

# Introdução

Linguagem C

# Linguagem C

- • Variáveis e Tipos de dados (`char`, `int`, `float`, `double`...)
- Condicionais (`if..else`)
- Repetição (`for`; `while`)
- Alocação estática
- Alocação dinâmica
- Ponteiros
- Estruturas (`struct`)
- Funções e Procedimentos (modularização)

# Variáveis

- "Posição de memória" (endereço)
  - declaração do **tipo de dado**
  - identificador (nome)
  - valor (conteúdo da variável)

```
char c = 'A';
```

```
int i = 10;
```

```
double d = 1.11;
```

```
float umValorRealDe32bits = 2.22f;
```

(tipo)	(nome)	(valor)
--------	--------	---------

# Variáveis

- "Posição de memória" (endereço)
  - declaração do **tipo de dado**
  - identificador (nome)
  - valor (conteúdo da variável)

```
char c = 'A';  
char c2;  
int i = 10;  
int j = i;  
double d = 1.11;
```

# (Alguns) Tipos de dados

- booleano: `bool` (\* C++)
- caractere: `char`
- inteiro: `int`, `long`, `long long`
- real: `float`, `double`
- Pergunta: Quantos bits?

# Quantos bits?

```
bool    : 1 bit?      (true / false)
char    : 8 bits      (1 byte: 0..255)
int     : 16 bits?    (2 bytes: 0..65535)
long    : 32 bits?
long long : 64 bits
float   : 32 bits
double  : 64 bits
```

# Quantos bits?

- Windows 64 bits (gcc 7.4.0)

```
sizeof(bool)   = 1 byte           (8 bits)
sizeof(char)   = 1 byte
sizeof(int)    = 4 bytes          (32 bits)
sizeof(long)   = 8 bytes
sizeof(long long) = 8 bytes      (64 bits)
sizeof(float)   = 4 bytes
sizeof(double) = 8 bytes
```

# eco.c

- Exemplo 1

```
#include <stdio.h>
int main() {
    int A;
    scanf ("%d", &A) ;
    printf ("%d\n", A) ;
    return 0;
}
```

— Entrada: 9223372036854775807

— Saída: 4294967295 (Erro??)



# eco.c

- Exemplo 1

```
#include <stdio.h>
int main() {
    int A;
    scanf("%d", &A);
    printf("%d\n", A);
    return 0;
}
```

(32 bits:  $2^{32} = 4.294.967.296$ )

— Entrada: 9.223.372.036.854.775.807

— Saída: 4.294.967.295 (Erro??)

# eco.c

- Exemplo 1

```
#include <stdio.h>
int main() {
    int A;
    scanf("%d", &A);
    printf("%d\n", A);
    return 0;
}
```

(32 bits: -2.147.483.647, +2.147.483.647)

— Entrada: -9223372036854775808

— Saída: 0 (Erro??)

# eco.c

- Pergunta: long ou long long?

```
#include <stdio.h>
int main() {
    long A;
    scanf("%ld", &A);
    printf("%ld\n", A);
    return 0;
}
```

# eco.c

- Pergunta: long ou long long?

```
#include <stdio.h>
int main() {
    long long A;
    scanf("%lld", &A);
    printf("%lld\n", A);
    return 0;
}
```

# Entrada e Saída

- Um programa possui **entrada** e **saída** de dados
- Exemplo de **entrada**
  - leitura via teclado (`scanf`)
- Exemplo de **saída**
  - impressão na tela (`printf`)

# eco.c

- Exemplo 1A

```
#include <stdio.h>
int main() {
    long A;
    scanf("%ld", &A);    // Entrada
    printf("%ld\n", A);  // Saída
    return 0;
}
```

# hello.c

```
#include <stdio.h>

int main() {
    printf("Hello!\n"); // Saída
    return 0;
}
```

# Leitura via teclado

```
#include <stdio.h>

int main() {
    int i;
    long l;
    scanf ("%d", &i);    // Entrada
    scanf ("%ld", &l);  // Entrada
    return 0;
}
```



# Entrada e Saída

```
#include <stdio.h>

int main() {
    int i;
    long l;
    scanf ("%d", &i);    // Entrada
    scanf ("%ld", &l);   // Entrada
    printf ("i = %d\n", i);    // Saída (32 bits)
    printf ("l = %ld\n", l);   // Saída (64 bits)
    return 0;
}
```

— Entrada: 100000000000000  
          100000000000000

— Saída: i = ?  
         l = ?

