

Trabalho Prático Final

Entrega Final: Documentação

Alunos: Gustavo Matos Calixto
Lucas Anselmo Martins Motta
Raphael Ferreira Ribeiro
Victor Ragonezi Amaral

Professor: Julio Cesar S. Reis
Turma: TF2

Introdução

Visando o desenvolvimento de um software para auxiliar na implantação de um projeto de coleta seletiva e também, de modo a atender as orientações propostas pelo professor Julio Cesar, o grupo optou pelo uso da linguagem C++ bem como pela utilização de um banco de dados MySQL (funcionalidade extra) para o sistema.

O usuário assume dois comportamentos simultâneos dentro do sistema, tanto o de doador quanto o de receptor dos resíduos destinados à coleta seletiva. O usuário - enquanto doador - pode, no ambiente responsável para o cadastro de resíduos, consultar a forma adequada de armazenamento do mesmo, tal como informar sua quantidade disponível para a doação. Além disso, enquanto doador, o usuário também pode acessar os resíduos que cadastrou para doação e removê-los (se for o caso).

Já o usuário, enquanto receptor, pode, a partir de uma lista prévia de todos os resíduos disponíveis para doação, selecionar o resíduo de seu interesse, fixar a data de coleta do resíduo e sugerir ao doador, o local para a coleta do mesmo.

Por fim, o usuário também terá acesso às socilitações dos resíduos, tanto as que fez, quanto as que recebeu, e por meio delas poderá confirmar ou cancelar a coleta. Ademais, o cliente similarmente poderá visualizar o ranking baseado nos usuários com a maior quantidade de doações no sistema.

Implementação

A implementação do sistema foi facilitada devido às funcionalidades descritas previamente através das User Stories e dos Cartões CRC, o primeiro passo adotado foi a criação de uma função “main” para fazer as principais requisições às classes e uma classe para cadastro do usuário, atendendo às necessidades de informações propostas.

Posteriormente, foi criada a classe para conexão com o banco de dados e todas as tabelas necessárias. A classe ‘Telas’ foi desenvolvida como o objetivo de auxiliar na dinâmica dos ambientes no interior do sistema. Além dela, foi desenvolvida a classe ‘Auxiliar’, que tem o papel de auxiliar na parte da validação dos dados do usuário, bem como as opções escolhidas por ele dentro do sistema.

Foram utilizados conceitos sobre programação orientada a objetos no programa, como por exemplo:

- Herança, utilizado por exemplo na classe Usuário, que por sua vez herda os métodos e atributos da classe Pessoa;
- Encapsulamento, de forma a definir quais partes do programa têm acesso aos dados das várias classes contidas no mesmo.

Um comentário a se destacar é que decidimos colocar as quantidades dos resíduos em string e não em inteiro devido às unidades de medidas, abrindo mão de uma padronização delas, de modo a atender as necessidades de cada doador.

Além disso, também foram utilizados conceitos de programação defensiva, de forma a evitar que o código seja “quebrado” e evitando erros no código que podem acontecer devido ao comportamento do usuário. Exemplos disso podem ser vistos com mais detalhe no próximo tópico.

Testes

- Validação de CPF/CNPJ

```
-----  
SISTEMA DE GESTAO DE RESIDUOS  
-----  
CADASTRO  
CPF/CPNJ: 12345678910 CPF Inválido  
Documento invalido! Tente novamente: 1234H5J7KT CPF com caracteres  
Documento invalido! Tente novamente: 000000000000 CPF composto de zeros  
Documento invalido! Tente novamente: 14223513001 CPF Válido  
Nome: _
```

```
1  bool Auxiliar::validaCPF(const int *const cpf)  
2  {  
3      int digito1, digito2, temp = 0;  
4      // Primeiro digito verificador  
5      for (char i = 0; i < 9; i++)  
6          temp += (cpf[i] * (10 - i));  
7      if (temp == 0)  
8          digito1 = 11;  
9      else {  
10         temp %= 11;  
11         if (temp < 2)  
12             digito1 = 0;  
13         else  
14             digito1 = 11 - temp; }  
15     // Segundo digito verificador  
16     .  
17     .  
18     .  
19     // Digtos Verificadores batem com o input?  
20     if (digito1 == cpf[9] && digito2 == cpf[10])  
21         return true;  
22     else  
23         return false;  
24 }
```

Trecho (incompleto) da função responsável pela validação do CPF. O algoritmo se baseia nos dígitos de 1 a 9 do CPF informado pelo cliente e partir deles, calcula qual deverá ser o 10º e 11º dígito para que o CPF seja válido e logo após verifica se correspondem aos dígitos informados pelo usuário.

Porém havia um caso específico de bug: quando os onze dígitos eram zeros. Numa das etapas para o cálculo do dígito verificador é feita a soma de várias multiplicações (linha 6), a correção para esse erro foi adicionar um laço (linha 7) para checar se essa soma é zero (o que ocorre quando todos os dígitos são zero) e caso seja, atribui um número de dois dígitos (linha 8) ao 1º dígito verificador. Portanto, caso isso ocorra, o laço (linha 20) que compara o dígito verificador com o informado pelo usuário sempre retornará falso, já que o dígito do usuário é único e o verificador é composto por dois números.

