Ken Thompson e Dennis Ritchie

Linguagem C

- Linguagem Imperativa
- Denis Ritchie (1941-2011) – 1972 – (AT&T Bell Labs)
- Desenvolvido para a implementação do Unix
- Uma das linguagens de programação mais populares
- C++ começou por ser uma extensão do C

Estrutura básica

```
Diretivas para o pré-processador

declaração de variáveis globais

declaração de funções

int main () {

   declaração de variáveis locais
   ...

}
```

Estrutura básica

Diretivas para o pré-processador

declaração de variáveis globais

declaração de funções

int main () {

  declaração de variáveis locais

  ...

}

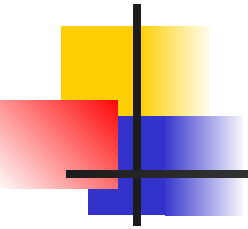
Exemplo:

```c
#include <stdio.h>

int a = 0;

int main() {
    int b;
    b = 7;

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

Variáveis

- Tipos básicos: **int**, **float**, **double**, **char**
  - sizeof() devolve o tamanho do tipo ou variável
    - `sizeof(int), int i; sizeof(i)`

```
main() {
     int valor_do_produto = 400;
     int grande;
     float x, y, somafinal;
     char sexo;
     char nome[10];

     somafinal = 37.0;
     sexo = 'm';
     grande = ( valor_do_produto > 1000 );
}
```



Estruturas de controlo

- if, if-else-if
- switch
- for (inicialização; cond; incremento) {}
- while (cond) {}
- do {} while (cond)

Estruturas de controlo

- if, if-else-if

```
#include <stdio.h>
main() {
      int a, b;

      printf ("Diga dois números: ");
      scanf ("%d%d", &a, &b);

      if ( b > a )
        printf ("B é maior do que A\n");

}
```

Estruturas de controlo

- if, if-else-if

```
#include <stdio.h>
main() {
      int a, b;

      printf ("Diga dois números: ");
      scanf ("%d%d", &a, &b);

      if ( b > a )
        printf ("B é maior do que A\n");
      else
        printf (”B não é maior do que A\n");

}
```

Estruturas de controlo

- if, if-else-if

```
#include <stdio.h>
main() {
      int a, b;
      printf ("Diga dois números: ");
      scanf ("%d%d", &a, &b);
      if ( b > a )
        printf ("B é maior do que A\n");
      else if ( a > b )
        printf ("A é maior do que B\n");
      else
        printf ("São iguais\n");
}
```

Estruturas de controlo

- Ciclos: for, while, do-while

```c
#include <stdio.h>
main() {
    int i, n, maior=0;
    i = 0;
    while ( i < 10 ) {
        printf ("Diga um número: ");
        scanf ("%d", &n);
        if ( n > maior ) maior = n;
        i++;
    }
    printf ("O maior é %d\n", maior);
}
```

Estruturas de controlo

- Ciclos: for, while, do-while

```c
#include <stdio.h>
main() {
    int i, n, maior=0;

    for ( i = 0; i <10; i++ ) {
        printf ("Diga um número: ");
        scanf ("%d", &n);
        if ( n > maior ) maior = n;
    }

    printf ("O maior é %d\n", maior);
}
```

Exemplos de operadores

Exemplo	Efeito
`x+y, x*y`	soma, multiplica
`x/y, x%y`	divisão, resto da divisão
`++x, x++`	incrementa 1 ao x, antes/depois de usar o seu valor
`--x, x--`	decrementa 1 ao x, antes/depois de usar o seu valor

`x += y`
`x *= y`
`x %= y`
	x = x + y
	x = x * y
	x = x % y

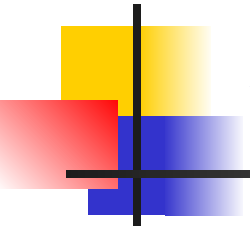
`==`
`!=`
`<=`
	igual
	diferente
	menor ou igual

`&&, ||, !`	operadores lógicos: e, ou, negação

Funções

- Sintaxe: `tipo nome(argumentos)`
  - Declarar a função antes da usar

```
#include <stdio.h>

void contagem ( int a ) {
    while ( a-- ) printf ("%d\n", a);
}

int cinco () {
    return 5;
}

main() {
    int i = cinco();
    contagem(i);
}
```



Arrays

- Exemplos:
  - `int a[5];`
    - dimensão 5 (índices de 0 a 4);
  - `int a[] = {1, 2, 3};`
  - `char s[] = {'a', 'b', 'c'};`
    - dimensão 3
  - `int c[5] = {1, 2, 7};`
    - dimensão 5, inicializa só as 3 primeiras posições;
- Cuidado não existe .length()

“strings”

- Em C não existe o tipo String
  - Em C, Strings são um array do tipo char que termina com o caracter 0 (‘\0’).
    - `char s[]={'h','e','l','l','o','\0'};`
  - Numa atribuição, o compilador coloca o ‘\0’.

