|  |
| --- |
| **PRÁTICA 01** |
|  |
| **Implemente um programa em C que faça o seguinte:**   * **Crie uma estrutura chamada "Estudante" que armazene um nome completo (até 100 caracteres) e o RU (7 dígitos).** * **Peça ao usuário para inserir o seu nome completo e RU.** * **Crie um ponteiro que aponte para essa estrutura.** * **Usando o ponteiro, verifique se o RU é par ou ímpar, imprimindo o resultado.** * **Use o ponteiro para imprimir o nome do estudante ao contrário. Por exemplo, se o nome for "João da Silva", o programa deve imprimir "avliS ad oãoJ".**   **Para demonstrar o funcionamento informe o seu nome e o seu RU, faça as capturas de tela do terminal de execução** |
| * **Apresentação do código-fonte (colorido):**   #include <stdio.h>  #include <locale.h>  #include <string.h>  #include <conio.h>  #define MAX\_NOME 99  struct Estudante {  char nome[MAX\_NOME];  int ru;  };  void ValidaImpar(struct Estudante \*ref) {  setlocale(LC\_ALL, "Portuguese");  if (ref->ru % 2 == 0) {  wprintf(L"\n->O RU é par!");  }  else {  wprintf(L"\n->O RU é impar!");  }  }  void ValidaNome(struct Estudante \*ref) {  printf("\n->Escreva seu nome: ");  fgets((ref)->nome, MAX\_NOME, stdin);  }  void ValidaRu(struct Estudante \*ref) {  printf("->Digite seu RU: ");  scanf("%d", &(ref)->ru);  }  void InverteNome(struct Estudante \*ref) {  size\_t length = strlen((ref)->nome);  if (length > 0 && (ref)->nome[length - 1] == '\n') {  (ref)->nome[length - 1] = '\0';  }  strrev((ref)->nome);  printf("\n->Nome invertido: %s\n", (ref)->nome);  printf("\nPressione qualquer tecla para finalizar..."); getch();  }  void ConstroiMenu() {  for (size\_t i = 0; i < 30; i++) {  printf("\*");  }  printf("\n");  printf("\*");  printf(" ");  printf("Bem vindo!");  printf(" ");  printf("\*");  printf("\n");  for (size\_t j = 0; j < 30; j++) {  printf("\*");  }  }  void Run() {  struct Estudante referencia;  ConstroiMenu();  ValidaNome(&referencia);  ValidaRu(&referencia);  ValidaImpar(&referencia);  InverteNome(&referencia);  }  int main() {  Run();  return 0;  } |
| * **Apresentar a captura de tela evidenciando o funcionamento:** |

|  |
| --- |
| **PRÁTICA 02** |
|  |
| **Escreva um programa em C que siga os seguintes passos:**   * **Solicite ao usuário que insira o seu RU.** * **Crie um vetor cujo tamanho seja igual ao número de dígitos no RU. Por exemplo, se o RU for "1234567", então o vetor terá 7 posições.** * **Peça ao usuário para inserir cada dígito do seu RU, armazenando cada dígito em uma posição do vetor. Por exemplo, se o RU for "1234567", o vetor deverá ter os elementos [1, 2, 3, 4, 5, 6, 7].** * **Usando ponteiros, calcule o produto de todos os números no vetor.**   **Para demonstrar o funcionamento informe o seu RU e faça as capturas de tela do terminal de execução.** |
| * **Apresentação do código-fonte (colorido):**   #include <stdio.h>  #include <stdlib.h>  #include <locale.h>  struct Registro {  char ru\_string[100];  int num\_digitos;  int \*ru\_digitos;  int produto;  };  void ConstroiMenu() {  for (size\_t i = 0; i < 30; i++) {  printf("-");  }  printf("\n");  printf("\*");  printf(" ");  printf("Bem vindo!");  printf(" ");  printf("\*");  printf("\n");  for (size\_t j = 0; j < 30; j++) {  printf("-");  }  }  void Ru(struct Registro \*ref) {  printf("\nDigite o seu RU: ");  scanf("%s", (ref)->ru\_string);  }  void QuantidadeDeDigitos(struct Registro \*ref) {  (ref)->num\_digitos = 0;  while ((ref)->ru\_string[(ref)->num\_digitos] != '\0') {  (ref)->num\_digitos++;  }  }  void Incremento(struct Registro \*ref) {  (ref)->ru\_digitos = malloc((ref)->num\_digitos \* sizeof(int));  for (int i = 0; i < (ref)->num\_digitos; i++) {  (ref)->ru\_digitos[i] = (ref)->ru\_string[i] - '0';  }  (ref)->produto = 1;  for (int j = 0; j < (ref)->num\_digitos; j++) {  (ref)->produto \*= (ref)->ru\_digitos[j];  }  }  void Resultado(struct Registro \*ref) {  setlocale(LC\_ALL, "Portuguese");  wprintf(L"\nVetor de dígitos do RU: ");  for (int i = 0; i < (ref)->num\_digitos; i++) {  printf("[%d]", (ref)->ru\_digitos[i]);  }  wprintf(L"\nProduto dos dígitos: %d\n", (ref)->produto);  free((ref)->ru\_digitos);  }  void Run() {  struct Registro referencia;  ConstroiMenu();  Ru(&referencia);  QuantidadeDeDigitos(&referencia);  Incremento(&referencia);  Resultado(&referencia);  }  int main() {  Run();  return 0;  } |
