Código: 018

- **PS_Prog_Muito_Facil_01_018**

```
#include <stdio.h>
// Funcao que converte Radianos para Graus.
double conver_rad_em_graus(double angulo){
        //Variaveis de interacão;
        int X = 180;

        // Tipo double por ter uma precisão melhor.
        double pi = 3.141592653;

        // Convesao de radianos para grau
        double graus = (angulo * X) / pi;
        return graus;
}

// Funcao Principal
int main(){

        // Variavel de entrada;
        double rad;
        printf("Digite um angulo em rad: ");
        scanf("%lf", &rad);

        // Impressão da conversão
        printf("Angulo em graus = %.1lf", conver_rad_em_graus(rad));
        return 0;
}
```

Lógica:

Este programa foi desenvolvido baseado na conversão matemática de radianos para graus, utilizando PI = 3.141592653, temos então que Graus = (angulo_radianos* 180) /PI, operação clássica da matemática básica.

- **PS_Prog_Muito_Facil_0_018**

```
#include <stdio.h>
// Função que converte horas em 0:meses/1:semanas/2:min/3:seg/4:miliseg;
void conversao(int hrs){
	printf("Conversao: [Mes] [Semana] [Min] [Seg] [Miliseg]\n");

	// Variável auxiliar e de interação "vetor";
	int conv[5], i;
	conv[0] = hrs * 30;

	// Horas que um mês deve ter;
	if(conv[0] < 720){
		conv[0] = 0;
	}
	conv[1] = hrs * 7;

	// Horas que uma semana deve ter;
	if(conv[1] < 168){
		conv[1] = 0;
	}

	// Horas em min, seg e miliseg;
	conv[2] = hrs * 60;
	conv[3] = hrs * 36000;
	conv[4] = conv[3] * 100;

	// Imprime os valores;
	printf("Conversao:");
	for(i = 0; i < 5; i++){
		printf("[%d] ", conv[i]);
	}
}
```

```
// Função Principal;
int main(){
	// Variavel de entrada;
	int horas;

	printf("Digite a quantidade de horas: ");
	scanf("%d", &horas);

	// Chamando a função conversão;
	conversao(horas);
	return 0;
}
```

Lógica:

Seguindo o princípio das conversões de tempo, utilizamos a seguinte lógica, 1 dia tem 24 horas, 1 mês tem 720 horas (30 dias), 1 semana tem 168 horas, podemos então fazer as conversões para minuto, segundo e milissegundo, representamos cada grandeza em relação a horas.

- **PS_Prog_Facil_01_018**

```
#include <stdio.h>
// Função imprime o número de repetições;
void repetir(char item[50], int n){
	// Variável de Controle;
	int i;
	printf("Repeticoes: ");
	// Impressão das repetições;
	printf("[");
	for(i = 0; i < n; i++){
		printf("\"%s\"", item);
		if(i < n-1){
			printf(",");
		}
	}
```

```
        printf("]");
    }
    // Função Principal;
    int main(){
        // Variáveis de entrada;
        char palavra[50];
        int repeticoes;
        printf("Digite um item para repetir: ");
        gets(palavra);
        printf("Digite a quantidade de repeticoes: ");
        scanf("%d", &repeticoes);
        // Chamando a função repetir;
        repetir(palavra, repeticoes);
        return 0;
    }
```

Lógica:

O programa segue uma lógica em conjunto com usuário, o usuário insere o item a se repetir e a quantidade de repetições, utilizando a ferramenta de repetição “FOR” é possível imprimir a quantidades inserida pelo usuário.

