Computação eletrônica: Conceitos básicos

Gurvan Huiban
ghuiban@cin.ufpe.br

03 de avril de 2014

Plano de aula

1. Estrutura de um programa em C
2. Variáveis e constantes
3. Comandos de entrada e saída

Programar

Objetivo

Um programa deve realizar uma tarefa específica.

Requisitos

- Entender a tarefa
- Entender como realizar a tarefa:
  Como eu, programador, realizaria esta tarefa?

Escrever um programa

Objetivo

Escrever os comandos que o computador vai executar para realizar a tarefa desejada.

Requisitos

- Definir uma sequência de comandos a serem executados.
- Quem vai executar os comandos é o computador.
- Cuidado com a ordem dos comandos!

O programador

Requisitos

- Conhecer as regras da linguagem usada (aqui: Linguagem C)
- Rigor
- Paciência

Programar: exemplo

Hello world!

Escrever um programa que imprima na tela a frase:

```
Hello world!
```

Como fazer?

- O que deve fazer o programa?
  Imprimir "Hello world!" na tela
- De que precisamos?
  - *O básico para que o programa seja reconhecido*
  - *Saber como imprimir um texto na tela*

Estrutura básica

- Inclusão de bibliotecas (se necessário): `#include<biblioteca>`
- Marcador de início de programa: `int main(void)`
- Marcador de início das instruções: `{`
- Declaração das variáveis (se necessário)
- Sequência de comandos
- Marcador de fim das instruções: `}`

Programa Hello world!

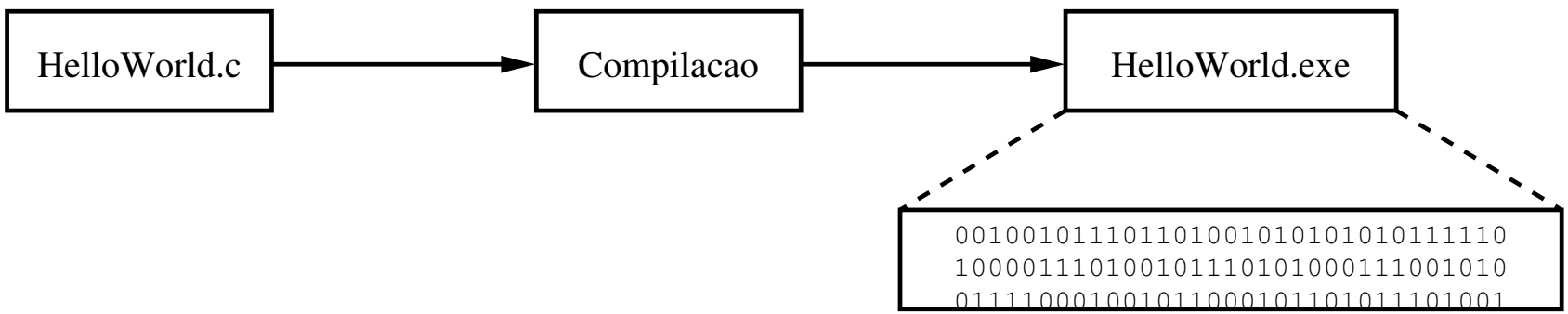
```c
#include<stdio.h>
int main(void)
{
  printf("Hello world\n");
  return 0;
}
```

Observações

- `stdio.h` é a biblioteca usada (contém o comando `printf`).
- As instruções são executadas na ordem de escrita.
- `printf` é o comando que imprime um texto na tela.
- Cada instrução termina com `;`
- Toda instrução deve estar entre os `{ }`
- Sem noção das instruções anteriores ou seguintes.

