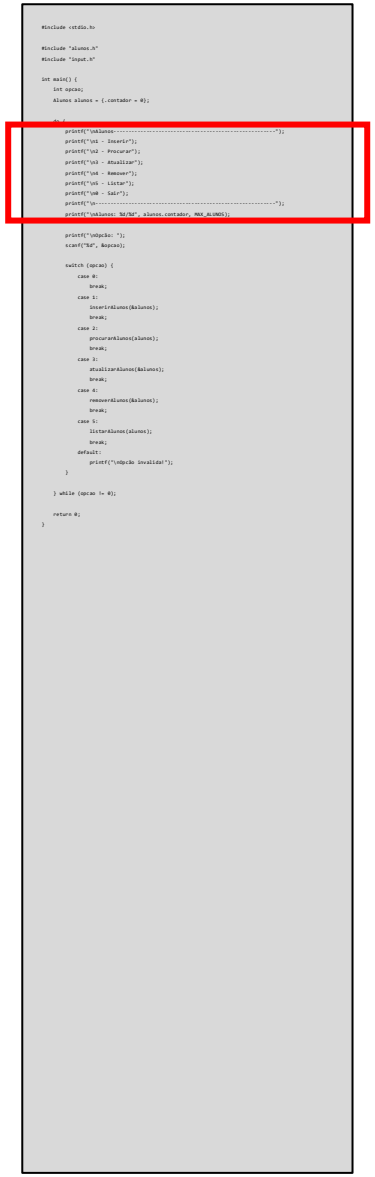
main.c

```
do {


…


} while (opcao != 0);
```



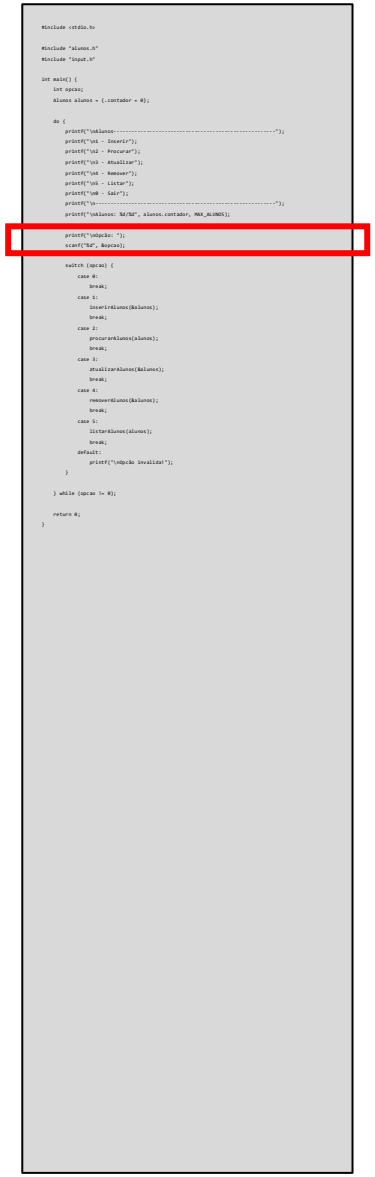
main.c

```
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);
```



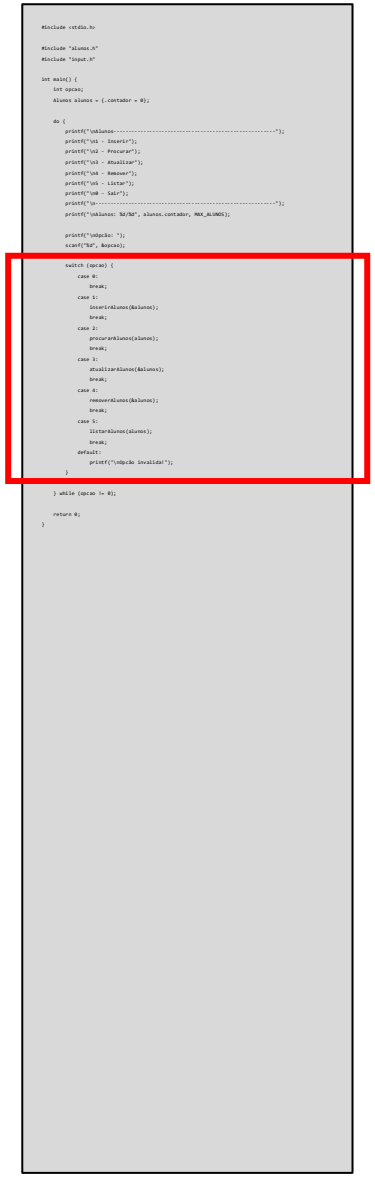
main.c

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



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("\nOpcão invalida!");
}
```



Leitura recomendada

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