- **PS_Prog_Facil_02_018**

```
#include <stdio.h>
#include <stdlib.h>
//Função conta_uns, conta a quantidades de "1" em binário;
int conta_uns(int numero_dado){
    // Variáveis de controle e interação com a função;
    int i, resto, cont_um = 0;
    //Converte o número e em seguida conta a quantidades de números 1;
    for(i = 0; numero_dado > 0; i++){
        resto = (numero_dado % 2);
        numero_dado = numero_dado / 2;
        // Conta a quantidade de numeros 1 em binario;
        if(resto == 1){
            cont_um++;
```

```
            }
        }
        return cont_um;
    }
    // Função Principal
    int main(){
        // variável de entrada;
        int numero;
        printf("Digite o numero: ");
        scanf("%d", &numero);
        printf("\nQuantidades de numeros 1 em binario: %d \n", conta_uns(numero));
        return 0;
    }
```

Lógica:

O usuário insere um número para converter em binário, ao inserir utilizamos o método do resto, ao fazer divisões consecutivas analisamos o resto da divisão caso o resto for 1 é adicionado 1 no binário, caso seja 0 é adicionado 0, para contar quantos números 1 existe na conversão binaria, colocamos um contador dentro do caso na qual o resto seja 1.

- **PS_Prog_Medio_01_018**

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

// Funcao comprimento calcula a o comprimento do segmento de reta;
float comprimento(float x1, float y1, float x2, float y2){

    // Variaveis de interacao;
    float somax,somay,result;

    // Somas e raiz dos quadrados;
    somax = pow((x2 - x1),2);
    somay = pow((y2 - y1),2);

    // Raiz dos quadrados;
    result = sqrt((somax+somay));

    return result;
}

// Funcao Principal;
```

```
int main(){

	// Variaveis de entrada;
	float X1, X2, Y1, Y2;
	printf("Digite 2 pontos cartesianos, exemplo: 2.5,6.3\n");
	printf("\n\nDigite o ponto A: X e Y simultaneamente- ");
	scanf("%f,%f", &X1, &Y1);
	printf("\nDigite o ponto B: X e Y simultaneamente- ");
	scanf("%f,%f", &X2, &Y2);

	// Chamando função comprimento e imprimindo o resultado;
	printf("Comprimento: %.2f", comprimento(X1, Y1, X2, Y2));
	return 0;
}
```

Lógica:

O programa recebe 2 pontos do plano cartesiano 2D e retorna o comprimento do segmento de reta entre esses dois pontos, utilizado a o recurso “pow” é calculado o quadrado da diferença das ordenadas e coordenadas dos pontos, comprimento seria a raiz quadrado dessas somas desses quadrados.

- **PS_Prog_Medio_02_018**

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

//Funcao argumentos faz a selecao dos multipos de 3 e 5;
void argumentos(int num){
	// Caso seja multiplio de 3 e 5;
	if(num % 3 == 0 && num % 5 == 0){
		printf("CrossBots");
	}

	// Caso seja multiplio de 3;
	else if(num % 3 == 0){
		printf("Cross");
	}

	// Caso seja multiplio de 5;
	else if(num % 5 == 0){
		printf("Bots");
	}

	//Resto dos casos na qual não se classificam acima;
	else{
		printf("%d",num);
	}
}

// Função principal;
int main(){
```

```
    // Variável de entrada;
    int numero;

    printf("Digite um numero: ");
    scanf("%d", &numero);

    // Chamando a função argumentos;
    argumentos(numero);
    return 0;
}
```

Lógica:

O programa recebe um numero do usuário e calcula se o número é múltiplo de 3 e 5 ou se é de 3 ou de 5, e cada caso ele imprime um resultado.

