MOSTRANDO COISA NA TELA

\n = Nova linha

\t = Tabulação

\b = Backspace

\r = Retorno

\\ = Barra

\" = Aspas

\? = Interrogação

\a ou \7 = Beep

%% = Porcentagem

TIPOS PRIMITIVOS

CHAR = Signed, Unsigned

INT (Ocupa 4 bytes) = Short, Long, Signed, Unsigned

Short (Ocupa 2 bytes) Signed (Se precisar do sinal)

Long (Ocupa 8 bytes) Unsigned (Ocupa-> desocupa o byte do sinal) Sempre positivo

FLOAT (Ocupa 4 espaços na memória)

Real ou ponto flutuante -> ex: (4.5, 0.075, -1.0893)

DOUBLE (Real, ocupa 8 espaços na memória)

VOID (Não dá retorno-> Vazio)

PARAMETROS DE FORMATAÇÃO (Saída ou Entrada)

%d = Int %e = Notação (Representação (e) da matemática, com notação exponencial)

%f = Float ou Double %hd = Short int %u = Unsigned

%c = Char , Caractere %1d = Long int %1u = Unsigned Long

%s = Cadeia (Palavras ou frases) %hu = Unsigned short

VARIÁVEIS REGRAS (Regras da Linguagem C)

Começa com uma letra

Maiúscula e minúsculas fazem diferença.

Obs: De preferência declarar com minúscula

Só usa letras e números (Sublinhado só Anderlaine)

Nada de acentos

Não pode conter espaços

Não pode ter símbolos (só o sublinhado-> Anderlaine)

Não pode ser uma palavra reservada

EX: #include <stdio.h>, Void main()

FORMAS DE DECLARAR VARIÁVEIS

int idade=40; char nome= “Albert”;

float peso=73.5; char *nome=”Albert”;

char sexo=’M’;

CONSTANTES

#Define VS **Const**

- **#define** é uma diretiva de pré-processamento
- **Const** é uma palavra reservada do C
- **Com # define**, o valor é substituído antes da compilação
- **Com Const**, a constante existe na memória
- **Com Const** é possível indicar o tipo primitivo da constante

EX: Const

```
#include <stdio.h>
Void main() {
Const int total = 8;
Printf(“%d”,toral);
}
```

EX: #define

```
#include <stdio.h>
#define total 8
void main(){
 printf(“%d”,total);
 }
```

GERAR NÚMEROS ALEATORIOS

```
#include <stdio.h
#include <stdlib.h>     Tem que importar essas 2 bibliotecas
#include <time.h>

Void main() {

Srand(time (NULL)); ( NULL) Tem que ser maiusculo

Int  n = rand() % 10 + 1; Começa a partir de ( 1 )

Printf("Eu gerei o número %d",n);
}

Obs: int n = rand() % 10;  Começa a partir de (zero)
     Int n = rand(); Gera qualquer número aleatório
```

CARACTERE,RECOMENDO USAR:(getchar();) LENDO STRING,RECOMENDO: (gets (variável);)

```
#Include <stdio.h>

void main() {

char r;
char s;

printf("Digite so uma letra:");
fflush(stdin);
r = getchar();

printf("Digite outra letra:");
fflush(stdin);
s = getchar();

printf(Letra \"%c\" é \"%c\"",r,s);
```

```
#include <stdio.h>

void main() {

char nome[30];
char ender[40];

pintf("Digite seu nome:");
fflush(stdin);
gets(nome);

printf("Seu endereço:");
fflush(stdin);
gets(ender);

printf("Nome: \"%s\" mora \"%s\"",nome,ender);
```

OPERADORES RELACIONAIS

COMPARAÇÕES:

[== Igual]	[!= Diferente]	[> Maior]	[< Menor]	[>= Maior igual]	[<= Menor igual]

Obs: Os valores relacionais resultam sempre um valor lógico! **[True = 1]** **[False = 0]**

OPERAÇÕES COM STRING

BIBLIOTECA: #include <string.h>

STRCPY(); Consegue fazer atribuições de valores dentro da string

STRLEN();Tamanho da string. Ex: Albert = 6 caractere

STRCMP(); Faz a comparação da string, se elas são iguais da zero, se uma é maior que a outra vai dar número positivo ou negativo.

STRCAT(); Faz a concatenação

STRUPR(); Joga para maiúscula

STRLWR(); Joga para minúscula

STRREV(); Função que inverte uma string ao contrario.

OPERADORES LÓGICOS

COMPARAÇÕES:

&& Conjução (e): So é verdade quando os dois é verdade.

|| Disjunção (ou): So é falso quando os dois é falso.

! Negação (Não)

Ordem de precedência: **1°** !(Negação), **2°** && (Conjunção), **3°** || (Disjunção)
Obs: Quando estiver sem parênteses

OPERADORES DE ATRIBUIÇÃO

x = x + 3; ou é o mesmo que **x += 3;**
a = a * 4; ou é o mesmo que **a *= 4;** **Obs: Não usar espaço entre os sinais!**
b = b / 2; ou é o mesmo que **b /= 2;**
c = c % 5; ou é o mesmo que **c %= 5;**

OPERADORES DE INCREMENTO E DECREMENTO

#include <stdio.h> *SOMA: **a++;*** *SUBTRAÇÃO: **b--;***

Void main(){
Int x = 3 + a++; **Obs: Se estivesse incrementado x = 3 + ++a; o resultado de b seria outro.**

Printf("Essa linha não faz o incremento por conta do sinal %d",a++);
Printf("O incremento de (x) so mostra na linha seguinte %d",x);

CAPTURANDO A DATA E HORA DO SISTEMA

```
#include <stdio.h>
#include <time.h>   Obs: Tem que fazer a carga da biblioteca (<time.h>)

Void main() {

time_t   t;      time_ t: (tempo primitivo)        t: (variável)

time(&t);     time (Função interna)    (&t): A função (time) esta recebendo o endereço de (t) para
pegar data e hora do sistema.

Struct  tm  *data;        struct: Comando de estrutura   tm:  Nome da estrutura (localtime)
                          *data: Ponteiro para (tm).

Data = localtime (&t);   localtime: pega data, hora, mês, vários parâmetros para poder mostrar
na tela

Int  d = data -> tm_mday;      d: variável do tipo int.     data: Ponteiro apontando para o dia atual
```

ALGUMAS VARIAVEIS INTERNAS DENTRO DO (struct)

tm_mday: Dia do mês [1 – 31]

tm_mon: Mês [0 – 11] (0) é Janeiro, tem que somar + 1

tm_year: Ano [A partir de 1900] tem que somar 1900 para chegar ao ano atual.

tm_wday: Dia da semana [0 – 6] (0) é domingo, (6) é sexta.

tm_yday: Dia do ano [0 – 365] (0) é 1° de Janeiro

tm_hour: Hora [0 – 23] (0) corresponde a meia noite.

tm_min: Minutos [0 – 59]

tm_sec: Segundos [0 – 59]