Estrutura básica de um programa em C

Created	@27 de fevereiro de 2025 08:35
Class	INFORMÁTICA PARA A ENGENHARIA

Estrutura Básica de um Programa em C

1. Hello World: Escreva um programa em C que exiba "Olá, Mundo!" na tela.

```
#include <stdio.h>

int main(void){
printf("Ola Mundo");
getchar();
return 0;
}
```

2. Inclusão de Biblioteca: Verifique em qual biblioteca temos as funções e apresente programas que usem essa funções (Livro C Completo e Total a partir da página 281):

a. fprintf() - **<stdio.h>**

e fread() - **<stdio.h>**

d. pow() e sqrt() - **<math.h>**

b. isdigit() e isupper() - **<ctype.h>**

e. ctime() - <time.h>

c. toupper() e tolower() - <ctype.h>

f. atof() e atoi() - **<stdlib.h>**

Tipos Básicos de Dados

3. Declaração de Variáveis: Crie um programa que declare variáveis dos tipos char, int, float, e double, atribua valores a elas e exiba esses valores na tela.

```c
#include <stdio.h>

int main() {
    char a = 'a';
    int b = 3;
    float c = 1.5;
    double d = 0.1;

    printf("%c\n", a);
    printf("%d\n", b);
    printf("%f\n", c);
    printf("%lf\n", d);
    return 0;
}
```

4. Tamanho dos Tipos: Escreva um programa que use a função sizeof para exibir o tamanho em bytes de cada um dos tipos básicos de dados (char, int, float, double).

```c
#include <stdio.h>

int main() {
    printf("%u", sizeof(char));
    printf("%u", sizeof(int));
    printf("%u", sizeof(float));
    printf("%u", sizeof(double));
    return 0;
}
```

```
1
4
4
8
```

Modificadores de Tipos Básicos

5. Overflow de Inteiros: Crie um programa que demonstre o que acontece quando ocorre um overflow em uma variável do tipo int.

```
#include <stdio.h>

int main() {
    int a = 2147483648;
    printf("%d", a);
    return 0;
}
```

Devido ao overflow, o sistema “reinicia” o valor da variável, já que como o limite de memória de uma variável int é 2147483647, um número maior que isso acaba causando o overflow, por exemplo, se o usuário denominar o valor 2147483648 a uma variável int, o computador apresenta -2147483648 ao invés dos 2147483648 previstos.

6. Signed vs Unsigned: Declare uma variável unsigned int e uma signed int, atribua a ambas o valor 3.000.000.000 (três milhões) e exiba seus valores. Explique a diferença.

```
#include <stdio.h>

int main() {
signed int a = 3000000000;
unsigned int b = 3000000000;
printf("%d\n", a);
printf("%u\n", b); //por algum motivo se usar %d ele considera que é signed ir

    return 0;
}
```

Quando utilizamos o comando printf para exibir os valores, podemos notar que o signed int apresenta overflow, visto que o limite de um signed int é de ~2.000.000.000, já o unsigned tem um limite de ~4.000.000.000, por isso, apresenta o valor corretamente.

Identificadores

7. Nomes Válidos e Inválidos: Liste cinco nomes de identificadores válidos e cinco inválidos, explicando por que cada um é válido ou inválido.

Válidos

- _a
- b_test
- c_print
- d_float
- e_int

Todas estas são válidas pois não começam com números, não utilizam palavras reservadas da linguagem, (as palavras reservadas podem ser usadas caso estejam de acordo com os exemplos acima.) também não há nenhum dígito especial (!, @, %, ., etc) nem espaços.

Inválidos

- !a - inválido por iniciar com caracter especial
- 23b - inválido por iniciar com um número
- int - inválido por usar uma palavra reservada da linguagem
- variavel...float - inválido por usar ...
- @teste - inválido por iniciar com @.

Variáveis

8. Variáveis Locais e Globais: Escreva um programa com uma variável global e uma local com o mesmo nome. Dentro de uma função, exiba o valor de cada uma para demonstrar o escopo das variáveis.

```
#include <stdio.h>
int a = 3; //variável global

int funcao(){
    int a = 4; //variável local
```

```
    printf("%d\n", a);
}

int main(void){
printf("%d\n", a);
funcao();
    return 0;
}
```

9. Modificador static: Crie uma função que use uma variável static para contar quantas vezes ela foi chamada e exiba o número de chamadas.

```
#include <stdio.h>
int c(){
    static int a = 0;
    a++;
    printf("%d\n", a);
}

int main() {
  for(int i = 0; i<9; i++){
  c();
}
    return 0;
}
```

Com o resultado:

1
2
3
4
5
6
7
8
9

10. Constantes: Declare uma constante Pl usando o modificador const e use-a para calcular a área de um círculo dado o raio.

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

const float pi = 3.14159265358979323846;

int main() {
int raio;
scanf("%d", &raio);
int area = pi * pow(raio, 2);
printf("%d", area);

return 0;
}
```

11. Static em Funções: Modifique o exercício anterior para que a variável static mantenha seu valor entre chamadas de função, demonstrando seu uso.

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

const float pi = 3.14159265358979323846;

int func(int raio){
static int c;
float area = pi * pow(raio, 2);
c++;
printf("Area %f, Vezes que a funcao foi chamada: %d\n", area, c);
}

int main() {
int raio;
scanf("%d", &raio);
func(raio);
```

```
func(2);
return 0;
}
```

Inicialização de Variáveis em Diferentes Bases Numéricas

12. Operações com Diferentes Bases: Realize operações aritméticas básicas (soma, subtração) com números inicializados em diferentes bases e exiba os resultados em decimal.

```
#include <stdio.h>

int hex = 0xA; //A = 10
int octal = 040; // 40 = 32
int binario = 0b101; // 101 = 5

int main(){
    int a = hex + octal;
    int b = octal - binario;
    int c = binario * hex;
    printf("%d\n", a);
    printf("%d\n", b);
    printf("%d\n", c);

    return 0;
}
```