- Validação das opções

```
-----  
          SISTEMA DE GESTAO DE RESIDUOS  
-----  
1.Cadastrar residuo para doacao  
2.Lista de residuos disponiveis  
3.Residuos que cadastrei  
4.Solicitacoes  
5.Ranking de doadores  
0.Sair  
  
Digite a opcao desejada: 12  
  
Opcao invalida! Tente novamente: h  
  
Opcao invalida! Tente novamente: -1
```

```
int Auxiliar::getOpcao(int primeira, int ultima)  
{  
    int opcao;  
    bool opcaoIncorreta;  
    std::cout << " Digite a opcao desejada: ";  
    do  
    {  
        opcaoIncorreta = !Auxiliar::numero(opcao) ||  
            !(opcao >= primeira && opcao <= ultima);  
        if (opcaoIncorreta)  
        {  
            std::cout << "\n Opcao invalida! Tente novamente: ";  
        }  
    } while (opcaoIncorreta);  
    return opcao;  
}
```

```
1 void Telas::menuPrincipal(Usuario &usuario)  
2 {  
3     int opcao;  
4     bool opcaoIncorreta;  
5     Telas::cabecalho();  
6     std::cout << " 1.Cadastrar residuo para doacao" << std::endl;  
7     std::cout << " 2.Lista de residuos disponiveis" << std::endl;  
8     std::cout << " 3.Residuos que cadastrei" << std::endl;  
9     std::cout << " 4.Solicitacoes" << std::endl;  
10    std::cout << " 5.Ranking de doadores" << std::endl;  
11    std::cout << " 0.Sair\n" << std::endl;  
12    opcao = Auxiliar::getOpcao(0, 5);
```

Trecho (incompleto) da função responsável por validar a opção desejada pelo usuário. Uma função recebe dois números como parâmetro, sendo um deles, o menor número de acordo com a opções possíveis e outro, o maior (linha 12). Através deles, a função só aceita como resposta um número inteiro compreendido nesse intervalo. Nesse caso, apenas um número inteiro compreendido entre 0 e 5.

Outros exemplos utilizando a mesma função:

```
int Telas::escolherTipoResiduo()
{
    std::cout << "\nSelecione o tipo do residuo:" << std::endl;
    std::cout << " 1.Solido" << std::endl;
    std::cout << " 2.Liquido" << std::endl;
    return Auxiliar::getOpcao(1, 2);
}
```

```
void Telas::cadastrarResiduo(Usuario &usuario)
{
    Telas::cabecalho();
    std::cout << " 1.Cadastrar um residuo" << std::endl;
    std::cout << " 0.Voltar" << std::endl;
    int opcao = Auxiliar::getOpcao(0, 1);
}
```

- Confirmação de Senha

```
do
{
    std::cout << " Senha: ";
    std::getline(std::cin, senha);
    std::cout << " Confirmar Senha: ";
    std::getline(std::cin, confirmacaoSenha);
    if (senha != confirmacaoSenha)
    {
        senhaIncorreta = true;
        std::cout << "\n As senhas nao coincidem! Tente novamente \n\n";
    }
    else
    {
        senhaIncorreta = false;
    }
} while (senhaIncorreta);
```

```
Senha: teste
Confirmar Senha: TESTE

As senhas nao coincidem! Tente novamente

Senha: teste
Confirmar Senha: te5Te

As senhas nao coincidem! Tente novamente
```

Trecho (incompleto) da função responsável pela confirmação da senha. Enquanto as senhas não coincidirem, o sistema segue pedindo ao usuário uma senha, até que a senha e a sua confirmação sejam iguais.

Conclusão

Algumas dificuldades surgiram durante a realização do projeto. As principais foram:

- Sintaxe e funções das bibliotecas padrões do C++, dificuldades que foram rapidamente solucionadas;
- Implementação do sistema ao aplicar simultaneamente os conceitos da programação defensiva;
- Conexão e implementação do funcionamento correto do banco de dados com as operações realizadas pelo C++;
- Problemas com alocação, erros de segmentação fault, problemas com o retorno de variáveis por endereço.

Por fim, conclui-se que o desenvolvimento de um projeto modularizado permitiu agilidade, transparência e integração dos diversos códigos produzidos pelas partes envolvidas, proporcionando maior fluidez apesar de todas as dificuldades técnicas encontradas.

O Projeto final pode ser encontrado em:

<https://github.com/ragonezi/garbage-collection>

Bibliografia



SPOLSKY, J. E ATWOOD, J.

Stack Overflow - Where Developers Learn, Share, & Build Careers

Bibliografia: Spolsky, J. and Atwood, J. (2008). Stack Overflow - Where Developers Learn, Share, & Build
_ Careers. Stack Overflow. Disponível em: <https://stackoverflow.com/>



BERBERT DE PAULA, F.

Viva o Linux - A maior comunidade GNU/Linux da América Latina!

Bibliografia: Berbert de Paula, F. (2002). Viva o Linux - A maior comunidade GNU/Linux da América
_ Latina!. Viva o Linux. Disponível em: <https://www.vivaolinux.com.br/>



BODNAR, J.

MySQL C API programming tutorial

Bibliografia: Bodnar, J. (2007). MySQL C API programming tutorial. ZetCode. Disponível em: <http://zetcode.com/db/mysqlc/>



MYSQL :: MYSQL 5.7 REFERENCE MANUAL :: 27.8 MYSQL C API

Bibliografia: Dev.mysql.com. MySQL :: MySQL 5.7 Reference Manual :: 27.8 MySQL C API. Disponível
_ em: <https://dev.mysql.com/doc/refman/5.7/en/c-api.html>

ECKEL, B. E ALLISON, C.

Thinking in C++

Texto do livro: (Eckel and Allison, 1999)

Bibliografia: Eckel, B. and Allison, C. (1999). Thinking in C++. 2nd ed. Pearson.

SAVITCH, W.

Absolute C++

Texto do livro: (Savitch, 2004)

Bibliografia: Savitch, W. (2004). Absolute C++. 1st ed. Pearson.