Aplicativo bancário em C

Utilizando o paradigma imperativo.

Programação Orientada a Objetos
Professora Adriana Carla Damasceno

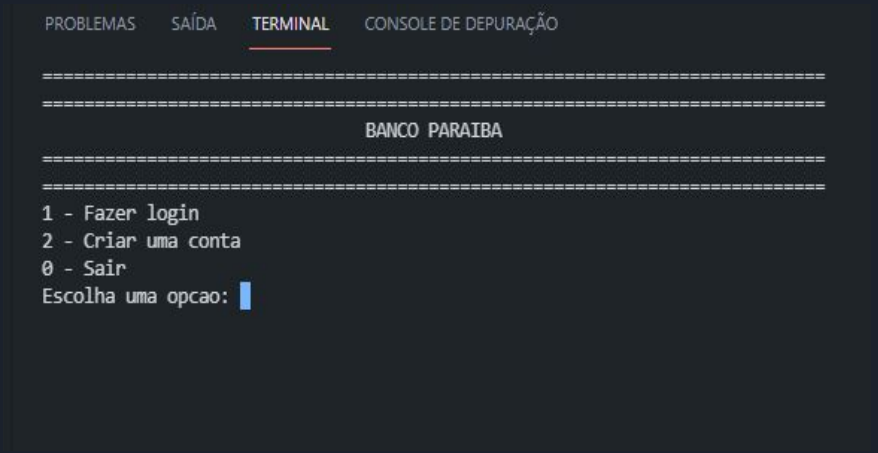
Participantes
Lucas Miranda de Aguiar
Filipe de Medeiros Santos
Deivison Rodrigues Jordao

Universidade
Federal da Paraíba

Curso de Ciência de Dados e IA - 2020.1

Visão Geral

- Explicação inicial do tema
  - Ideia geral
  - Operações em db
- Implementação
  - Tipos estruturados
  - FILE
- Erros na implementação
- Exposição do código



Explicação inicial do tema

- Ideia geral
  - Criação de um aplicativo de banco
- Operações em database
  - Aplicação do CRUD

Implementação teórica

- Tipo estruturado:
  - Definição da struct client

```
struct client{
    char accnt[7];
    char pass[9];

    float balance;

    int check;
    int block;
};
```

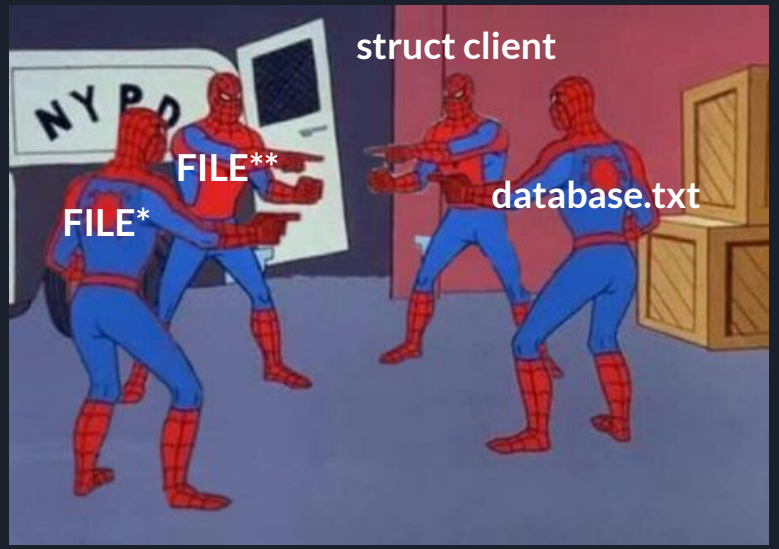
Implementação teórica

- Acessando dados em arquivo:
  - Base de dados em arquivo de texto
  - FILE*

```
archive = fopen("database.txt", "r+");

n_clients = 0;
while (!feof(archive)) {
    get_client(archive, &users[n_clients]);

    if (!strcmp(users[n_clients].accnt, account))
        user = n_clients;

    n_clients += 1;
}
```