```
#include <stdio.h>

main() {
   char s[10] = "batata";
   printf("%s\n", s);
}
```

s:

b	a	t	a	t	a	\0	?	?	?

“strings”

- Para imprimir um dado array de caracteres, independentemente do número de posições

```
for (i=0; s[i] != '\0'; i++ ) {
     printf ("%c", s[i]);
}
```

“strings”

- Para imprimir um dado array de caracteres, independentemente do número de posições

```
for (i=0; s[i] != '\0'; i++ ) {
     printf ("%c", s[i]);
}
```

Ou então

```
printf(”%s", s);
```

“strings”: funções

- `strcpy(destino, origem)`: Copiar de uma para outra
- `strcat(para,de)`: Concatenar duas strings
- `strlen`: Obter o tamanho
- `strcmp(s1,s2)`: Comparar duas strings. Retorna 0 se forem iguais
- ...

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

main() {
    char s1[10];
    strcpy(s1, "batata");
    if ( strlen(s1) == 6 )
      printf("%s\n", s1);
}
```

“strings”: funções

- `strcpy(destino, origem)`: Copiar de uma para outra
- `strcat(para,de)`: Concatenar duas strings
- `strlen`: Obter o tamanho
- `strcmp(s1,s2)`: Comparar duas strings. Retorna 0 se forem iguais
- ...

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

main() {
    char s1[10];
    strcpy(s1, "batata");
    if ( strlen(s1) == 6 )
      printf("%s\n", s1);
}
```

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

main() {
    char s1[10] = "batata";
    char s2[20], s3[5];
    strcpy(s2, s1);
    printf("%s\n", s2);
}
```

Leitura e escrita

```
printf ("formatos", var1, var2,...)
scanf ("formatos", &var1, &var2,...)
fgets(), ...
```

```c
#include <stdio.h>

main() {
    int idade;
    char nome[40];

    printf("Diga nome: ");
    scanf("%s", nome);
    printf("Diga idade: ");
    scanf("%d", &idade);

    printf("Seu nome: %s, sua idade: %d\n",nome, idade);
}
```

Leitura e escrita

```
printf ("formatos", var1, var2,...)
scanf ("formatos", &var1, &var2,...)
fgets(), ...
```

```
#include <stdio.h>

main() {
    int idade;
    char nome[40];

    printf("Diga nome: ");
    scanf("%s", nome);
    printf("Diga idade: ");
    scanf("%d", &idade);

    printf("Seu nome: %s, sua idade: %d\n",nome, idade);
}
```

%d	inteiro
%f	float
%c	char
%s	palavra
%lf	double
%g	double
%p	ponteiro

Leitura e escrita

- `scanf("%s",nome)` lê apenas uma palavra
  - no caso de introduzir duas palavras a segunda será lida no scanf seguinte (erro)

m	a	r	i	a		r	i	t	a	\n
1	7	\n								

```
#include <stdio.h>

main() {
    int idade;
    char nome[40];

    printf("Diga nome: ");
    scanf("%s", nome);
    printf("Diga idade: ");
    scanf("%d", &idade);

    printf("Seu nome: %s, sua idade: %d\n",nome, idade);
}
```

Leitura e escrita

- `fgets` lê uma linha inteira, incluindo o '\n'

```
#include <stdio.h>
#include <string.h>
main() {
    int idade;
    char tmp[50], nome[40];

    printf("Diga nome: ", nome);
    fgets(nome, 40, stdin);
    nome[ strlen(nome) - 1 ] = '\0';

    printf("Diga idade: ");
    fgets(tmp, 50, stdin);
    idade = atoi ( tmp );

    printf("Seu nome: %s, sua idade: %d\n",nome, idade);
    printf("O seu nome tem %d letras\n", strlen(nome);
}
```

Ficheiros

- Escrever

```
main() {
    FILE *f;
    f = fopen ("teste.txt", "w");




    fprintf (f, "Hello World\n" );
    fclose ( f );
}
```

