Prova 2 - LINGUAGEM DE PROGRAMAÇÃO 2

Aluno: Daniel Sant' Anna Andrade
Matrícula: 20200036904
Aluno: Guilherme Henrique Ferraz Thomé
Matrícula; 20200002254
Aluna: Ana Clara Costa Gonçalves
Matrícula: 20200025339

1)

```
#include <stdio.h>
#define MAX_VAL 65 /* valor máximo da função */

int plano_cartesiano(int t);

int main(void) {
  char grafico[MAX_VAL + 2];
  int i, j, funval;

  /* Posiciona de 5 em 5, até 65, os valores de x */
  for (i = 0; i <= MAX_VAL; i += 5) {
    printf("%5d", i);
  }
  printf("\n");

  /* Posiciona as barras para indicar os valores de x */
  for (i = 0; i <= MAX_VAL; i += 5) {
    printf("    |");
  }
  printf("\n");

  /* Inicializa a array de x, colocando um valor em branco em cada espaço */
  for (i = 0; i <= MAX_VAL + 1; ++i) {
    grafico[i] = ' ';
  }

  /* Computa o grafico plano_cartesiano(t) para cada valor de t indo de 0 até 10 */
  for (j = 0; j <= 10; ++j) {
    funval = plano_cartesiano(j);
    grafico[funval] = '*';
    grafico[funval + 1] = '\0';
    printf("t=%2d%s\n", j, grafico);
    grafico[funval] = ' ';
    grafico[funval + 1] = ' ';
  }

  return (0);
}

int plano_cartesiano(int t) {
  return ((t * t) - (4 * t) + 5);
}
```

2a)
Para graficar uma função do primeiro grau só precisar modificar o return da função “f” (ou como foi mudado ali em cima para plano_cartesiano).

```
int plano_cartesiano(int t) {
  return ((5 * t) + 5);
}
```

2b)

```
#include <stdio.h>
#define MAX_VAL 90 /* valor máximo da função */

int plano_cartesiano(int t);

int main(void) {
  char grafico[MAX_VAL + 2];
  int i, j, funval;

  /* Posiciona de 5 em 5, até 65, os valores de x */
  for (i = 0; i <= MAX_VAL; i += 5) {
    printf("%5d", i);
  }
  printf("\n");

  /* Posiciona as barras para indicar os valores de x */
  for (i = 0; i <= MAX_VAL; i += 5) {
    printf("    |");
  }
  printf("\n");

  /* Inicializa a array de x, colocando um valor em branco em cada espaço */
  for (i = 0; i <= MAX_VAL + 1; ++i) {
    grafico[i] = ' ';
  }

  /* Computa o grafico plano_cartesiano(t) para cada valor de t indo de 0 até 10 */
  for (j = 7; j <= 13; ++j) {
    funval = plano_cartesiano(j);
    grafico[funval] = '*';
    grafico[funval + 1] = '\0';
    printf("t=%2d%s\n", j, grafico);
    grafico[funval] = ' ';
    grafico[funval + 1] = ' ';
  }

  return (0);
}
```

```
int plano_cartesiano(int t) {
  return ((t * t) - (6 * t) - 1);
}
```

Foi mudado o valor inicial de “j” para 7, pois, os valores antes desse eram número negativos. O máximo de j foi alterado para 13 e o “MAX_VAL“ foi mudado para 95, para compreender mais alguns valores.

3)

```
//struct_1.c
#include <stdio.h>

struct ponto {
  float abscissa;
  float ordenada;
};

struct ponto medio(struct ponto, struct ponto);

int main() {
  struct ponto p1;
  struct ponto p2;
  struct ponto pm;

  printf("Ingressando as coordenadas de dois pontos:\n");
  printf("\tDo primeiro ponto (x1, y1) : ");
  scanf("%f %f", &p1.abscissa, &p1.ordenada);
  fflush(stdin);
  printf("\tDo segundo ponto (x2, y2) : ");
  scanf("%f %f", &p2.abscissa, &p2.ordenada);

  pm = medio(p1, p2);

      printf("As coordenadas do ponto medio sao :(%.2f, %.2f).\n", pm.abscissa,
pm.ordenada);

  return 0;
}

struct ponto medio(struct ponto a, struct ponto b) {
  a.abscissa = (a.abscissa + b.abscissa) / 2;
  a.ordenada = (a.ordenada + b.ordenada) / 2;
  return a;
}
```

A estrutura struct ponto está armazenado duas variáveis que são a “abscissa” e a “ordenada”. Primeiramente o código pede para o usuário inserir 2 valores para o primeiro ponto, o primeiro para a abscissa (x1) e o segundo para a ordenada (y1). Depois ele pede o mesmo para o outro ponto (x2 e y2). Em seguida ele envia as duas structs para a função e soma as abscissa as de cada ponto e divide por dois e faz o mesmo com as ordenadas. E por final ele retorna a struct já com os pontos médios.

4)

```
//struct_1.c
#include <stdio.h>
#include <Math.h>

struct ponto {
  float abscissa;
  float ordenada;
};

float distancia_pontos(struct ponto, struct ponto);

int main() {
  struct ponto p1;
  struct ponto p2;
  float reta;

  printf("Ingressando as coordenadas de dois pontos:\n");
  printf("\tDo primeiro ponto (x1, y1) : ");
  scanf("%f %f", &p1.abscissa, &p1.ordenada);
  fflush(stdin);
  printf("\tDo segundo ponto (x2, y2) : ");
  scanf("%f %f", &p2.abscissa, &p2.ordenada);

  reta = distancia_pontos(p1, p2);

  printf("O segmento de reta e: %.2f\n", reta);

  return 0;
}

float distancia_pontos(struct ponto a, struct ponto b) {
  float abscissa, ordenada;
  abscissa = pow((a.abscissa - b.abscissa), 2);
  ordenada = pow((a.ordenada - b.ordenada), 2);

  return sqrt(abscissa + ordenada);
}
```

Foi trocada a função de struct para float. Ela recebe os valores dos pontos e usa a seguinte fórmula $\sqrt{(x1 - x2)^2 + (y1 - y2)^2}$ para encontrar a distância entre os dois pontos.

5)

A sequência de Fibonacci é uma sequência resultante da soma de um número com o seu anterior. A sequência se inicia com 1 e 1 e segue com 2, 3, 5, 8, ... Ela pode ser representada por $f_n = f_{n-1} + f_{n-2}$. O código abaixo imprime na tela os primeiros N termos de fibonnacci.

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

```
double fibonacci(double anterior, double atual, int contador, int posicao);

int main() {
  double anterior = 1;
  double atual = 1;
  int contador = 0;
  int posicao;

  printf("Insira ate qual valor da sequencia fibonacci voce deseja: ");
  scanf("%d", &posicao);
  printf("%.0f %.0f ", anterior, atual);

  fibonacci(anterior, atual, contador, posicao);
  return 0;
}

double fibonacci(double anterior, double atual, int contador, int posicao) {
  double temporaria;
  if (posicao > contador + 2) {
    temporaria = anterior + atual;
    anterior = atual;
    atual = temporaria;

    contador++;
    printf("%.0f ", atual);

    fibonacci(anterior, atual, contador, posicao);
  }
}
```

6)
Lista encadeada é uma lista dentro de uma lista, formando uma matriz. Ela pode ser implementada no c da seguinte forma: float array[10][10]; Assim se cria uma lista com 10 linhas e 10 colunas. Para se inserir valores dentro dessa lista é necessário criar um for dentro de um for. O primeiro for irá indicar a linha, e o segundo irá indicar a coluna.