Um Tipo Abstrato de Dados é um conjunto de atributos e operações que tem como função principal abstrair as características de um elemento em um tipo.

Os TADs facilitam a forma de lidar com entidades que possuem vários atributos únicos e operações que devem seguir regras específicas, possibilitando o encapsulamento da entidade.

roteiro-03/ex01-02.c

```
#include <stdio.h>
   #include "cubo.h"
 2
 3
   double lado (Cubo* c) {
 5
        return c->lado;
 6
   }
 7
   double area(Cubo* c){
        double l = lado(c);
 9
        return l * l;
10
11
   }
12
13
   double volume(Cubo* c){
14
        double a = area(c);
15
        double l = lado(c);
16
        return a * l;
17
   }
18
19
   int main() {
20
       double l;
21
22
        printf("Lado do cubo: ");
        scanf("%lf", &l);
23
24
        Cubo c;
25
        c.lado = l;
26
27
        printf("Lado = %.2lf\n", lado(&c));
28
        printf("Area = %.2lf\n", area(&c));
29
        printf("Volume = %.2lf\n", volume(&c));
30
31
        return 0;
32 }
```

roteiro-03/cubo.h

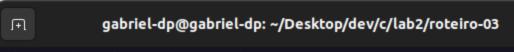
```
#ifndef CUBO_H

#define CUBO_H

typedef struct {
    double lado;
} Cubo;

double lado(Cubo* c);
double area(Cubo* c);
double volume(Cubo* c);

#endif
```











gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-03\$./ex01-02

Lado do cubo: 5 Lado = 5.00

Area = 25.00

Volume = 125.00

gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-03\$

roteiro-03/ex01-03.c

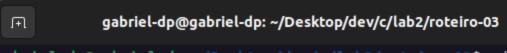
```
#include <stdio.h>
 2
   #include <stdlib.h>
 3
 4
   #include "inteiros.h"
 5
 6
   Inteiros criar() {
 7
        Inteiros novo;
 8
        novo.alocado = 100;
 9
        novo.elementos = (int*)malloc(sizeof(int) * novo.alocado);
10
        novo.tamanho = 0;
11
        return novo;
12
   }
13
14
    int tamanho(Inteiros* c) {
15
        return c->tamanho;
16
   }
17
18
   int busca(Inteiros* c, int numero) {
19
        for (int i = 0; i < tamanho(c); i++) {
20
            if (c->elementos[i] == numero) return i;
21
22
        return -1;
23
   }
24
25
   void inserir(Inteiros* c, int numero) {
26
        if (tamanho(c) >= c->alocado) {
27
            c->elementos = (int*)realloc(c->elementos, c->alocado * 2);
28
29
        c->elementos[tamanho(c)] = numero;
30
        (c->tamanho)++;
31
   }
32
33
    void remover(Inteiros* c, int posicao) {
34
        if (posicao < tamanho(c) - 1) {</pre>
35
            for (int i = posicao; i < tamanho(c); i++) {</pre>
36
                c->elementos[i] = c->elementos[i + 1];
37
38
            (c->tamanho)--;
39
        }
40
41
42
    Inteiros uniao(Inteiros* c1, Inteiros* c2) {
43
        Inteiros c3 = criar();
        for (int i = 0; i < tamanho(c1); i++) {
44
45
            inserir(&c3, c1->elementos[i]);
46
        }
47
        for (int i = 0; i < tamanho(c2); i++) {
48
            inserir(&c3, c2->elementos[i]);
49
50
        return c3;
51
   }
52
```

```
Inteiros intersecao(Inteiros* c1, Inteiros* c2) {
 53
 54
         Inteiros c3 = criar();
55
         for (int i = 0; i < tamanho(c1); i++) {
             if (busca(c2, c1->elementos[i]) != -1) {
 56
 57
                 inserir(&c3, c1->elementos[i]);
 58
             }
 59
         }
 60
         return c3;
 61
    }
 62
     Inteiros diferenca(Inteiros* c1, Inteiros* c2) {
 63
         Inteiros c3 = criar();
 64
 65
         for (int i = 0; i < tamanho(c1); i++) {
             if (busca(c2, c1->elementos[i]) == -1) {
 66
                 inserir(&c3, c1->elementos[i]);
 67
 68
             }
 69
         }
 70
         return c3;
71
    }
72
73
    int menor(Inteiros* c) {
74
         if (tamanho(c) == 0) {
 75
             return -1;
 76
         }
 77
         int min = 0;
 78
 79
         for (int i = 1; i < tamanho(c); i++) {
 80
             if (c->elementos[min] > c->elementos[i]) {
 81
                 min = i;
 82
             }
 83
         }
 84
 85
         return min;
 86
    }
 87
88
     int maior(Inteiros* c) {
89
         if (tamanho(c) == 0) {
 90
             return -1;
 91
         }
 92
 93
         int max = 0;
 94
         for (int i = 1; i < tamanho(c); i++) {
 95
             if (c->elementos[max] < c->elementos[i]) {
 96
                 max = i;
 97
             }
 98
         }
99
100
         return max;
101
     }
102
103
     int iguais(Inteiros* c1, Inteiros* c2) {
104
         int diff = 1;
         for (int i = 0; i < tamanho(c1); i++) {
105
106
             if (busca(c2, c1->elementos[i]) == -1) {
107
                 diff = 0;
108
                 break;
```

```
109
             }
110
111
112
         return diff;
113
     }
114
115
     int vazio(Inteiros* c) {
116
         return (tamanho(c) == 0);
117
     }
118
119
     void imprime(char* mensagem, Inteiros* c) {
         printf("%s", mensagem);
120
121
         for (int i = 0; i < tamanho(c); i++) {
122
             printf("%d ", c->elementos[i]);
123
124
         printf("\n");
125
     }
126
127
     int main() {
128
         Inteiros i1 = criar();
129
         inserir(&i1, 23);
130
         inserir(&i1, 12);
131
         inserir(\&i1, 5);
132
         inserir(\&i1, 47);
133
         inserir(&i1, 81);
134
         imprime("Conjunto 1 = ", &i1);
135
136
         Inteiros i2 = criar();
137
         inserir(&i2, 15);
138
         inserir(\&i2, 13);
139
         inserir(&i2, 5);
140
         inserir(&i2, 99);
141
         inserir(&i2, 81);
         imprime("Conjunto 2 = ", &i2);
142
143
144
         Inteiros une = uniao(&i1, &i2);
145
         Inteiros inter = intersecao(&i1, &i2);
         Inteiros diff = diferenca(&i1, &i2);
146
147
         imprime("Uniao = ", &une);
         imprime("Intersecao = ", &inter);
148
149
         imprime("Diferenca = ", &diff);
150
151
         printf("Igual Uniao & Intersecao = %s\n", iguais(&une, &inter) ? "sim" : "
152
         printf("Vazio Uniao = %s\n", vazio(&une) ? "sim" : "nao");
         printf("Tamanho Uniao = %d\n", tamanho(&une));
153
154
         printf("Maior Uniao = %d\n", une.elementos[maior(&une)]);
155
         printf("Menor Uniao = %d\n", une.elementos[menor(&une)]);
156
         return 0;
157 }
```

roteiro-03/inteiros.h

```
1 #ifndef INTEIROS H
   #define INTEIROS H
3
4 typedef struct {
5
       int* elementos;
6
       int tamanho, alocado;
7 } Inteiros;
  Inteiros criar();
10 int tamanho(Inteiros* c);
11 void inserir(Inteiros* c, int numero);
12 void remover(Inteiros* c, int posicao);
13 Inteiros uniao(Inteiros* c1, Inteiros* c2);
14 Inteiros intersecao(Inteiros* c1, Inteiros* c2);
15 Inteiros diferenca(Inteiros* c1, Inteiros* c2);
16 int menor(Inteiros* c);
17 int maior(Inteiros* c);
18 int busca(Inteiros* c, int numero);
19 int iguais(Inteiros* c1, Inteiros* c2);
20 int vazio(Inteiros* c);
21 void imprime(char* mensagem, Inteiros* c);
22
23 #endif
```



```
Q =
```







```
gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-03$ ./ex01-03
```

```
Conjunto 1 = 23 12 5 47 81
Conjunto 2 = 15 13 5 99 81
```

Intersecao =
$$5 81$$

Diferenca =
$$23$$
 12 47

gabriel-dp@gabriel-dp:~/Desktop/dev/c/lab2/roteiro-03\$