Erros na implementação

- Dificuldades na identificação de erros
  - Ponteiro para arquivo



Erros na implementação

- Fechamento do arquivo no programa
  - Diferença entre free() e fclose()

Erros na implementação

```
85
86 void new_account(FILE *data, char accnt[], char pass[], char dataset[]) {
87     data = fopen(dataset, "a");
88     fprintf(data, "\n%s;%s;0;0.000000", accnt, pass);
89
90     fflush(data);
91     free(data);
92 }
93
```

bank.cpp | database.txt

database.txt

```
1  100000;12345678;0;0.000000
2  100001;23456789;1;45.630001
3  100002;29990221;0;90.080002
4  100003;01201201;0;38.8000499
```

PROBLEMAS SAÍDA TERMINAL CONSOLE DE DEPURAÇÃO

```
==========
==========
BANCO PARAIBA
==========
==========
Conta criada com sucesso!
Sua conta sera 100004
1 - Tela inicial
0 - Sair
Escolha uma opcao:
```

```
void new_account(FILE *data, char accnt[], char pass[], char dataset[]) {
    data = fopen(dataset, "a");
    fprintf(data, "\n%s;%s;0;0.000000", accnt, pass);

    fflush(data);
    fclose(data);
}
```

bank.cpp | database.txt

database.txt

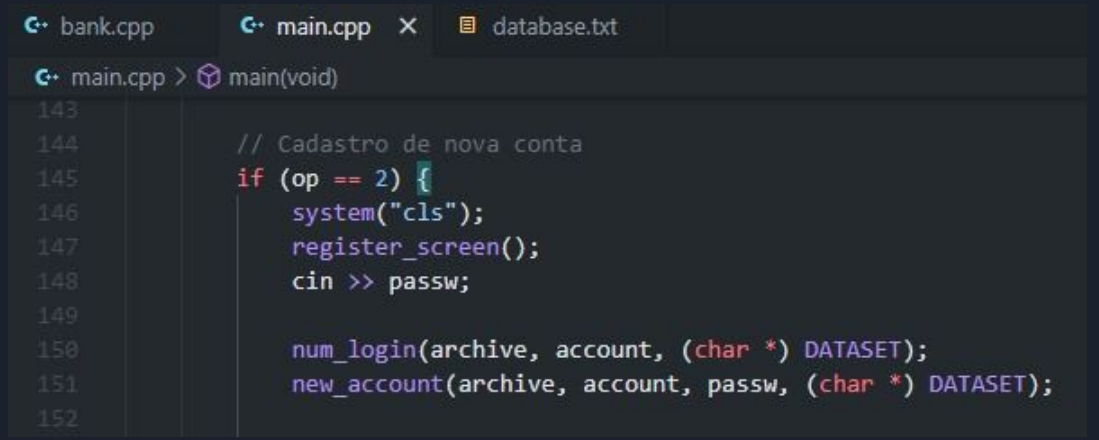
```
1  100000;12345678;0;0.000000
2  100001;23456789;1;45.630001
3  100002;29990221;0;90.080002
4  100003;01201201;0;38.8000499
5  100004;515253;0;0.000000
```