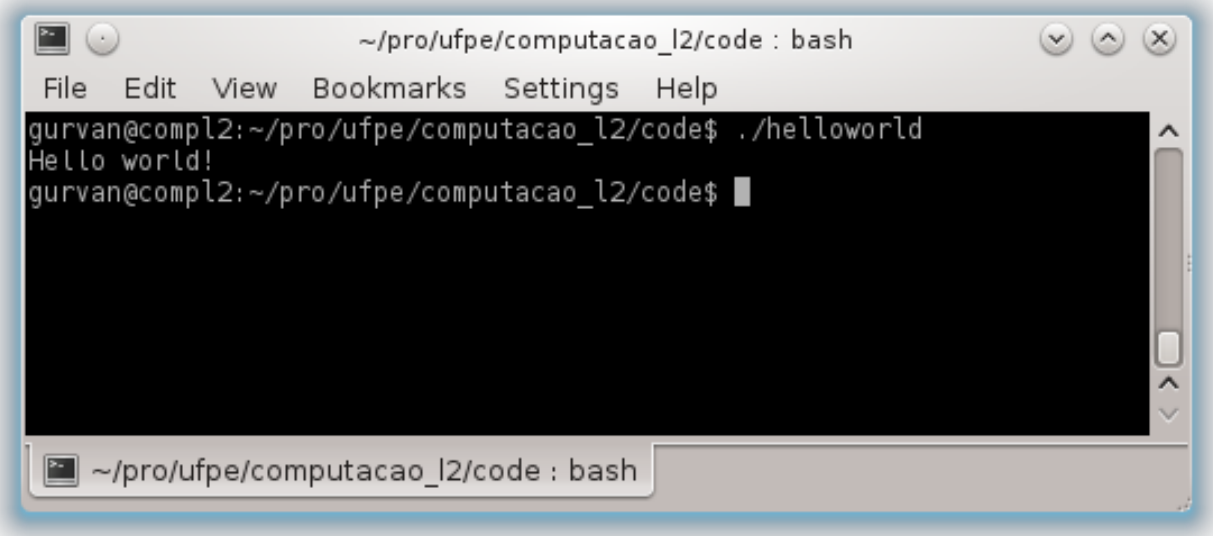
Compilação do Hello world!

Só falta compilar o programa...



Execução do Hello world!

e executar o programa:



Passo a passo (programa executável)

Programa compilado

```
00100101110110100101010101011110
10000111010010111010100011100101 0
01111000100101100010110101101001
```

Tela

Passo a passo (programa executável)

Programa compilado

```
00100101110110100101010101011110
10000111010010111010100011100101 0
01111000100101100010110101101001
```

Tela

Passo a passo (programa executável)

Programa compilado

```
00100101110110100101010101011110
10000111010010111010100011100101 0
01111000100101100010110101101001
```

Tela

```
Hello world!
```

Passo a passo (programa executável)

Programa compilado

```
00100101110110100101010101011110
10000111010010111010100011100101
01111000100101100010110101101001
```

Tela

```
Hello world!
```

Passo a passo (Código fonte)

Código fonte

```
int main(void)
{
  printf("Hello world\n");
}
```

Tela

Passo a passo (Código fonte)

Código fonte

```
int main(void)
{
  printf("Hello world\n");
}
```

Tela

Passo a passo (Código fonte)

Código fonte

```
int main(void)
{
  printf("Hello world\n");
}
```

Tela

Passo a passo (Código fonte)

Código fonte

```
int main(void)
{
  printf("Hello world\n");
}
```

Tela

```
Hello world!
```

Passo a passo (Código fonte)

Código fonte

```
int main(void)
{
  printf("Hello world\n");
}
```

Tela

```
Hello world!
```

Variáveis e constantes

Constante

Uma constante tem valor:

- Fixo
  Ele não muda durante a execução do programa.
- Conhecido no momento da escrita do programa.

Variável

Uma variável tem valor:

- que pode mudar durante a execução do programa.
- não necessariamente conhecido no momento da escrita do programa.

Constantes

Exemplos

- `3.14`
- `'S'`
- `"Hello world!"`

Observações

- Carácter: entre aspas simples `'`
- Cadeia de caracteres: entre aspas duplas `"`

Memória

Analogia do gaveteiro

Em cada gaveta, guardamos uma informação.
Exemplo:

- Um número inteiro
- Um número real
- Um carácter

Variável = gaveta

Definição

Uma variável é definida por

- Um nome
  Post-it colado na gaveta
- Um tipo
  Descrição do conteúdo da gaveta.
- Um valor
  Valor efetivo na gaveta

Exemplos de tipo

- Inteiro: `int`
- Carácter: `char`
- Número real: `float`

Tipos

Por que?