# Entrada e Saída

```
#include <stdio.h>

int main() {
    int i;
    long l;
    scanf ("%d", &i);
    scanf ("%ld", &l);
    printf ("i = %d\n", i);    // 32 bits: overflow!
    printf ("l = %ld\n", l);  // 64 bits
    return 0;
}
```

— Entrada: 100000000000000  
100000000000000

— Saída: i = -727379968  
l = 100000000000000

# Mais Leitura via teclado

```
#include <stdio.h>

int main() {
    char c;
    float f;
    double d;
    scanf ("%c", &c);    // 8 bits
    scanf ("%f", &f);    // 32 bits
    scanf ("%lf", &d);   // 64 bits
    return 0;
}
```

# Tipo Char

- Exemplo 2

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

```
int main() {
```

```
    char letra1 = 'a', letra2 = 97;
```

```
    printf ("%c %c\n", letra1, letra2);
```

```
    return 0;
```

```
}
```

- 

– Saída:     a   ?

# Tipo Char

- Exemplo 2

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

```
int main() {
```

```
    char letra1 = 'a', letra2 = 97;
```

```
    printf ("%c %c\n", letra1, letra2);
```

```
    return 0;
```

```
}
```

- 

– Saída:     a a

# Tabela ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32; <b>Space</b>		64	40	100	&#64; <b>@</b>		96	60	140	&#96; <b>`</b>	
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33; <b>!</b>		65	41	101	&#65; <b>A</b>		97	61	141	&#97; <b>a</b>	
2	2	002	<b>STX</b> (start of text)	34	22	042			66	42	102	&#66; <b>B</b>		98	62	142	&#98; <b>b</b>	
3	3	003	<b>ETX</b> (end of text)	35	23	043			67	43	103	&#67; <b>C</b>		99	63	143	&#99; <b>c</b>	
4	4	004	<b>EOT</b> (end of transmission)	36	24	044			68	44	104	&#68; <b>D</b>		100	64	144	&#100; <b>d</b>	
5	5	005	<b>ENQ</b> (enquiry)	37	25	045			69	45	105	&#69; <b>E</b>						
6	6	006	<b>ACK</b> (acknowledge)	38	26	046			70	46	106	&#70; <b>F</b>						
7	7	007	<b>BEL</b> (bell)	39	27	047			71	47	107	&#71; <b>G</b>						
8	8	010	<b>BS</b> (backspace)	40	28	048			72	48	108	&#72; <b>H</b>						
9	9	011	<b>TAB</b> (horizontal tab)	41	29	049			73	49	109	&#73; <b>I</b>						
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	050			74	4A	110	&#74; <b>J</b>						
11	B	013	<b>VT</b> (vertical tab)	43	2B	051			75	4B	111	&#75; <b>K</b>						
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	052			76	4C	112	&#76; <b>L</b>						
13	D	015	<b>CR</b> (carriage return)	45	2D	053			77	4D	113	&#77; <b>M</b>						
14	E	016	<b>SO</b> (shift out)	46	2E	054			78	4E	114	&#78; <b>N</b>						
15	F	017	<b>SI</b> (shift in)	47	2F	055			79	4F	115	&#79; <b>O</b>						
16	10	020	<b>DLE</b> (data link escape)	48	30	060			80	50	120	&#80; <b>P</b>						
17	11	021	<b>DC1</b> (device control 1)	49	31	061			81	51	121	&#81; <b>Q</b>						
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50; <b>2</b>		82	52	122	&#82; <b>R</b>		114	72	162	&#114; <b>r</b>	
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51; <b>3</b>		83	53	123	&#83; <b>S</b>		115	73	163	&#115; <b>s</b>	
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52; <b>4</b>		84	54	124	&#84; <b>T</b>		116	74	164	&#116; <b>t</b>	
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53; <b>5</b>		85	55	125	&#85; <b>U</b>		117	75	165	&#117; <b>u</b>	
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54; <b>6</b>		86	56	126	&#86; <b>V</b>		118	76	166	&#118; <b>v</b>	
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55; <b>7</b>		87	57	127	&#87; <b>W</b>		119	77	167	&#119; <b>w</b>	
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56; <b>8</b>		88	58	130	&#88; <b>X</b>		120	78	170	&#120; <b>x</b>	
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57; <b>9</b>		89	59	131	&#89; <b>Y</b>		121	79	171	&#121; <b>y</b>	
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58; <b>:</b>		90	5A	132	&#90; <b>Z</b>		122	7A	172	&#122; <b>z</b>	
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59; <b>:</b>		91	5B	133	&#91; <b>[</b>		123	7B	173	&#123; <b>{</b>	
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60; <b>&lt;</b>		92	5C	134	&#92; <b>\</b>		124	7C	174	&#124; <b> </b>	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61; <b>=</b>		93	5D	135	&#93; <b>]</b>		125	7D	175	&#125; <b>}</b>	
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62; <b>&gt;</b>		94	5E	136	&#94; <b>^</b>		126	7E	176	&#126; <b>~</b>	
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63; <b>?</b>		95	5F	137	&#95; <b>_</b>		127	7F	177	&#127; <b>DEL</b>	

Source: [www.LookupTables.com](http://www.LookupTables.com)

# (Alguns) Tipos de dados

- • booleano: `bool` (`#include <stdbool.h>` // C++)

## Tipos Lógicos

- Em C, não existe tipo lógico.
- Usa-se um inteiro: 0 = falso, 1 = verdadeiro.
  - Na realidade qualquer coisa diferente de 0 é verdadeiro.