PROBLEMAS SAÍDA TERMINAL CONSOLE DE DEPURAÇÃO

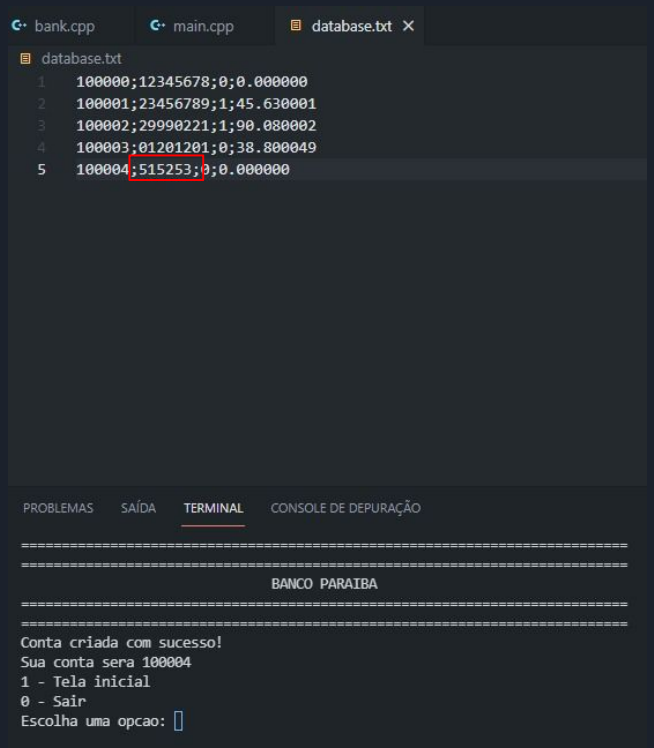
```
==========
==========
BANCO PARAIBA
==========
==========
Conta criada com sucesso!
Sua conta sera 100004
1 - Tela inicial
0 - Sair
Escolha uma opcao:
```

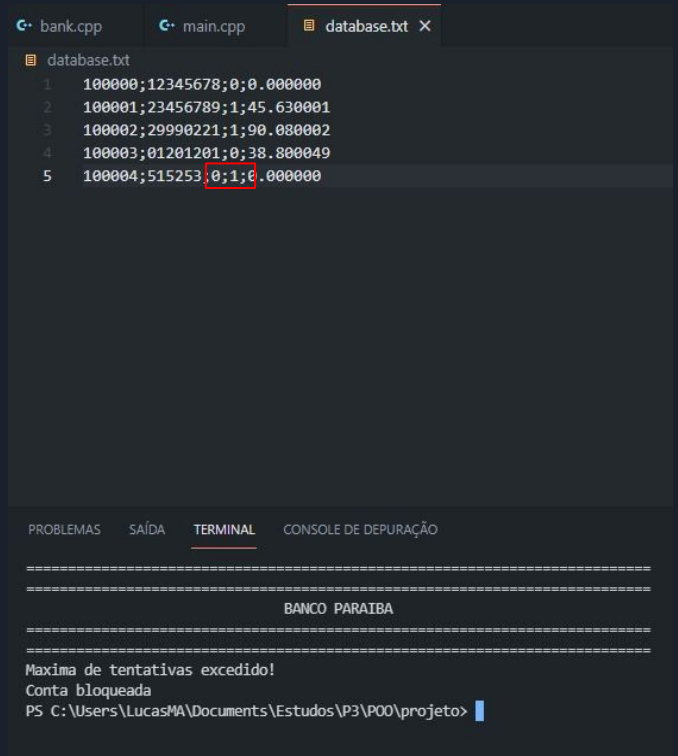
Erros na implementação

- Ausência de limitação de caracteres na senha
  - Atualização degradada dos dados



Erros na implementação





Exposição do código

- bank.h

```c
#define BANK_H


// Declaração de tipo estruturado com dados necessários
struct client{
    char accnt[7];
    char pass[9];

    float balance;

    int check; // inicia com 0, é incrementado para um pela função de verificação de senha
    int block; // inicia com 0, é incrementado no terceiro erro da senha
};

/************************
* Protótipo das funções *
************************/

/* Tela do caixa eletrônico */
int bank_screen(char *accnt);
/* Leitura dos dados dos clientes */
void get_client(FILE *data, struct client *current);
/* Cabeçalho de apresentação do banco */
void header(void);
/* Tela para inserção de dados do cliente */
void login_screen(void);
/* Menu do banco */
int main_screen(void);
/* Registro de uma nova conta */
void new_account(FILE* data, char accnt[], char pass[], char dataset[]);
/* Gera o próximo numero de conta disponível de acordo com o dataset */
void num_login (FILE *data, char *n_accnt, char dataset[]);
/* Tela para inserção de senha */
void pass_screen(int);
/* Inserção de senha em uma nova conta  */
void register_screen(void);
/* "UPDATE DATABASE", realiza as atualizações após a finalização das operações */
void upd_db(FILE *data, struct client current[], int rows);

#endif
```