| * **Apresentar a captura de tela evidenciando o funcionamento:** |

|  |
| --- |
| **PRÁTICA 03** |
|  |
| **Escreva um programa em C que siga os seguintes passos:**   * **Solicite ao usuário que insira o comprimento do lado (L) de um pentágono regular.** * **Implemente uma função chamada calc\_penta que recebe o comprimento do lado do pentágono e dois ponteiros para float. Esta função deve calcular a área e o perímetro do pentágono e armazenar os resultados nos endereços apontados pelos ponteiros. A função deve seguir o protótipo abaixo:**   **void calc\_penta(float l, float \*area, float \*perimetro);**   * **No programa principal, depois de receber o valor do lado do usuário, chame a função calc\_penta, passando o valor do lado e os endereços de duas variáveis de float (para armazenar a área e o perímetro calculados).** * **Imprima a área e o perímetro calculados na tela.** * **O programa deve continuar pedindo ao usuário um novo valor de lado e calculando a área e o perímetro até que o usuário insira um valor negativo para o lado.**   **Para demonstrar o funcionamento utilize para o valor do lado o primeiro dígito do seu RU.**  **Nota: A área (A) e o perímetro (P) de um pentágono regular podem ser calculados com as seguintes fórmulas:** |
| * **Apresentação do código-fonte (colorido):**   #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #include <locale.h>  #define M\_PI 3.14159265358979323846  struct entrada {  float inp;  };  struct area {  float are;  };  struct perimetro {  float per;  struct area \*area;  };  void ConstroiMenu() {  printf("\n");  for (size\_t i = 0; i < 30; i++) {  printf(">");  }  printf("\n");  printf("\*");  printf(" ");  printf("Bem vindo!");  printf(" ");  printf("\*");  printf("\n");  for (size\_t j = 0; j < 30; j++) {  printf("<");  }  }  void ValorLado(struct entrada \*ref) {  printf("\n\nInsira um valor: ");  scanf("%f", &(ref)->inp);  }  void Calc\_penta(float l, struct area \*area, struct perimetro \*perimetro) {  (area)->are = (5 \* pow(l, 2)) / (4 \* tan(M\_PI / 5));  (perimetro)->per = 5 \* l;  wprintf(L"\nÁrea: %.2lf\n", (area)->are);  wprintf(L"Perímetro: %.2lf\n", (perimetro)->per);  free((perimetro)->area);  }  void Run() {    struct entrada entrada;  ConstroiMenu();  do {  setlocale(LC\_ALL, "Portuguese");  struct area area;  struct perimetro perimetro;  ValorLado(&entrada);  if ((entrada).inp != 0) {  (perimetro).area = malloc(sizeof(struct area));  Calc\_penta((entrada).inp, &area, &perimetro);  printf("\nOu digite [0] para finalizar o programa: ");  }  else {  break;  }  } while(1);  }  int main() {  Run();  return 0;  } |
| * **Apresentar a captura de tela evidenciando o funcionamento:** |

|  |
| --- |
| **PRÁTICA 04** |
|  |
| **Desenvolva, em linguagem C, um programa que calcule o valor RMS (*Root Mean Square*) de um sinal elétrico. O valor RMS é uma medida estatística da magnitude de um sinal variável. Ele pode ser especialmente útil ao lidar com quantidades variáveis no tempo, como a corrente ou a tensão em um circuito elétrico.**  **Seu programa deverá:**   * **Solicitar ao usuário para inserir o número de leituras do sinal que deseja analisar.** * **Receber do usuário essa sequência de leituras, uma por uma.** * **Utilizar uma função recursiva para calcular a soma dos quadrados dessas leituras.** * **Finalmente, calcular e imprimir o valor RMS do sinal. Lembre-se, o valor RMS é a raiz quadradada média dos quadrados das leituras.** * **Para a função recursiva, você pode usar o protótipo:**   **double somaQuadrados(double leituras[ ], int tamanho, int indice)**  **Para demonstrar o funcionamento informe a quantidade de medidas igual a quantidade de dígitos do seu RU e para cada uma das medidas informe cada dígito do seu RU.** |