# Sintaxe

- Lendo valores via teclado

```
scanf("%EspecificadorDeFormato", &NomeDaVariavel);
```

- Exibindo os valores

```
printf("%EspecificadorDeFormato", NomeDaVariavel);
```



# Formatos

Tipo	Especificador de Formato
char	%c
int	%d
long	%ld
long long	%lld
float	%f
double	%lf

# Linguagem C

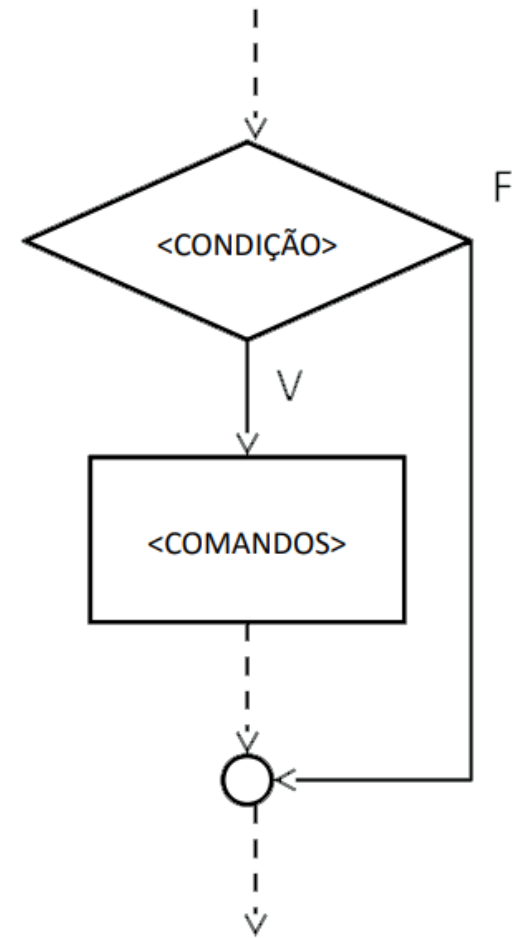
- Variáveis e Tipos de dados (`char`, `int`, `float`, `double`...)
- • Condicionais (`if..else`)
- Repetição (`for`; `while`)
- Alocação estática
- Alocação dinâmica
- Ponteiros
- Estruturas (`struct`)
- Funções e Procedimentos (modularização)

# Condicionais

- if
- if ... else
- if ... else if ... else if ... else ...

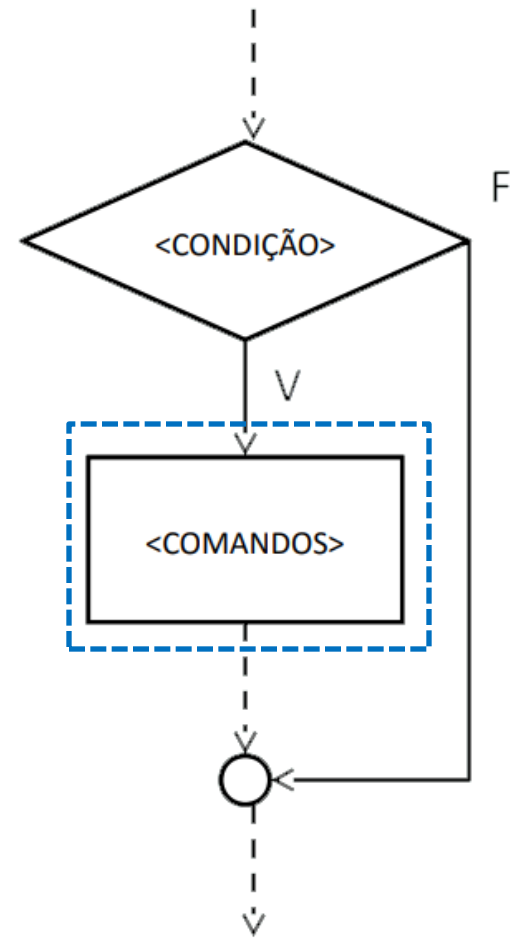
# Condicionais

```
...  
if ( <CONDICAO> ) {  
    <COMANDOS>  
    ...  
}  
...
```



