Algoritmos e Programação Estruturada

Lista 12

1. **Sobre a seguinte estrutura:**

|  |
| --- |
| typedef struct Aluno {  char nome[100]; int faltas; float nota;  }aluno\_t; |

1

2

3

4

5

6

Crie um programa em C que carregue dinamicamente (via malloc) um novo aluno, carregue os dados dele e exiba em um printf.

Apresente também se ele foi ou não aprovado.

ARQUIVO C EM ANEXO

1. **Acerca de alocação dinâmica de memoria**

Responda:

1. Qual o comportamento da função free?
2. Após chamar free, o ponteiro pode ser utilizado?
3. O que causa vazamentos de memoria?
4. O que a instrução malloc retorna quando não consegue realizar a alocação?
5. Explique a instrução calloc.
6. Qual a diferença entre as instruções malloc e calloc?

**2. Alocação Dinâmica de Memória**

**a)** O comportamento da função free é liberar a memória alocada dinamicamente, permitindo que o sistema operacional possa reutilizá-la. Após o uso de free, a memória é desmarcada para reutilização, mas o ponteiro que apontava para ela se torna um "ponteiro pendente", não mais válido.

1. **b)** Após chamar free, o ponteiro não deve ser utilizado diretamente. O ponteiro se torna inválido, e acessá-lo pode causar comportamentos imprevisíveis, como falhas no programa. Uma boa prática é definir o ponteiro como NULL após chamar free para evitar o uso acidental.
2. **c)** Vazamentos de memória ocorrem quando a memória é alocada dinamicamente (usando malloc, calloc ou realloc), mas não é liberada adequadamente com free. Isso faz com que o programa "perca" a referência a essa área de memória, que nunca poderá ser reutilizada.
3. **d)** Quando malloc não consegue realizar a alocação, ela retorna NULL.
4. **e)** A instrução calloc aloca memória de forma semelhante a malloc, mas também inicializa a memória alocada com zero.
5. **f)** A diferença entre malloc e calloc é que malloc apenas aloca a memória sem inicializá-la, enquanto calloc aloca a memória e a inicializa com zero.

**3 Acerca de Manipulação de Arquivos.**

1. Explique os diferentes modos de abertura de arquivos.
2. Explique o funcionamento das funções fgets, fprintf, fread, fwrite.

**3. Manipulação de Arquivos**

**a)** Os diferentes modos de abertura de arquivos são:

* "r": abre o arquivo para leitura (o arquivo deve existir).
* "w": abre o arquivo para escrita (cria o arquivo se não existir, apaga o conteúdo se existir).
* "a": abre o arquivo para anexação (cria o arquivo se não existir, adiciona ao final do arquivo se existir).
* "rb", "wb", "ab": versões binárias dos modos acima.
* "r+": abre o arquivo para leitura e escrita.
* "w+": abre o arquivo para leitura e escrita, criando o arquivo se não existir e apagando o conteúdo se existir.
* "a+": abre o arquivo para leitura e anexação.

**b)** Explicação das funções:

* fgets: lê uma linha de um arquivo ou entrada padrão (stdin) até um limite de caracteres.
* fprintf: escreve uma string formatada em um arquivo.
* fread: lê dados binários de um arquivo para um buffer.
* fwrite: escreve dados binários de um buffer para um arquivo.

**4 (0,2 pts) Qual a saída do seguinte código?**

|  |
| --- |
| *#include <stdio.h>*  int main() { int a = 5; int b = 11; float c;  scanf("%d %d", &a, &b); |

1

2

3

4

5

6

7

8

|  |  |  |
| --- | --- | --- |
|  | if(a > b || !(a > c = (float)(b | 0)) {  / a); |
|  | } else { c = (float)(a | / b); |
|  | }  printf("%.2f\n", | c); |
| } | return 0; | |

9

* Análise:
* O código realiza uma comparação entre a e b. A operação scanf espera que o usuário forneça os valores de a e b.
* Se a > b ou a <= 0 for verdadeiro, a variável c recebe b / a. Caso contrário, ela recebe a / b.
* A operação de divisão entre inteiros (b / a ou a / b) pode resultar em truncamento (sem casas decimais).
* O valor de c será impresso com duas casas decimais.

12

13

14

15

16

17

18

**5 Analise o valor das variáveis após executar o seguinte código:**

|  |
| --- |
| *#include <stdio.h>*  int main() {  int a = 1, b = 2, c = 3, d = 4, e = 5;  printf("a \* b / c = %.3f\n", (float)a \* b / c); printf("a \* b %% c + 1 = %d\n", a \* b % c + 1); printf("++a \* b - c-- = %d\n", ++a \* b - c--); printf("7 - - b \* ++d = %d\n", 7 - - b \* ++d); printf("a / b / c = %.3f\n", (float)a / b / c); printf("7 + c \* --d / e = %.3f\n", 7 + c \* --d / (float)e); printf("2 \* a %% - b + c + 1 = %d\n", 2 \* a % - b + c + 1); printf("39 / - ++e - + 29 %% c = %.3f\n", 39.0 / - ++e - + 29 % c); printf("7 - + ++a %% (3+b) = %d\n", 7 - + ++a % (3+b));  return 0;  } |

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

Analise o valor das variáveis em cada linha.

a \* b / c = 0.667

a \* b % c + 1 = 3

++a \* b - c-- = 1

7 - - b \* ++d = 17

a / b / c = 0.500

7 + c \* --d / e = 8.600

2 \* a % - b + c + 1 = 3

39 / - ++e - + 29 % c = -7.500

7 - + ++a % (3+b) = 4

**Análise:** O código realiza várias operações aritméticas e usa operadores de incremento (++), decremento (--), e o operador módulo (%), resultando em diferentes valores para cada linha de printf.

**6 Qual o valor de** w **após a execução do seguinte trecho código:**

**Análise:**

* Inicialmente, w recebe a soma de y e z, ou seja, w = 5 + 11 = 16.
* Como y não é maior que z, o valor de w permanece 16.
* O valor impresso será 16.

|  |
| --- |
| *#include <stdio.h>*  int main() { int y = 5;  int z = 11; int w;  w = y + z; if (y > z) { w = y \* z;  }  printf("%d", w); return 0;  } |

1

2

3

4

5

6

7

8

9

10

11

12

13

1. **Crie um programa em C que receba os dados de um estudante e avalie se este estudante ´e aprovado se ele obtiver nota m´mínima de** 7 **e frequência m´mínima de** 75%**.**

Adicione toda a lógica em um único if.

ARQUIVO C EM ANEXO

1. **O que ´e um vetor?**

Explique o funcionamento de um vetor e como ele ´e tratado no C.

**8. Definição e funcionamento de um vetor:**

Um **vetor** é uma estrutura de dados que armazena uma coleção de elementos do mesmo tipo em uma sequência contínua de memória. No C, um vetor é tratado como um ponteiro para o primeiro elemento, e o acesso aos seus elementos é feito por meio de índices, começando do índice 0.

Exemplo de declaração de vetor:

c

Copiar código

int v[10]; // vetor de 10 elementos inteiros

O vetor é tratado de forma eficiente, mas os índices devem estar dentro dos limites válidos (de 0 a n-1, onde n é o tamanho do vetor).