- Precisa de mais espaço para representar um número real que um inteiro
  $\Rightarrow$O número real ocupa mais espaço na memória
  $\Rightarrow$O tipo indica o espaço ocupado na memória
- Especificidade de operações:
  - raiz quadrada de um carácter?
  - Divisão inteira de um número real?

Nome de variáveis

Regras

- Uma letra seguida de letras, dígitos ou _
- Não é permitido espaço em branco ou outros caracteres como: `@ * ; , . /`
- Diferença entre maiúsculas e minúsculas: `MinhaVariavel` é diferente de `minhavariavel`
- Cada nome deve ser único.

Exemplos

- Permitido: `A`, `Nota`, `Matricula`, `Lucro_Total`
- Proibido: `5B`, `X-Y`, `A:B`, `Terca-Feira`, `km/h`

Declaração de variáveis

No início do programa (antes das instruções)

```
int main(void)
{
  int i,j;
  char cont;
  float x,y;
  int idade;
  ...
```

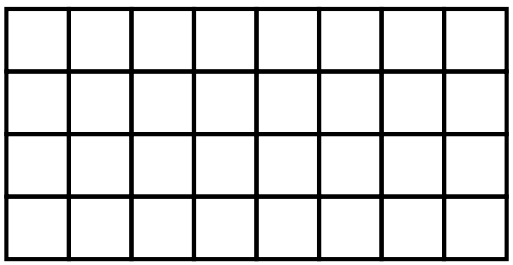
Boas práticas

- Nomes significativos, razoavelmente curtos
- Evitar nomes parecidos:

```
int idade, Idade;
```

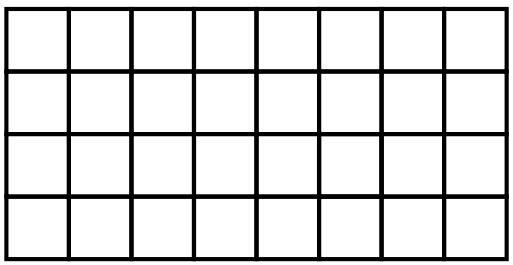
Declaração: Passo a passo

```
int main(void)
{
  int i,j;
  char cont;
  float x,y;
  int idade;
  ...
```



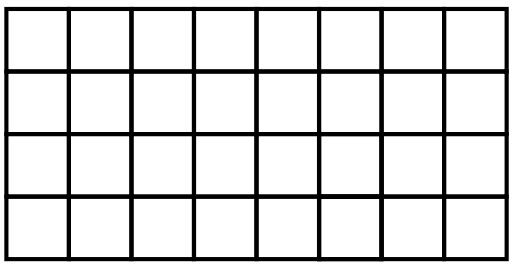
Declaração: Passo a passo

```
int main(void)
{
  int i,j;
  char cont;
  float x,y;
  int idade;
  ...
```



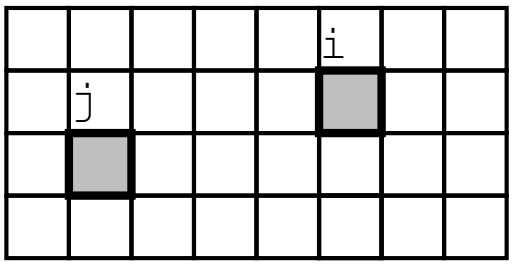
Declaração: Passo a passo

```
int main(void)
{
   int i,j;
   char cont;
   float x,y;
   int idade;
   ...
```



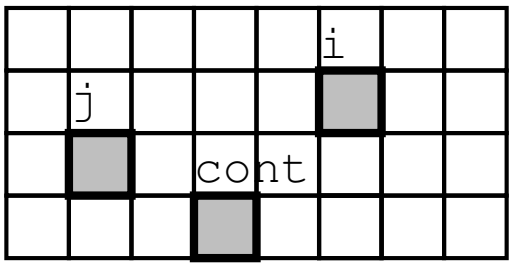
Declaração: Passo a passo

```
int main(void)
{
  int i,j;
  char cont;
  float x,y;
  int idade;
  ...
```