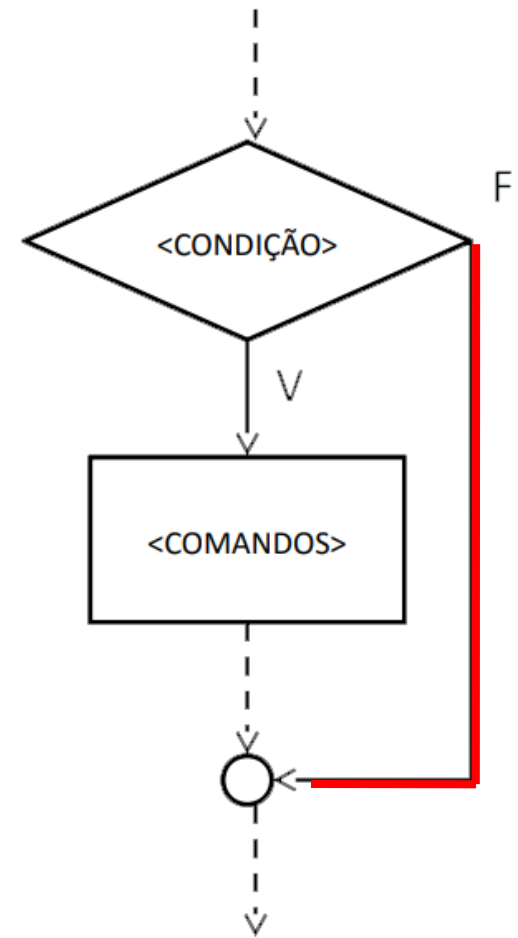
# Condicionais

```
...  
if ( <CONDICAO> ) {  
    <COMANDOS>  
    ...  
}  
...
```



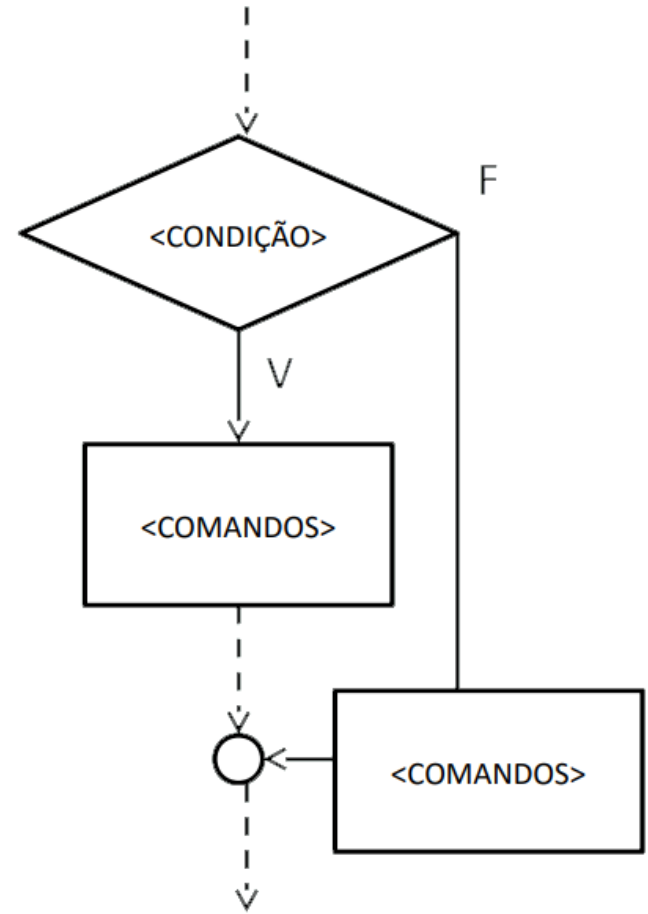
# Condicionais

```
...  
if ( <CONDICAO> ) {  
    <COMANDOS>  
    ...  
}  
...
```



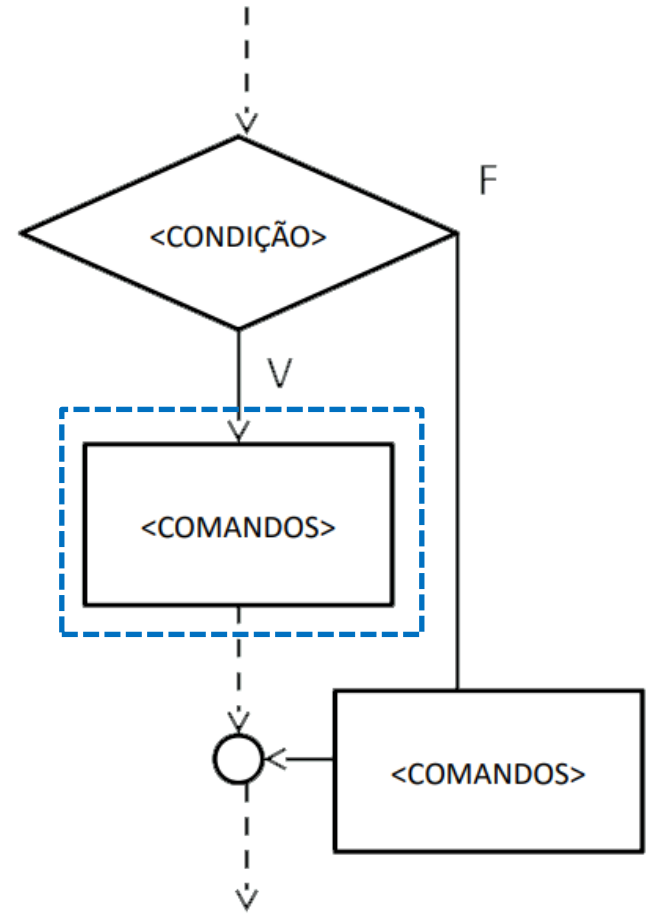
# Condicionais

```
...  
if ( <CONDICAO> ) {  
    <COMANDOS>  
    ...  
}  
else {  
    <COMANDOS>  
    ...  
}  
...
```



