

INF1007 - PROGRAMAÇÃO II

LISTA DE EXERCÍCIOS 5

1. Considere uma aplicação gráfica que define dois tipos estruturados identificados pelos nomes `Ponto` e `Circulo`. O tipo estruturado `Ponto` representa o objeto ponto em duas dimensões, que é composto por duas coordenadas tipo `float`. O tipo estruturado `Circulo` representa o objeto círculo composto por um nome (que nunca contém mais do que 10 caracteres, p.ex. `Circulo015`), por um ponto que define o seu centro e por um raio com a precisão de um `float`. A Figura abaixo ilustra estas estruturas:

Ponto
X
Y

Círculo
Nome
Centro
Raio

Escreva um programa completo (com includes, structs, funções e a main), em um único arquivo `.c`, que implementa e testa as seguintes funções:

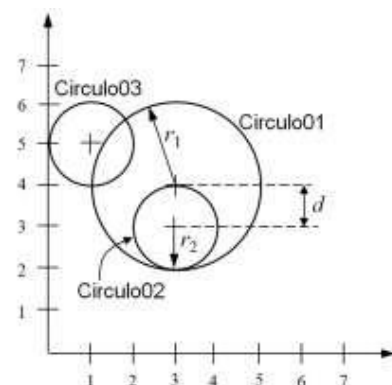
- `criaCirculo` que cria um novo círculo. Esta função recebe um nome, as coordenadas `x` e `y` do centro e o raio para este novo círculo, e retorna um ponteiro para `Circulo`.
- `contem` que testa se um determinado círculo contém outro círculo dado. Esta função recebe dois ponteiros para o tipo estruturado `Circulo` e retorna 0 se o primeiro círculo não contém o segundo círculo e diferente de zero se contém.

Um círculo c_1 contém outro círculo c_2 se a distância entre os seus centros (d) somada ao raio de c_2 for menor ou igual ao raio de c_1 , isto é: $d + r_2 \leq r_1$. A distância d entre dois pontos com coordenadas (x_1, y_1) e (x_2, y_2) é calculada pela seguinte fórmula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

A seguinte função `main` cria os 3 círculos da figura abaixo e testa se o primeiro círculo contém um dos dois outros:

```
int main(void)
{
    Circulo * c1, * c2, * c3;
    c1 = criaCirculo("Circulo01",3.0,4.0,2.0);
    c2 = criaCirculo("Circulo02",3.0,3.0,1.0);
    c3 = criaCirculo("Circulo03",1.0,5.0,1.0);
    printf("%d\n", contem(c1,c2));
    printf("%d\n", contem(c1,c3));
    free(c1);
    free(c2);
    free(c3);
    return 0;
}
```



2. Considere o tipo estruturado `PlanoDeSaude`, que representa um plano de saúde com o qual uma determinada clínica trabalha:

```
struct planoDeSaude
{
    char nomeDoPlano[21];
    int numDePacientes;
};
typedef struct planoDeSaude PlanoDeSaude;
```

Considere também o tipo estruturado `Paciente`, que representa um paciente dessa mesma clínica:

```
struct paciente
{
    char nome[51];
    int idade;
    char plano[21];
};
typedef struct paciente Paciente;
```

Implemente as seguintes funções:

- `criaNovoPaciente` - que recebe dados de um paciente (um nome (cadeia de caracteres), uma idade (um inteiro) e um plano (cadeia de caracteres)), e cria, com alocação dinâmica de memória, um novo paciente, retornando-o (o seu endereço). Caso não seja possível criar o paciente, a função retorna `NULL`.
- `processaCadastroDosPlanos` que recebe um vetor de estruturas do tipo `PlanoDeSaude` e o número de planos, um vetor de estruturas do tipo `Paciente` e o número de pacientes. Inicialmente o campo `numDePacientes` de todos os planos do vetor de planos está zerado. A função tem como objetivo armazenar em cada um dos planos de saúde o número de pacientes desse plano. Para isso ela percorre o vetor de estruturas do tipo `Paciente` e, para cada paciente, busca no vetor de planos o plano correspondente, incrementando o número de pacientes desse plano. Caso um ou mais planos não sejam encontrados, após o processamento de todos os pacientes, a função retorna 0. Se o cadastro foi processado corretamente, a função retorna 1.

Para testar o seu programa utilize a main definida abaixo:

```

int main(void)
{
    int i;
    PlanoDeSaude planos[] = {{"Amil", 0}, {"SulAmerica", 0}, {"Unimed", 0},
                              {"GoldenC", 0}};

    Paciente pacientes[] = {{"Ana", 50, "Amil"}, {"Paulo", 55, "Amil"},
                             {"Abdala", 30, "PlanoX"}, {"Beatrice", 51, "SulAmerica"},
                             {"Braz", 18, "GoldenC"}, {"Fred", 75, "Unimed"},
                             {"Cid", 80, "SulAmerica"}, {"Maria", 30, "SulAmerica"},
                             {"Ada", 15, "Unimed"}, {"Carlos", 70, "Unimed"}};

    if (processaCadastroDosPlanos(planos, 4, pacientes, 10) == 0)
        printf("Atencao: Existem pacientes com planos desconhecidos!\n\n");

    printf("Total de pacientes por plano:\n");
    for (i = 0; i < 4; i++)
    {
        printf("%s - Total: %d\n", planos[i].nomeDoPlano,
              planos[i].numDePacientes);
    }

    return 0;
}

```

3. Considere uma aplicação que gera senhas para os usuários do sistema de computação de uma empresa. O cadastro dos usuários usa uma estrutura identificada pelo nome `Usuario`. A estrutura é composta por três cadeias de caracteres: o nome do usuário (que não ultrapassa 30 caracteres), a data de nascimento do usuário no formato `dd/mm/aaaa` (e.g. `27/05/1982`), e o nome da mãe do usuário (que não ultrapassa 30 caracteres).

Usuario
Nome
Nascimento
NomeMae

Escreva um programa completo (com includes, structs, funções e a main), em um único arquivo `.c`, que implementa e testa as seguintes funções:

- `geraSenha` que gera uma nova senha com a seguinte lei de formação: os 8 primeiros caracteres correspondem à data de nascimento na ordem ano, mês e dia, os caracteres seguintes correspondem às iniciais do nome do candidato e após um `*`, vem o primeiro nome da mãe do candidato. Esta função recebe o ponteiro para um específico usuário e retorna a sua nova senha em um novo espaço alocado (do tamanho exato necessário). Por exemplo, para o usuário `{ "Rui Abreu de Lima", "27/05/1982", "Clara Soares" }`, a nova senha é `19820527RA dL*Clara`. Se não houver espaço de memória suficiente, esta função deve retornar `NULL`.

- `obtemMae` que recebe a senha de identificação de um usuário e retorna o primeiro nome da mãe deste usuário. O único argumento desta função é a cadeia de caracteres que contém a senha. Nenhum novo espaço de memória deve ser criado.

A seguinte função `main` é uma sugestão de teste para as funções pedidas:

```
int main (void)
{
    char * id;
    Usuario user01 = {"Rui Abreu Soares", "27/05/1982", "Clara Soares"};
    Usuario * u1 = &user01;
    if ((id = geraSenha(u1))!= NULL);
    {
        printf("%s\n",id);           // deve imprimir -> 19820527RAS*Clara
        printf("%s\n",obtemMae(id)); // deve imprimir -> Clara
    }
    return 0;
}
```