



Doce sob Medida

Gabriel dos Santos Pessoli
Lucas Darós Parente
Marcus Vinícius de Souza Gomes
Pedro Henrique Pereira de Almeida



Sumário

- Problemas
- Justificativa
- Objetivos do projeto
- Conceitos utilizados
- Conclusões
- Perguntas



Problemas Encontrados

- Dificuldade para calcular quantidades de ingredientes para diferentes porções
- Desperdício de ingredientes
- Dificuldade de organizar e juntar todas as receitas numa lista



Justificativa



- Auxiliar no processo de produção da comunidade
- Facilitar a vida de confeitores
- Futuramente evoluir para um aplicativo mais avançado



Objetivos do projeto

- Facilitar a execução de receitas de diferentes tamanhos
- Evitar o possível desperdício de ingredientes
- Elevar a eficiência da preparação de diferentes doces
- Padronizar e potencializar processos de confeitaria antes feitos de maneira imprecisa, maximizando a eficiência do cotidiano profissional do público alvo.
- Incrementar os lucros de pequenas confeitarias



Conceitos utilizados



- Linguagem C
 - estruturas condicionais
 - estruturas de repetição
 - vetores/matrizes
 - struct
 - funções
 - indentação
 - nomes de variáveis representativas
 - validação da entrada de dados



Estruturas condicionais

```
if (total_receita >= NUM_MAX_CAD) {  
    printf("\nErro: O cadastro já atingiu sua capacidade máxima!\n");  
    system("pause");  
    break;  
}
```

```
if (count_ingrediente >= MAX_INGREDIENTES) {  
    printf("Erro: Número máximo de ingredientes atingido!\n");  
    break;  
}
```

Estruturas de repetição

```
do {
    if (count_ingrediente >= MAX_INGREDIENTES) {
        printf("Erro: Número máximo de ingredientes atingido!\n");
        break;
    }

    printf("\nNome do ingrediente a ser adicionado na receita: ");
    scanf(" %255[^\n]", vetor_receita[total_receita].nome_ingrediente[count_ingrediente]);

    printf("\nDigite a quantidade do ingrediente: ");
    scanf("%f", &vetor_receita[total_receita].quant_ingred[count_ingrediente]);

    printf("\nDigite a unidade de medida do ingrediente: ");
    scanf(" %255[^\n]", vetor_receita[total_receita].medida[count_ingrediente]);

    vetor_receita[total_receita].count_ingrediente = ++count_ingrediente;

    printf("\nVocê deseja adicionar mais algum ingrediente? (s/n) ");
    scanf(" %c", &ch);
    ch = toupper(ch);
} while (ch == 'S');
```




Vetor e matriz

```
float quant_ingred[MAX_INGREDIENTES];  
char nome_ingrediente[MAX_INGREDIENTES][256]; //Matriz de string  
int count_ingrediente;  
char medida[20][256]; //Armazena a unidade de medida de cada ingrediente
```



Struct

```
typedef struct Receita {  
    int ocupado;  
    char nome_receita[256];  
    char cod_receita[3]; // O código tem 2 caracteres, sobra espaço para '\0'  
    char tipo_rendimento[256];  
    float rendimento; // Virou float pra dar pra fazer conta  
  
    float preco;  
  
    float quant_ingred[MAX_INGREDIENTES];  
    char nome_ingrediente[MAX_INGREDIENTES][256]; //Matriz de string  
    int count_ingrediente;  
    char medida[20][256]; //Armazena a unidade de medida de cada ingrediente  
} Receita;
```



Funções

```
void gerarArquivoReceitas(Receita vetor_receita[], int total_receita) {  
  
    //comando para gerar o arquivo  
    FILE *file = fopen("receitas.txt", "w");  
    if (file == NULL) {  
        printf("Erro ao criar o arquivo.\n");  
        return;  
    }  
}
```



```
fprintf(file, "Total de Receitas Cadastradas: %d\n\n", total_receita);

// Percorre todas as receitas e copia pro arquivo
for (int i = 0; i < total_receita; i++) {
    if (vetor_receita[i].ocupado) {
        fprintf(file, "Código da receita: %s\n", vetor_receita[i].cod_receita);
        fprintf(file, "Nome da receita: %s\n", vetor_receita[i].nome_receita);
        fprintf(file, "Ingredientes e quantidades:\n");

        //laço para imprimir todos os ingredientes salvos na receita
        for (int j = 0; j < vetor_receita[i].count_ingrediente; j++) {
            fprintf(file, "%s: %.2f %s\n",
                    vetor_receita[i].nome_ingredient[j],
                    vetor_receita[i].quant_ingred[j],
                    vetor_receita[i].medida[j]);
        }

        fprintf(file, "Preço: R$ %.2f\n", vetor_receita[i].preco);
        fprintf(file, "Rendimento: %.1f %s\n", vetor_receita[i].rendimento, vetor_receita[i].tipo_rendimento);
        fprintf(file, "\n-----\n\n");
    }
}

//fecha o arquivo depois de todas as receitas da memória terem sido adicionadas
fclose(file);
printf("Arquivo 'receitas.txt' gerado com sucesso!\n");
}
```



Conclusões



- A maior precisão reduz desperdício, economiza tempo e aumenta a qualidade do produto
- O armazenamento e impressão das receitas e cálculo rápido de porções agiliza o processo de confecção
- A padronização das receitas torna o produto mais profissional



Perguntas ?



Fim