# Condicionais

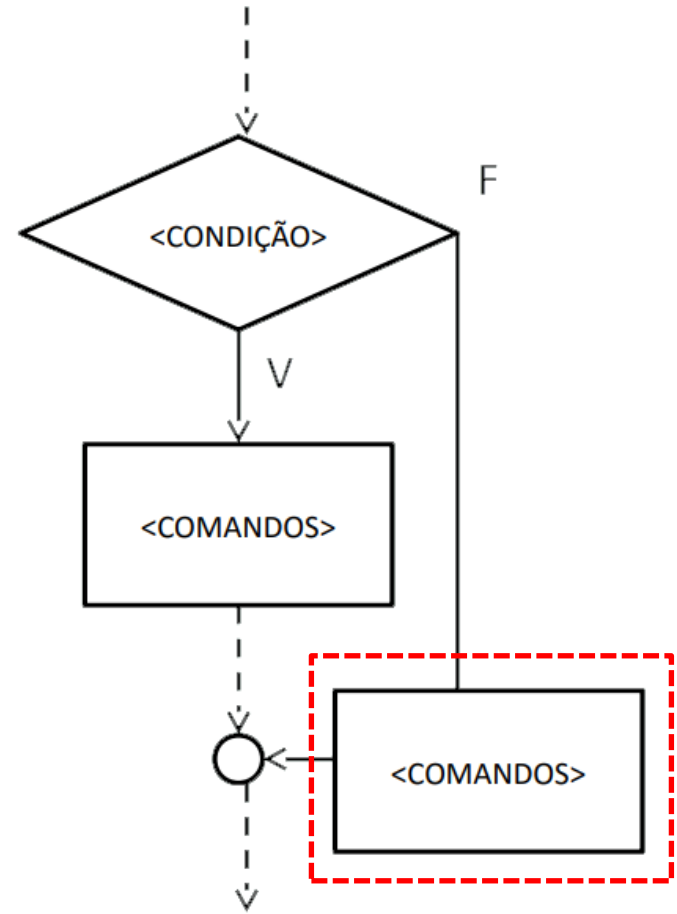
```
...  
if ( <CONDICAO> ) {  
    <COMANDOS>  
    ...  
}  
else {  
    <COMANDOS>  
    ...  
}  
...
```





# Condicionais

```
...  
if ( <CONDICAO> ) {  
    <COMANDOS>  
    ...  
}  
else {  
    <COMANDOS>  
    ...  
}  
...
```



# Condicionais

- if ... else

- • if ... else if ... else if ... else ...

```
...
if ( <CONDICAO> ) {

    <COMANDOS>
    ...

}
else if ( <CONDICAO> ) {

    <COMANDOS>
    ...

}
else
...

```

# imc.c

```
float a, p, imc;
scanf("%f %f", &a, &p);
imc = p/(a*a);
if (imc < 16) {
    printf("Magreza grave\n");
}
else if (imc < 17) {
    printf("Magreza moderada\n");
}
else if (imc < 18.5) {
    printf("Magreza leve\n");
}
else if (imc < 25) {
    printf("Saudavel\n");
}
else if (imc < 30) {
    printf("Sobrepeso\n");
}
...
```

# Linguagem C

- Variáveis e Tipos de dados (`char`, `int`, `float`, `double`...)
- Condicionais (`if..else`)
- • Repetição (`for`; `while`)
- Alocação estática
- Alocação dinâmica
- Ponteiros
- Estruturas (`struct`)
- Funções e Procedimentos (modularização)

# Repetição

- for

- • while

- do ... while

# Qual a SAÍDA?

```
#include <stdio.h>

int main () {
    int i = 0;
    while (i < 10) {
        printf ("%d\n", i);
        i++;
    }
    return 0;
}
```

(Saída: ?)