- **PS_Prog_Medio_03_018**

```
function control(inputs) {


  //      Motores
  var Motor_Esq, Motor_Dir;


  // Ultrassoms
  var ultrassom_Dir = inputs["distance-right"];
  var ultrassom_Esq  = inputs["distance-left"];


  // Sensores infravermelhos
  var Infra_Esq  = inputs["front-left"];
  var Infra_Dir  = inputs["front-right"];


  // Variaveis axuiliares;
  var dist = 300;
  var vel_ataque = 40;
  var vel_procura = 10;


  // Caso os dois sensores encontrem a borda
  if(Infra_Esq > 0.25 && Infra_Dir > 0.25 ){
      Motor_Esq = -vel_ataque;
```

```
        Motor_Dir = vel_ataque;
}

// Caso sensor direito encontre a borda
if(Infra_Esq > 0.25 || Infra_Dir > 0.25 ){
       Motor_Esq = -vel_procura;
        Motor_Dir = -vel_procura;
}

// Caso sensor esquerdo encontre a borda
if(Infra_Esq > 0.25 || Infra_Dir > 0.25||Infra_Esq > 0.25 && Infra_Dir > 0.25 ){
        Motor_Esq = vel_procura ;
        Motor_Dir = vel_procura;
}

// Caso o robô esteja dentro da pista
else if(Infra_Esq < 0.25 || Infra_Dir < 0.25 || Infra_Esq < 0.25 && Infra_Dir < 0.25 ){

         // Caso ultrassom direito encontre oponente, vire a direita
          if ( ultrassom_Dir < dist && ultrassom_Esq == dist ) {
             Motor_Esq = vel_procura ;
       Motor_Dir = vel_procura;
    }

         // Caso ultrassom esquerdo encontre oponente, vire a esquerda
    else if ( ultrassom_Dir == dist && ultrassom_Esq < dist ) {
       Motor_Esq = -vel_procura;
       Motor_Dir = -vel_procura;
    }

         // Caso OS 2 ultrassons identifiquem o oponente, frente
    else if ( ultrassom_Dir < dist && ultrassom_Esq < dist ) {
```

```
            Motor_Esq = vel_ataque;
            Motor_Dir = -vel_ataque;
          }

            // Esse if procura o opononente, roda para o lado direito identificando o local
          else if ( ultrassom_Dir == dist && ultrassom_Esq == dist ) {
            Motor_Esq = vel_procura;
            Motor_Dir = vel_procura;
          }
        }

      // retorna os valores do programa para o robô, seria 'Output' saída de dados;
      return {
        leftSpeed:  Motor_Esq,
        rightSpeed: Motor_Dir,
        log: [
          { name: 'ultrassom_Dir', value: ultrassom_Dir, min: 0, max: 300 },
          { name: 'ultrassom_Esq', Value: ultrassom_Esq,  min: 0, max: 300 },
        ]
      };
    }
```

Lógica:

O programa do robô sumô apresenta componentes internos e externos no código, temos 2 sensores ultrassom, 2 sensores infravermelho e 2 motores. A seguinte lógica foi utilizada, caso os sensores infravermelhos detectam a borda, um conjunto de casos impede que o robô saia da arena, caso esses sensores detectem que o robô esteja na arena outros casos são acionados para identificar o adversário, caso os dois sensores ultrassom identifique o adversário o ataque é acionado no caso de apenas um sensor seja acionado o robô alinha para fazer o ataque.

- **PS_Prog_Dificl_03_018**

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

```
// Função que calcula valor aproximado de Pi
float pi_aproximado(){

	// variáveis de interação;
	int i = 0;
	float pi = 3.14159265,aleatorio = 0,aproximado = 0;

	// srand a cada vez que o programado rode retorne valores aleatórios;
	srand( (unsigned)time(NULL) );

	// While roda até que chege a um valor de pi aproximado;
	while(i < 1){
		// Gera os valores aleatorios de 0-1
		aleatorio = rand() % 101;
		aleatorio = aleatorio / 10000000;

		// Caso o valor seja igual ou próximo de PI;
		if(aproximado <= pi)	{
			aproximado = aproximado + aleatorio;
			}

		// Caso o valor seja maior que Pi quebra o while;
		else if(aproximado > pi){
			i = 1;
		}
	}
	return aproximado;
}

// Função Principal;
int main(){
	// Resultado amostral do valor de Pi usado com o valor aproximado;
```

```
    printf("Valor de PI usado = 3.14159265");
    printf("\nResultado calculado Aproximado de PI: %.8f...\n", pi_aproximado());
    return 0;
}
```

Lógica:

A lógica utilizada para aproximar ao valor de PI foi fazer múltiplas somas de números muitos pequenos entre 0 – 1, dado isso a somatória desses pequenos valores aproxima-se ao valor de PI, a margem de erro é muito pequena comparada ao número de PI utilizado.