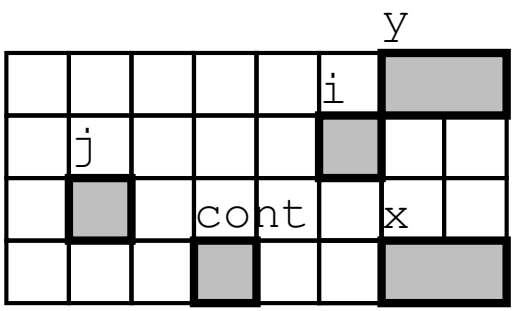
Declaração: Passo a passo

```
int main(void)
{
  int i,j;
  char cont;
  float x,y;
  int idade;
  ...
```



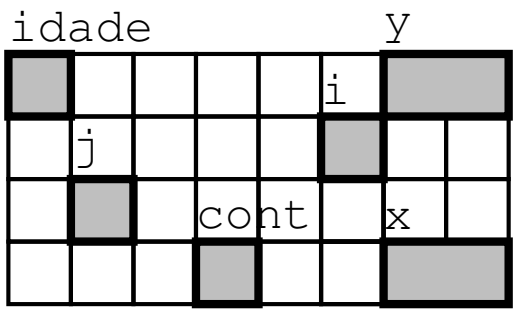
Declaração: Passo a passo

```
int main(void)
{
   int i,j;
   char cont;
   float x,y;
   int idade;
   ...
```



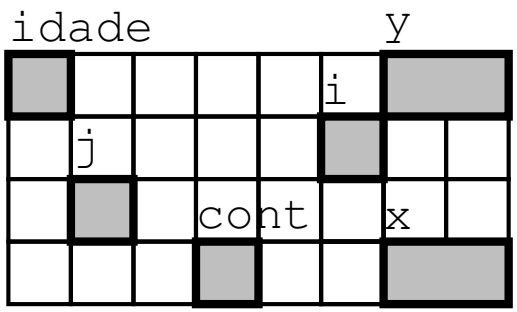
Declaração: Passo a passo

```
int main(void)
{
  int i,j;
  char cont;
  float x,y;
  int idade;
  ...
```



Declaração: Passo a passo

```
int main(void)
{
  int i,j;
  char cont;
  float x,y;
  int idade;
  ...
```



Palavras reservadas

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Não podem ser usada como nome de variável.

Atribuição

Operador =

Atribui um valor à uma variável.

```
int num,valor;
num = 12;
valor = num;
```

Observação

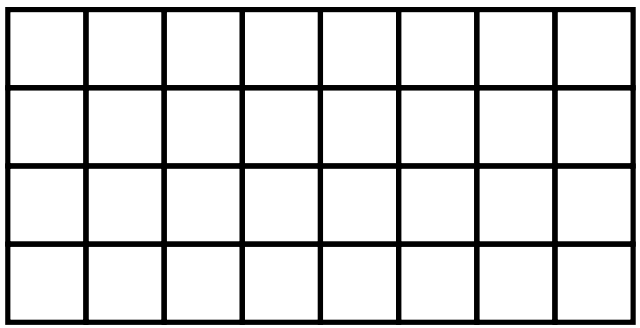
Não é simétrico:

```
12 = num;
```

não faz sentido na linguagem C!

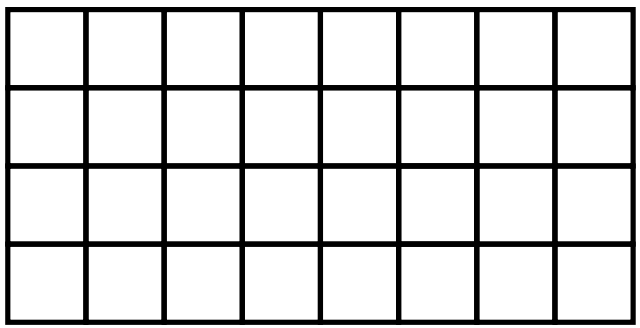
Atribuição: Passo a passo

```
int main(void)
{
  char cont;
  float x,y;
  int idade;
  cont = 's';
  x = 3.14;
  idade = 25;
  y = x;
  idade = idade+1;
  ...
```