# Qual a SAÍDA?

```
#include <stdio.h>

int main () {
    int i = 0;           // inicializacao
    while (i < 10) {     // condicao
        printf ("%d\n", i);
        i++;             // incremento
    }
    return 0;
}
```

Saída: 0

1

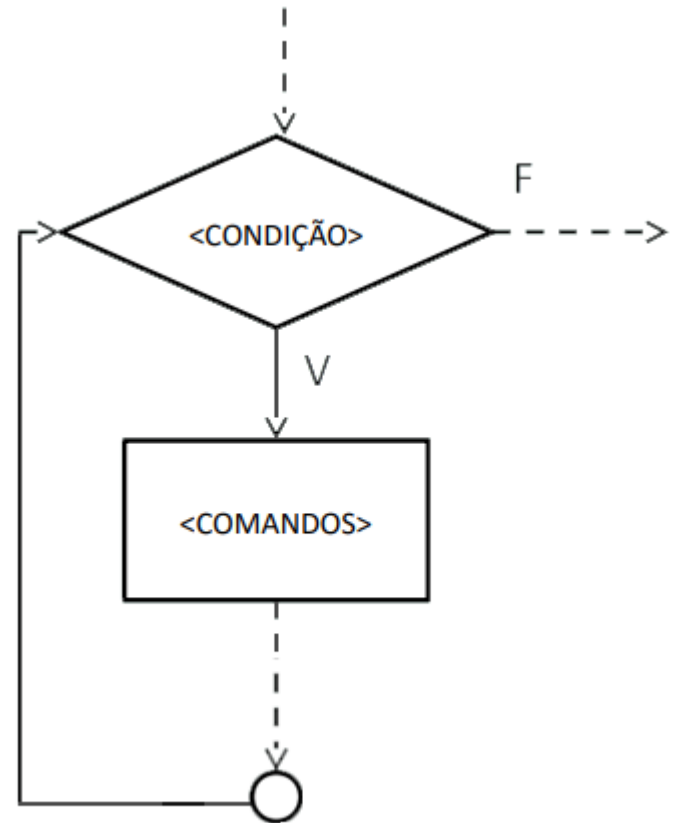
2

...

9


# While

```
...  
while ( <CONDICAO> )  
{  
    <COMANDOS>  
    ...  
}  
...
```





# Repetição

- for
  - while
  - do ... while
- 
- (Similaridades?)

# While x For

```
#include <stdio.h>

int main () {
    int i = 0;           // inicializacao
    while (i < 10) { // condicao
        printf ("%d\n", i);
        i++;             // incremento
    }
    return 0;
}
```

Saída: 0

1

2

...

9

```
#include <stdio.h>

int main () {
    int i;
    for (i = 0; i < 10; i++) {
        printf ("%d\n", i);
    }
    return 0;
}
```

for (inicialização; condição; incremento)

Saída: 0


1

2

...

9

# Repetição

- for
  - while
  - do ... while
- 
- (Existe diferença?)

# Qual a SAÍDA?

```
#include <stdio.h>

int main () {
    int i = 10;
    while (i < 10) {
        printf ("%d\n", i);
        i++;
    }
    return 0;
}
```

(Saída: ?)

```
#include <stdio.h>

int main () {
    int i = 10;
    do {
        printf ("%d\n", i);
        i++;
    } while (i < 10);
    return 0;
}
```

(Saída: ?)

# Qual a SAÍDA?

```
#include <stdio.h>

int main () {
    int i = 10;
    while (i < 10) {
        printf ("%d\n", i);
        i++;
    }
    return 0;
}
```

(Saída: )

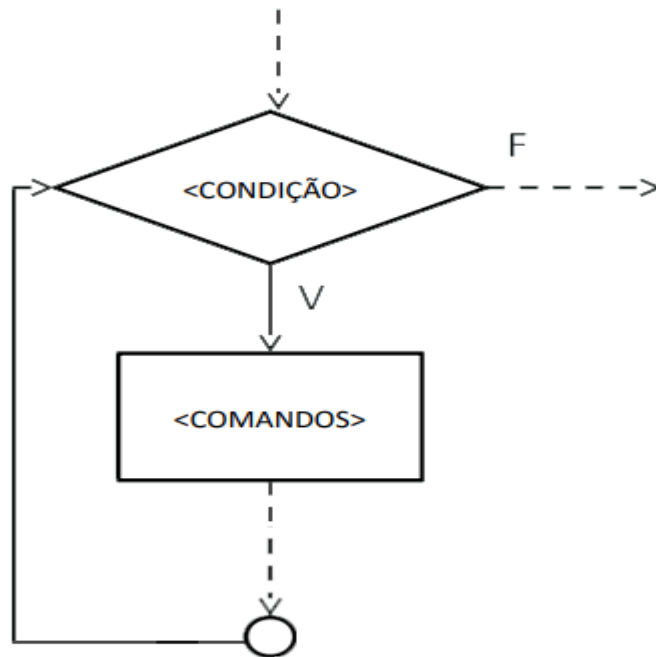
```
#include <stdio.h>

int main () {
    int i = 10;
    do {
        printf ("%d\n", i);
        i++;
    } while (i < 10);
    return 0;
}
```

