Funções

Carlos Zeve

FUNÇÕES – Por Cópia

```
int teste (int x, int y){ // exemplo da função
 x = x + 2;
 y = y + 3;
 return (x + y);
}

printf ("\n %d", teste (i , j));//chamada da função
```

FUNÇÕES – Por Referência

```
int teste3 (int *x, int *y){ // “*” apontador para o endereço
 *x = *x + 2;
 *y = *y + 3;
 return (*x + *y);
}

printf ("\n %d", teste3 (&i , &j));// &passa endereço de memória
```

FUNÇÕES - Vetor

```
void le_vetor (int v[]){
        int i;
        for(i=0;i<tam;i++)
                v[i]=1;
}
```

```
int vet[tam];
le_vetor(vet);
mostra_vetor(vet);
```

```
void le_vetor1 (int *v){
        int i;
        for(i=0;i<tam;i++)
                v[i]=2;
}
```

```
int vet[tam];
le_vetor1(vet);
mostra_vetor(vet);
```

FUNÇÕES - Vetor

```
void le_vetor2 (int v[tam]){
        int i;
        for(i=0;i<tam;i++)
                v[i]=1;
}
```

```
int vet[tam];
le_vetor2(vet);
mostra_vetor(vet);
```

```
void mostra_vetor (int v[]){
 int i;
 for(i=0;i<tam;i++)
      printf("\nV[%i]=%i",i+1,v[i]);
}
```

FUNÇÕES - Matriz

```
void le_matriz (int m[lin][col]){
        int i,j;
        for(i=0;i<lin;i++)
                for(j=0;j<col;j++)
                        m[i][j]=1;
}--------------------------------------
int mat[lin][col];

le_matriz(mat);
mostra_matriz(mat);
```

```
void le_matriz1 (int m[][col]){
        int i,j;
        for(i=0;i<lin;i++)
                for(j=0;j<col;j++)
                        m[i][j]=2;
}--------------------------------------
int mat[lin][col];

le_matriz(mat);
mostra_matriz(mat);
```

FUNÇÕES - Matriz

```
void mostra_matriz (int m[lin][col]){
        int i,j;
        for(i=0;i<lin;i++){
          printf("\n");
          for (j=0;j<col;j++)
                printf(" %i ", m[i][j]);
        }
}---------------------------------------
int mat[lin][col];

mostra_matriz(mat);
```

FUNÇÕES – Registro por Cópia

```
#define tam 3
struct reg{
        char nome[30];
        int idade;
};
void mostra_registro (struct reg a){
        printf("\nNome: %s",a.nome);
        printf("\nIdade: %i",a.idade);
}
void le_registro(struct reg a){
        strcpy(a.nome,"ana");
        a.idade = 40;
        mostra_registro(a);
}
```

```
main(){
        struct reg aluno;

        strcpy(aluno.nome,"carlos");
        aluno.idade =50;
        le_registro(aluno);
        mostra_registro(aluno);


        getch();
}
```

FUNÇÕES – Registro por Referência

```
#define tam 3

struct reg{
        char nome[30];
        int idade;
};

void mostra_registro (struct reg a){
        printf("\nNome: %s",a.nome);
        printf("\nIdade: %i",a.idade);
}

void le_registro(struct reg *a){
        strcpy(a->nome,"ana");
        a->idade = 40;
        mostra_registro(*a);
}
```

```
main(){
        struct reg aluno;

        strcpy(aluno.nome,"carlos");
        aluno.idade =50;
        le_registro(&aluno);
        mostra_registro(aluno);


        getch();
}
```

FUNÇÕES – Vetor Registro

```
#define tam 3
struct reg{
        char nome[30];
        int idade;
};
void mostra_vregistro (struct reg *a){
        int i;
        for(i=0;i<tam;i++){
                printf("\nNome: %s",a[i].nome);
                printf("\nIdade: %i",a[i].idade);
}}
void le_vregistro(struct reg *a){
        int i;
        for(i=0;i<tam;i++){
                fflush(stdin);
                gets(a[i].nome);
                fflush(stdin);
                scanf("%i",&a[i].idade);
}}
```

```
main(){
        struct reg aluno[tam];

        le_vregistro(aluno);
        mostra_vregistro(aluno);


        getch();
}
```

FUNÇÕES – Retornar um registro

```
#define tam 3
struct reg{
        char nome[30];
        int idade;
};
void mostra_registro (struct reg a){
        printf("\nNome: %s",a.nome);
        printf("\nIdade: %i",a.idade);
}
struct reg le_registro(char *n, int i){
        struct reg a;
        strcpy(a.nome,n);
        a.idade = i;
   return a;
}
```

```
main(){
 struct reg aluno;

  aluno = le_registro("carlos",50);
  mostra_registro(aluno);

  getch();
}
```