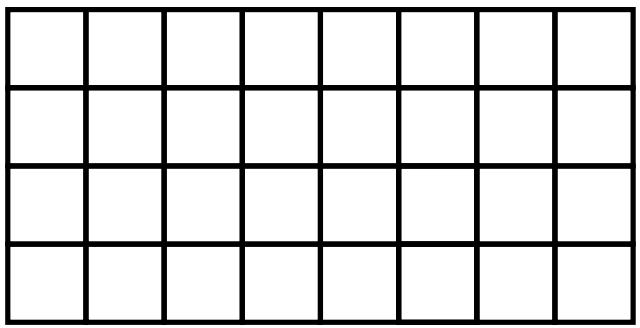
Atribuição: Passo a passo

```
int main(void)
{
   char cont;
   float x,y;
   int idade;
   cont = 's';
   x = 3.14;
   idade = 25;
   y = x;
   idade = idade+1;
   ...
```



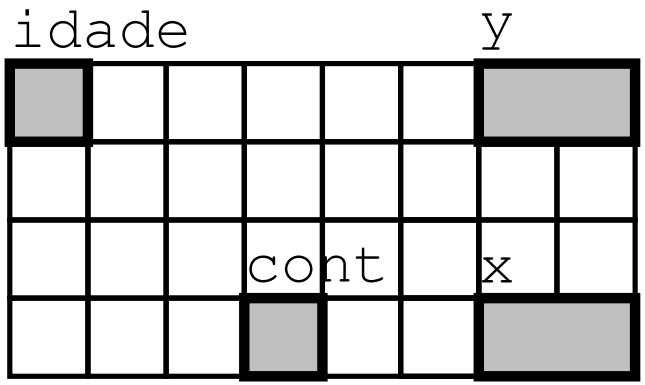
Atribuição: Passo a passo

```
int main(void)
{
   char cont;
   float x,y;
   int idade;
   cont = 's';
   x = 3.14;
   idade = 25;
   y = x;
   idade = idade+1;
   ...
```



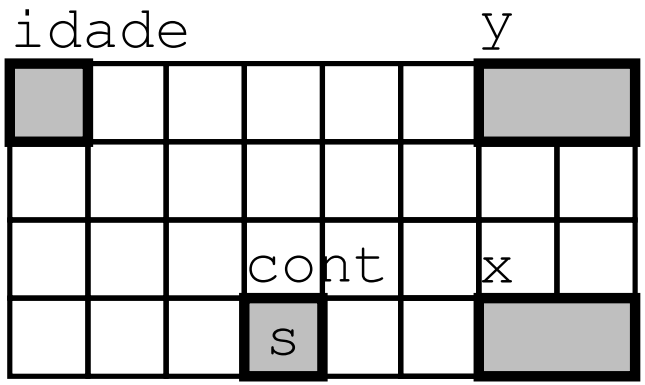
Atribuição: Passo a passo

```
int main(void)
{
  char cont;
  float x,y;
  int idade;
  cont = 's';
  x = 3.14;
  idade = 25;
  y = x;
  idade = idade+1;
  ...
```



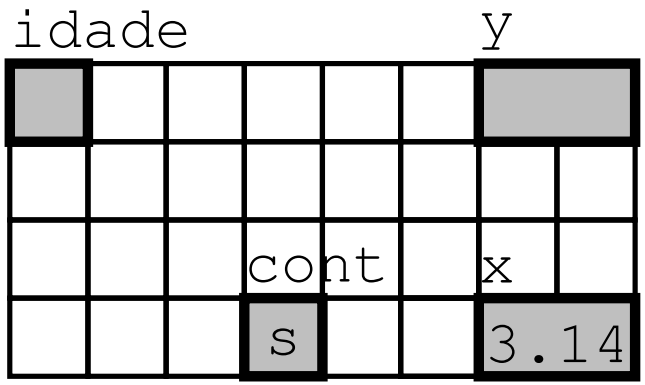
Atribuição: Passo a passo

```
int main(void)
{
  char cont;
  float x,y;
  int idade;
  cont = 's';
  x = 3.14;
  idade = 25;
  y = x;
  idade = idade+1;
  ...
```