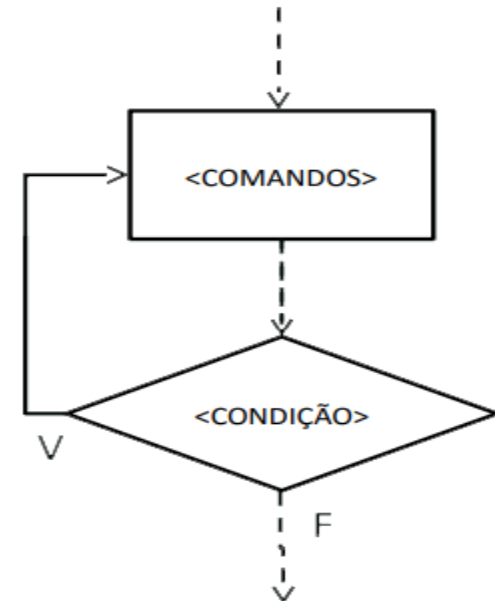
(Saída: 10)

# While x Do...While

```
...  
while ( <CONDICAO> ) {  
  
    <COMANDOS>  
    ...  
  
}  
...
```



```
...  
do {  
  
    <COMANDOS>  
    ...  
  
} while ( <CONDICAO> );  
...
```



# Linguagem C

- Variáveis e Tipos de dados (`char`, `int`, `float`, `double`...)
- Condicionais (`if..else`)
- Repetição (`for`; `while`)
- • Alocação estática
- Alocação dinâmica
- Ponteiros
- Estruturas (`struct`)
- Funções e Procedimentos (modularização)

(Vetor, Matriz, ...)

# Alocação Estática x Dinâmica

```
#include <stdio.h>

int main () { // constante
    int i, vetor[10];
    for (i = 0; i < 10; i++) {
        vetor[i] = 2*i;
    }
    return 0;
}
```

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

int main () { // variável n
    int i, n, *vetor;
    scanf ("%d", &n);

    vetor = malloc(
        n*sizeof(int));

    for (i = 0; i < n; i++) {
        vetor[i] = 2*i;
    }
    free (vetor);
    return 0;
}
```

vetor = {0, 2, 4, 6, 8, ..., 18}  
índices: 0 1 2 3 4 9



# Alocação Estática

```
#include <stdio.h>

int main () {
    int i, j, matriz[2][5];
    for (i = 0; i < 2; i++) {
        for (j = 0; j < 5; j++) {
            matriz[i][j] = j + (i*5);
        }
    }
    return 0;
}
```

Matriz = { {0, 1, 2, 3, 4};  
          {5, 6, 7, 8, 9} }

# Alocação Dinâmica

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

int main () {
    int i, j, **matriz;

    matriz = malloc(2*sizeof(int *));
    for (i = 0; i < 2; i++) {
        matriz[i] = malloc(5*sizeof(int));
    }

    ...
    // free()?

    return 0;
}
```

Matriz = { {0, 1, 2, 3, 4};  
          {5, 6, 7, 8, 9} }

# Alocação Dinâmica

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    int i, j, **matriz;

    matriz = malloc(2*sizeof(int *));
    for (i = 0; i < 2; i++) {
        matriz[i] = malloc(5*sizeof(int));
    }
    ...

    // free()?
    for (i = 0; i < 2; i++) {
        free (matriz[i]);
    }
    free (matriz);
    return 0;
}
```

# Feio!

```
#include <stdio.h>

int main () {
    int i, n;    // variável n
    scanf ("%d", &n);

    int vetor[n];

    ...
    return 0;
}
```

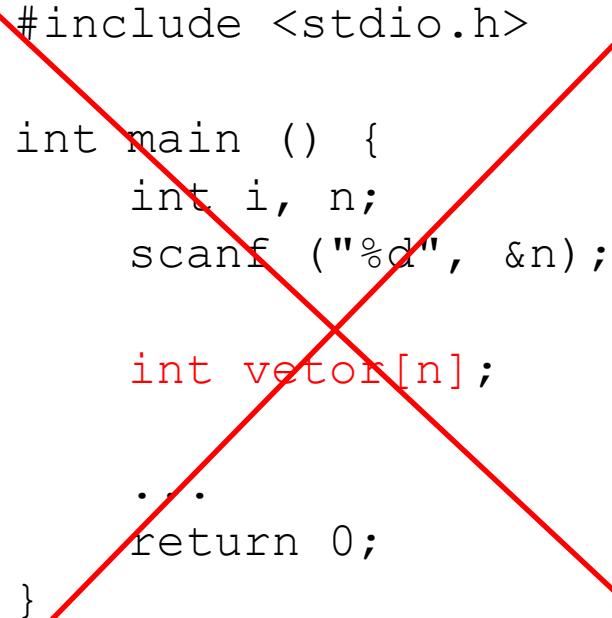
# Feio x Bonito

```
#include <stdio.h>

int main () {
    int i, n;
    scanf ("%d", &n);

    int vetor[n];

    ...
    return 0;
}
```



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

int main () {
    int i, n; // variável n
    scanf ("%d", &n);

    int *vetor;
    vetor = malloc (n * sizeof (int));

    ...
    return 0;
}
```