Exposição do código

- bank.cpp
  - header()

```cpp
// Cabeçalho do banco
void header(void) {
    cout << "================================================================================" << endl;
    cout << "================================================================================" << endl;
    cout << "                                BANCO PARAIBA                                   " << endl;
    cout << "================================================================================" << endl;
    cout << "================================================================================" << endl;
}
```

Exposição do código

- bank.cpp
  - get_cliente()

```
archive = fopen("database.txt", "r+");

// Leitura de todo o dataset atribuindo a user a posição do usuário d
n_clients = 0;
while (!feof(archive)) {
    get_client(archive, &users[n_clients]);

    if (!strcmp(users[n_clients].accnt, account)) // Verificando igua
        user = n_clients;

    n_clients += 1;
}
```

```
// Função para a leitura dos dados de um cliente
void get_client(FILE *data, struct client *current) {
    char grb;

    fgets(current->accnt, 7, data);
    fscanf(data, "%c", &grb); // ignorando caracteres que não compõem os dados pertinentes
    fgets(current->pass, 9, data);
    fscanf(data, "%c", &grb);
    fscanf(data, "%d", &current->block);
    fscanf(data, "%c", &grb);
    fscanf(data, "%f", &current->balance);
    fscanf(data, "%c", &grb);

    current->check = 0;
}
```

Exposição do código

- bank.cpp
  - num_loguin()
  - new_account()

```
void num_login (FILE *data, char *n_accnt, char dataset[]) {
    char grb[24];

    data = fopen(dataset, "r+");

    // Leitura de todo o dataset até o último número de conta
    while (!feof(data)) {
        fgets(n_accnt, 7, data);
        fgets(grb, 40, data); // ignorando caracteres que não
    }

    n_accnt[5]++;
}
```

```
void new_account(FILE *data, char accnt[], char pass[], char dataset[]) {
    data = fopen(dataset, "a");
    fprintf(data, "\n%s;%s;0;0.000000", accnt, pass);

    fflush(data);
    fclose(data);
}
```

Exposição do código

- bank.cpp
  - bank_screen(), loguin_screen(), main_screen()

```cpp
// Tela do caixa eletrônico
int bank_screen(char *accnt) {
    int op;

    header();

    cout << "Bem-vindo " << accnt << endl;
    cout << "1 - Consultar saldo" << endl;
    cout << "2 - Deposito" << endl;
    cout << "3 - Saque" << endl;
    cout << "0 - Sair" << endl;

    cout << "Escolha uma opcao: ";
    cin >> op;

    return op;
}
```

```cpp
// Função para a apresentação da tela de captura de conta
void login_screen(void) {
    header();

    cout << "Insira o numero da sua conta: ";
}

// Função para a apresentação da tela inicial
int main_screen(void) {
    int op;

    header();

    cout << "1 - Fazer login" << endl;
    cout << "2 - Criar uma conta" << endl;
    cout << "0 - Sair" << endl;

    cout << "Escolha uma opcao: ";
    cin >> op;

    return op;
}
```

Exposição do código

- bank.cpp
  - pass_screen(), register_screen()

```
void pass_screen(int att) {

    header();

    if (att)
        cout << "Restam " << 3 - att << " tentativas!\nInsira sua senha: ";
    else
        cout << "Insira sua senha: ";
}

// Função para a apresentação da tela de criação de conta
void register_screen(void) {
    int op;

    header();

    cout << "Insira uma senha com 8 caracteres: ";
}
```

Exposição do código

- bank.cpp
  - pass_screen(), register_screen()

```
// Função para a atualização do arquivo com o dataset
void upd_db(FILE *data, struct client current[], int rows) {
    rewind(data);

    for (int j=0; j<rows; j++) {
        if (j == (rows-1))
            fprintf(data, "%s;%s;%d;%f", current[j].accnt, current[j].pass, current[j].block, current[j].balance);
        else
            fprintf(data, "%s;%s;%d;%f\n", current[j].accnt, current[j].pass, current[j].block, current[j].balance);
    }

    fclose(data);
}
```

Funcionamento do código