Exercício 1: Gerenciamento de Funcionários e Relatórios

Descrição: O código abaixo gerencia os dados dos funcionários e gera relatórios. No entanto, ele mistura responsabilidades como a manipulação dos dados dos funcionários e a geração de relatórios. Refatore o código para seguir o SRP.

Tarefa: Separe a lógica de gerenciamento de funcionários e a geração de relatórios em classes ou funções distintas.

```
#include <stdio.h>


typedef struct {
   char nome[50];
   float salario;
} Funcionario;


void GerenciarFuncionario(Funcionario *f, float novoSalario) {
   // Atualizar o salário do funcionário
   f->salario = novoSalario;


   // Gerar relatório
   printf("Relatório do Funcionário: %s\n", f->nome);
   printf("Salário: %.2f\n", f->salario);
}


int main() {
   Funcionario f = {"João", 5000.00};
   GerenciarFuncionario(&f, 6000.00);
   return 0;
}
```

REPOSTA1:

```
#include <stdio.h>

typedef struct {
    char nome[50];
    float salario;
} Funcionario;

void AtualizarSalario(Funcionario *f, float novoSalario) {
    f->salario = novoSalario;
}

void GerarRelatorio(Funcionario *f) {
    printf("Relatório do Funcionário: %s\n", f->nome);
    printf("Salário: %.2f\n", f->salario);
}

int main() {
    Funcionario f = {"João", 5000.00};
    AtualizarSalario(&f, 6000.00);
    GerarRelatorio(&f);
    return 0;
}
```

Exercício 2: Sistema de Pedidos e Notificações

Descrição: Este código é responsável por processar pedidos e também por enviar notificações para o cliente. No entanto, ele viola o SRP ao combinar essas responsabilidades. Refatore o código para separá-las.

Tarefa: Crie uma classe para processar pedidos e outra para enviar notificações, seguindo o SRP.

```
#include <stdio.h>

typedef struct {
    int id;
    float valor;
} Pedido;

void ProcessarPedido(Pedido *pedido) {
    // Processa o pedido
    printf("Processando pedido #%d de valor %.2f\n", pedido->id, pedido->valor);

    // Enviar notificação ao cliente
    printf("Enviando notificação para o cliente do pedido #%d\n", pedido->id);
}

int main() {
    Pedido p = {1, 100.00};
    ProcessarPedido(&p);
    return 0;
}
```

REPOSTA2:

```
#include <stdio.h>

typedef struct {
    int id;
    float valor;
} Pedido;

void ProcessarPedido(Pedido *pedido) {
    printf("Processando pedido #%d de valor %.2f\n", pedido->id, pedido->valor);
}

void EnviarNotificacao(Pedido *pedido) {
    printf("Enviando notificação para o cliente do pedido #%d\n", pedido->id);
}

int main() {
    Pedido p = {1, 100.00};
    ProcessarPedido(&p);
    EnviarNotificacao(&p);
    return 0;
}
```

3. Exercício 3: Manipulação de Arquivos e Processamento de Dados

Descrição: Neste código, temos a leitura e a escrita de arquivos misturadas com o processamento de dados. Sua tarefa é aplicar o SRP para separar essas responsabilidades.

Tarefa: Separe a leitura e escrita de arquivos e o processamento de dados em funções distintas.

```
#include <stdio.h>


void GerenciarArquivo(char* nomeArquivo) {
    // Leitura de arquivo
    printf("Lendo o arquivo: %s\n", nomeArquivo);

    // Processamento de dados
    printf("Processando os dados do arquivo...\n");

    // Escrita no arquivo
    printf("Escrevendo no arquivo: %s\n", nomeArquivo);
}


int main() {
    GerenciarArquivo("dados.txt");
    return 0;
}
```

REPOSTA3:

```
#include <stdio.h>

void LerArquivo(char* nomeArquivo) {
   printf("Lendo o arquivo: %s\n", nomeArquivo);
}

void ProcessarDados() {
   printf("Processando os dados do arquivo...\n");
}

void EscreverArquivo(char* nomeArquivo) {
   printf("Escrevendo no arquivo: %s\n", nomeArquivo);
}

int main() {
   LerArquivo("dados.txt");
   ProcessarDados();
   EscreverArquivo("dados.txt");
   return 0;
}
```

Exercício 4: Sistema de Cadastro e Validação de Dados

Descrição: O código abaixo mistura o cadastro de novos usuários com a validação dos dados do cadastro. Aplique o SRP para organizar essas responsabilidades de maneira mais coesa.

Tarefa: Crie uma função para o cadastro e outra para a validação de dados, garantindo que cada função tenha apenas uma responsabilidade.

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

typedef struct {
   char nome[50];
   char email[50];
} Usuario;

void CadastrarUsuario(Usuario *u) {
   // Cadastro de usuário
   printf("Cadastrando usuário: %s\n", u->nome);
   // Validação do e-mail
   if (strchr(u->email, '@') == NULL) {
      printf("Email inválido!\n");
   } else {
      printf("Email válido.\n");
   }
}
int main() {
   Usuario u = {"Maria", "maria.exemplo.com"};
   CadastrarUsuario(&u);
   return 0;
}
```

EXERCICIO4:

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

typedef struct {
    char nome[50];
    char email[50];
} Usuario;

void CadastrarUsuario(Usuario *u) {
    printf("Cadastrando usuário: %s\n", u->nome);
}

int ValidarEmail(char *email) {
    if (strchr(email, '@') == NULL) {
        printf("Email inválido!\n");
        return 0;
    } else {
        printf("Email válido.\n");
        return 1;
    }
}

int main() {
    Usuario u = {"Maria", "maria.exemplo.com"};
    if (ValidarEmail(u.email)) {
        CadastrarUsuario(&u);
    }
    return 0;
}
```

Exercício 5: Relógio e Alarme

Descrição: O código mistura a funcionalidade de um relógio com a de um alarme. Refatore o código para aplicar o SRP, criando classes ou funções específicas para cada responsabilidade.

Tarefa: Separe a lógica do relógio e a do alarme em funções ou classes separadas.

```
#include <stdio.h>

#include <time.h>


void RelogioAlarme(int horaAlarme) {

   // Mostrar a hora atual

   time_t t;

   time(&t);

   printf("Hora atual: %s", ctime(&t));


   // Verificar alarme

   struct tm* localTime = localtime(&t);

   if (localTime->tm_hour == horaAlarme) {

      printf("Alarme! Acorde!\n");

   }

}


int main() {

   RelogioAlarme(8);  // Definir alarme para 8 horas

   return 0;

}
```

EXERCICIO5:

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

void MostrarHoraAtual() {
   time_t t;
   time(&t);
   printf("Hora atual: %s", ctime(&t));
}

void VerificarAlarme(int horaAlarme) {
   time_t t;
   time(&t);
   struct tm* localTime = localtime(&t);
   if (localTime->tm_hour == horaAlarme) {
      printf("Alarme! Acorde!\n");
   }
}

int main() {
   MostrarHoraAtual();
   VerificarAlarme(8);  // Definir alarme para 8 horas
   return 0;
}
```