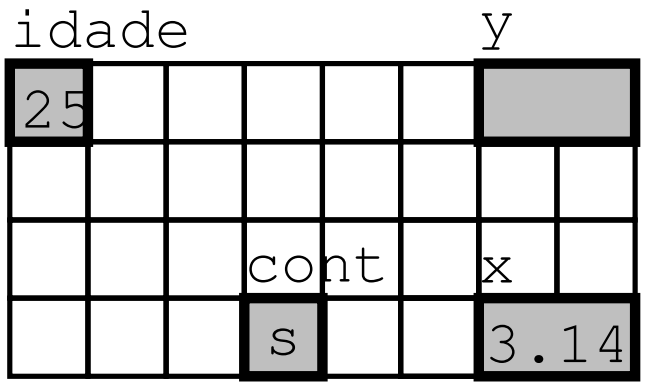
Atribuição: Passo a passo

```
int main(void)
{
   char cont;
   float x,y;
   int idade;
   cont = 's';
   x = 3.14;
   idade = 25;
   y = x;
   idade = idade+1;
   ...
```



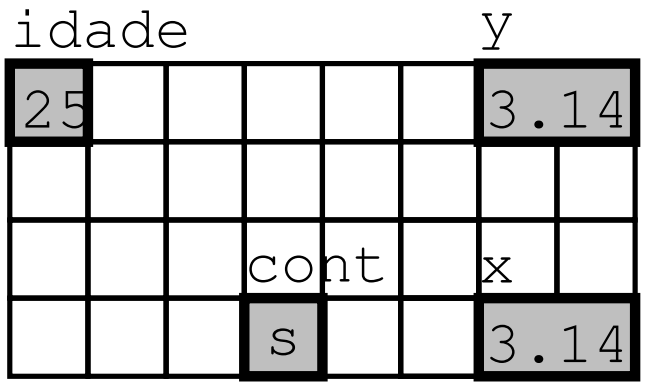
Atribuição: Passo a passo

```
int main(void)
{
   char cont;
   float x,y;
   int idade;
   cont = 's';
   x = 3.14;
   idade = 25;
   y = x;
   idade = idade+1;
   ...
```



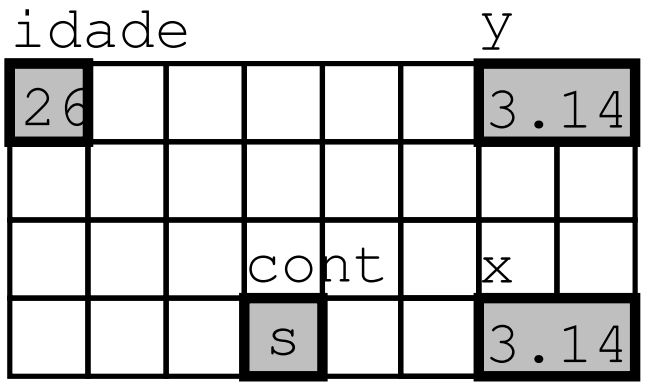
Atribuição: Passo a passo

```
int main(void)
{
  char cont;
  float x,y;
  int idade;
  cont = 's';
  x = 3.14;
  idade = 25;
  y = x;
  idade = idade+1;
  ...
```



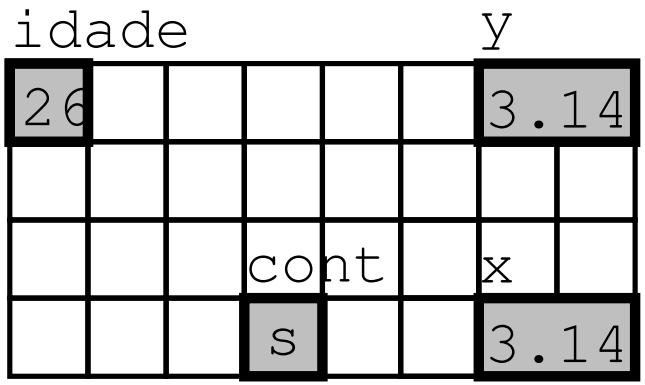
Atribuição: Passo a passo

```
int main(void)
{
   char cont;
   float x,y;
   int idade;
   cont = 's';
   x = 3.14;
   idade = 25;
   y = x;
   idade = idade+1;
   ...
```



Atribuição: Passo a passo

```
int main(void)
{
  char cont;
  float x,y;
  int idade;
  cont = 's';
  x = 3.14;
  idade = 25;
  y = x;
  idade = idade+1;
  ...
```



Exemplo: Idade

Idade

Escreva um programa que pergunta ao usuário a sua idade, e que a imprima na tela.