# Linguagem C

- Variáveis e Tipos de dados (`char`, `int`, `float`, `double`...)
- Condicionais (`if..else`)
- Repetição (`for`; `while`)
- Alocação estática
- Alocação dinâmica
- Ponteiros
- • Estruturas (`struct`)
- Funções e Procedimentos (modularização)

# Structs

```
#include <stdio.h>

struct Ponto {
    int x;
    int y;
};

int main () {
    struct Ponto p;
    scanf ("%d %d", &p.x, &p.y);

    printf ("%d %d\n", p.x*2, p.y*2);
    return 0;
}
```

Entrada: 10 11

Saída: 20 22

# Typedef

```
#include <stdio.h>

typedef struct Ponto_s {
    int x;
    int y;
} Ponto;

int main () {
    Ponto p;
    scanf ("%d %d", &p.x, &p.y);

    printf ("%d %d\n", p.x*2, p.y*2);
    return 0;
}
```

Entrada: 10 11

Saída: 20 22



# Typedef

```
#include <stdio.h>

typedef char Caractere;
typedef long Inteiro;
typedef double Real;

int main () {
    Caractere c;
    Inteiro i;
    Real r;
    scanf ("%c %ld %lf", &c, &i, &r);

    printf ("%c %ld %lf\n", c, i, r);
    return 0;
}
```

Entrada: a 1 1.1

Saída: [a 1 1.100000]

# Ponteiro (1)

```
#include <stdio.h>

typedef struct Ponto_s {
    int x;
    int y;
} Ponto;

int main () {

    Ponto *p;
    scanf ("%d %d", &p->x, &p->y);

    printf ("%d %d\n", p->x*2, p->y*2);
    return 0;
}
```

(Onde está o erro?)

# Ponteiro (1)

```
#include <stdio.h>

typedef struct Ponto_s {
    int x;
    int y;
} Ponto;

int main () {

    Ponto *p;
    scanf ("%d %d", &p->x, &p->y);

    printf ("%d %d\n", p->x*2, p->y*2);
    return 0;
}
```

(Onde está o erro? **\*p** é apenas um ponteiro, precisamos alocar memória!)

# Ponteiro (2)

```
#include <stdio.h>

typedef struct Ponto_s {
    int x;
    int y;
} Ponto;

int main () {
    Ponto tmp;
    Ponto *p = &tmp;
    scanf ("%d %d", &p->x, &p->y);

    printf ("%d %d\n", p->x*2, p->y*2);
    return 0;
}
```

Entrada: 10 11  
Saída: 20 22

(Onde está o erro? **\*p** é apenas um ponteiro, precisamos alocar memória!)

# Ponteiro (3)

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Ponto_s {
    int x;
    int y;
} Ponto;

int main () {
    Ponto *p = malloc (sizeof (Ponto));
    scanf ("%d %d", &p->x, &p->y);

    printf ("%d %d\n", p->x*2, p->y*2);
    free (p);
    return 0;
}
```

Entrada:	10	11
Saída:	20	22

(Onde está o erro? **\*p** é apenas um ponteiro, precisamos alocar memória!)

# Ponteiro (4)

```
#include <stdio.h>
#include <stdlib.h>
struct Ponto {
    int x;
    int y;
};

int main () {
    struct Ponto *p = malloc (sizeof (struct Ponto));
    scanf ("%d %d", &p->x, &p->y);

    printf ("%d %d\n", p->x*2, p->y*2);
    free (p);
    return 0;
}
```

Entrada: 10 11

Saída: 20 22

# Linguagem C

- Variáveis e Tipos de dados (`char`, `int`, `float`, `double`...)
- Condicionais (`if..else`)
- Repetição (`for`; `while`)
- Alocação estática
- Alocação dinâmica
- Ponteiros
- Estruturas (`struct`)
- • Funções e Procedimentos (modularização)

# Funções

```
#include <stdio.h>

int soma (int a, int b) {
    return a + b;
}

int main () {
    int a, b;
    scanf ("%d %d", &a, &b);

    printf ("%d\n", soma (a, b));
    return 0;
}
```

Entrada: 1 2

Saída: 3

**Assinatura:** <tipo devolvido> <nome da função> <argumentos>



# Procedimentos

```
#include <stdio.h>

void troca (int a, int b) {
    int tmp = a;
    a = b;
    b = tmp;
}

int main () {
    int a, b;
    scanf ("%d %d", &a, &b);

    troca (a, b);

    printf ("%d %d\n", a, b);
    return 0;
}
```

Entrada: 1 2

Saída: 1 2 (???)

# Procedimentos

```
#include <stdio.h>

void troca (int *a, int *b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

int main () {
    int a, b;
    scanf ("%d %d", &a, &b);

    troca (&a, &b);    // Passagem de argumentos por referência.

    printf ("%d %d\n", a, b);
    return 0;
}
```

Entrada: 1 2

Saída: 2 1