| * **Apresentação do código-fonte (colorido):**   #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #include <locale.h>  struct Leitura {  int tamanho, indice;  double leituras[100];  double soma;  };  void ConstroiMenu() {  printf("\n");  for (size\_t i = 0; i < 30; i++) {  printf("~");  }  printf("\n");  printf("\*");  printf(" ");  printf("Bem vindo!");  printf(" ");  printf("\*");  printf("\n");  for (size\_t j = 0; j < 30; j++) {  printf("~");  }  }  void Init(struct Leitura \*ref) {  printf("\n\nQuantas leituras deseja analisar? ");  scanf("%d", &(ref)->tamanho);  printf("\n");  }  void Incremento(struct Leitura \*ref) {  for ((ref)->indice = 0; (ref)->indice < (ref)->tamanho; (ref)->indice++) {  printf("->Informe a leitura %d : ", (ref)->indice + 1);  scanf("%lf", &(ref)->leituras[(ref)->indice]);  }  }  double somaQuadrados(double leituras[], int tamanho, int indice) {  if (indice == tamanho - 1) {  return leituras[indice] \* leituras[indice];  } else {  return leituras[indice] \* leituras[indice] + somaQuadrados(leituras, tamanho, indice + 1);  }  }  void Finalizando(struct Leitura \*ref) {  (ref)->soma = somaQuadrados((ref)->leituras, (ref)->tamanho, 0);  wprintf(L"\nO valor RMS do sinal é: %.2lf\n", sqrt((ref)->soma / (ref)->tamanho));  }  void Run() {  setlocale(LC\_ALL, "Portuguese");  struct Leitura Leitura;  ConstroiMenu();  Init(&Leitura);  Incremento(&Leitura);  Finalizando(&Leitura);  }  int main() {  Run();  return 0;  } |
| * **Apresentar a captura de tela evidenciando o funcionamento:** |

|  |
| --- |
| **PRÁTICA 05** |
|  |
| **Seu objetivo é criar um programa em linguagem C para manipular e armazenar informações pessoais.**  **Você deve começar solicitando ao usuário que insira dados para seis diferentes registros. Cada registro deve incluir as seguintes informações: Nome, CPF, Telefone (DDD+n° do telefone) e E-mail.**  **Uma vez coletados, você deve armazenar esses dados em um arquivo no formato CSV. Para manter os dados organizados, use o caractere ponto e vírgula (;) como delimitador de campo.**  **O nome do arquivo a ser gerado deve ser o seu número de Registro Único (RU). Assim, se o seu RU for 123456, o nome do arquivo deve ser "123456.csv".** |
| * **Apresentação do código-fonte (colorido):**   #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <locale.h>  struct Init {  char ru[10];  char nomeArquivo[14];  };  struct Pessoa {  char nome[100];  char cpf[15];  char telefone[15];  char email[100];  };  void ConstroiMenu() {  printf("\n");  for (size\_t i = 0; i < 30; i++) {  printf("+");  }  printf("\n");  printf("\*");  printf(" ");  printf("Bem vindo!");  printf(" ");  printf("\*");  printf("\n");  for (size\_t j = 0; j < 30; j++) {  printf("+");  }  }  void Entrada(struct Init \*ref) {  wprintf(L"\n\nDigite o seu número de Registro Único (RU): ");  scanf("%s", ref->ru);  }  void coletarInformacoes(struct Pessoa \*registro) {  printf("\nNome: ");  scanf("%s", registro->nome);  printf("CPF: ");  scanf("%s", registro->cpf);  wprintf(L"Telefone (DDD+Número): ");  scanf("%s", registro->telefone);  printf("E-mail: ");  scanf("%s", registro->email);  }  void Coleta(struct Pessoa registros[], int numRepeticoes) {  for (int i = 0; i < numRepeticoes; i++) {  printf("\nRegistro [%d]\n", i + 1);  coletarInformacoes(&registros[i]);  }  }  void CriaCSV(struct Init \*ref) {  sprintf(ref->nomeArquivo, "%s.csv", ref->ru);  }  void SalvarEmCSV(struct Init \*nomeArquivo, struct Pessoa registros[], int numRepeticoes) {  FILE \*arquivo = fopen(nomeArquivo->nomeArquivo, "w");  if (arquivo == NULL) {  perror("Erro ao abrir o arquivo");  exit(1);  }  for (int i = 0; i < numRepeticoes; i++) {  fprintf(arquivo, "%s;%s;%s;%s\n", registros[i].nome, registros[i].cpf, registros[i].telefone, registros[i].email);  }  fclose(arquivo);  printf("\nOs dados foram salvos no arquivo %s.\n", nomeArquivo->nomeArquivo);  }  void Run() {  setlocale(LC\_ALL, "Portuguese");  struct Init Init;  struct Pessoa registros[10];  ConstroiMenu();  Entrada(&Init);  int numDigitosRU = strlen(Init.ru);  Coleta(registros, numDigitosRU);  CriaCSV(&Init);  SalvarEmCSV(&Init, registros, numDigitosRU);  }  int main() {  Run();  return 0;  } |
| * **Apresentar a captura de tela evidenciando o funcionamento:** |
| * **Apresentar a captura de tela do arquivo CSV:** |