Entrada de dados

Objetivos

- Receber dados do usuário
- Receber dados de outros computadores

Equipamentos

- Teclado
- Webcam
- Touchscreen
- Microfone
- Placa de rede
- ...

Entrada de dados com `scanf`

`scanf`

Ler dados do teclado e armazena em variáveis.

Sintaxe

```
scanf("ControleTipo", &var1);
```

- `ControleTipo` : Descreve o tipo de dados esperado

```
scanf("%d", &idade);
scanf("%f", &altura);
```

- `&var1` : Valor lido armazenado na variável `var1`.

Cadeia de controle para entrada

Tipo de dados lidos

Cada tipo de dado está associado a uma cadeia de controle:

`%c`	Carácter
`%d` `%i`	Valor inteiro
`%f`	Valor real
`%s`	Cadeia de caracteres

Observação

O tipo lido deve ser de acordo com o tipo da variável:

```
float altura;
scanf("%f", &altura);
```

Saída de dados

Objetivos

- Informar o usuário
- Comunicação com outros computadores

Equipamentos

- Tela
- Impressora
- Placa de som
- Placa de rede
- ...

Impressão na tela

Regras

- Da esquerda para direita
- Quebra de linha $\Rightarrow$A esquerda de uma nova linha

Ou seja, (para nós) a impressão na tela é incremental.

Impressão com `printf`

Comando `printf`

- Impressão de texto na tela
- Impressão do valor de variáveis
- Formatação do texto

Sintaxe

```
printf("textoFormatado", &var1, &var2, &var3);
```

- `textoFormatado` : (Cadeia de controle)
  Descreve o quê, como e o valor de quais variáveis será impresso na tela
- `var1` : valor da variável `var1`.
- ...

Cadeia de controle para saída

- Mais complicada que para entrada
- Inclui
  - Texto
  - Códigos de tipo (`int`, `char`, ...)
  - Códigos de formatação (quebra de linha, tabulação, ...)

Exemplos

```
printf("Hello!\n");
printf("Voce tem %d anos\n", idade);
```

Se a variável `idade` contém o valor `25`, o programa vai imprimir na tela:

```
Hello!
Voce tem 25 anos
```

Códigos

Códigos de tipo

`%c`	Carácter
`%d`	Valor inteiro
`%i`	
`%f`	Valor real
`%s`	Cadeia de caracteres

Códigos de formatação

`\t`	Tabulação
`\n`	Quebra de linha
`\\`	\ (barra)
`%%`	% (percentagem)

Exercício

O que vai aparecer na tela do computador?

```
printf("Um\nDois\nTres!\n");
printf("%c e a letra numero %d do alfabeto\n",
       'e', 5);
printf("1\t2\t3\n4\t5\t6\n");
printf("x=%f\n", x);
```

(Assumindo que a variável `x` valha 3.14159)

Exemplo: Idade

Idade

Escreva um programa que pergunta ao usuário a sua idade, e que a imprima na tela.

```
int main(void)
{
    int idade;
    printf("Qual eh a sua idade? ");
    scanf("%d",&idade);
    printf("Sua idade eh %d\n", idade);
}
```

Tamanho dos campos

É possível estabelecer um tamanho mínimo para a impressão de um campo:

```
printf("x=%6d\n", 142);
printf("x=%3d\n", 142);
printf("x=%2d\n", 142);
```

Tela

```
x=   142
x=142
x=142
```

Precisão de um número reais

É possível definir a quantidade de casas decimais de um número real

```
printf("Pi=%.6f\n", 3.14159);
printf("Pi=%.3f\n", 3.14159);
printf("Pi=%.1f\n", 3.14159);
printf("Pi=%10.1f\n", 3.14159);
printf("Pi=%10.4f\n", 3.14159);
```

Tela

```
Pi=3.141590
Pi=3.142
Pi=3.1
Pi=       3.1
Pi=    3.1416
```