Ficheiros

- Escrever
  - verificando se ocorreu algum erro

```
main() {
    FILE *f;
    f = fopen ("teste.txt", "w");
    if ( f == NULL ) {
        printf ( "não correu bem....\n");
        exit ( 1 );
    }
    fprintf (f, "Hello World\n" );
    fclose ( f );
}
```

Ficheiros

- Ler: `fscanf(f,...)` `fgets(...,f)`

```c
#include <stdio.h>
main() {
    FILE *f;
    char s[80];
    int i = 0;
    f = fopen ("teste.txt", "r");
    while ( fgets (s, 100, f ) != NULL ) {
        printf ( "%d: %s", ++i, s );
    }
    printf ("\n");
    fclose ( f );
}
```

Estruturas

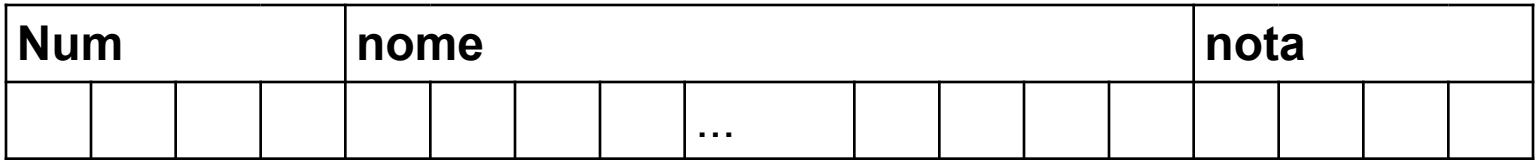
- declaração
  - `a.num = 10;`
  - `a.nota = c.nota+1`

```
struct Aluno {
   int num;
   char nome[100];
   int nota;
} a, b, c;
```

Estruturas

- declaração
  - `a.num = 10;`
  - `a.nota = c.nota+1`

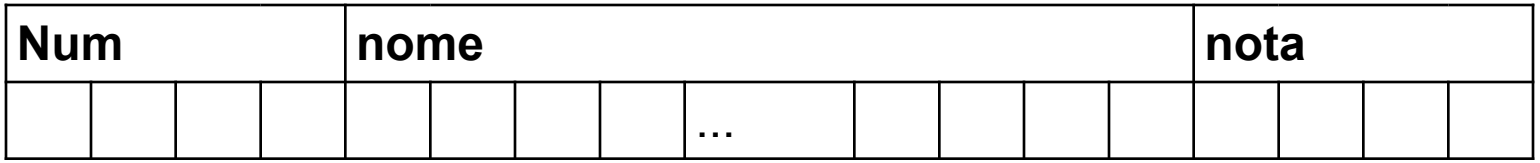
```
struct Aluno {
   int num;
   char nome[100];
   int nota;
} a, b, c;
```



Estruturas

- declaração
  - `a.num = 10;`
  - `a.nota = c.nota+1`

```
struct Aluno {
   int num;
   char nome[100];
   int nota;
} a, b, c;
```



  - `sizeof(struct aluno)`
    - 108 bytes
  - `sizeof(a)`
    - 108 bytes

Estruturas

- declaração
  - `a.num = 10;`
  - `a.nota = c.nota+1`

```
struct Aluno {
   int num;
   char nome[100];
   int nota;
} a, b, c;
```

- typedef
  - `typedef float Peso`
  - `Peso x;`
  - `x=98.3;`

```
typedef struct {
   int num;
   char nome[100];
   int nota;
} tipoAluno;

tipoAluno a, b, c;
int i;
```

Ponteiros

- Memória dinâmica
  - malloc permite reservar mais memória
  - p é um apontador para um inteiro

```
#include <stdio.h>
#include <stdlib.h>
main() {
     int a, *p;
     p = malloc ( sizeof(int) );
     a = 10;
     *p = 10;
     printf ( "%d %d\n", a, *p );
}
```

```
10 10
```

Ponteiros

- O seguinte programa possivelmente resultaria num erro

```
#include <stdio.h>
#include <stdlib.h>
main() {
    int *p;
    *p = 10;
    printf ( "%d\n", *p );
}
```

Ponteiros

- Reservar espaço para 10 inteiros
  - *(p+1) representa a posição de mem seguinte

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

const TAMANHO = 10;
main() {
    int a, *p = NULL;
    p = malloc ( sizeof(int) * TAMANHO );
    *p = 15;
    *(p+1) = 20;
    *(p+2) = *p + 10;
    printf ( "%d %d %d\n", *p, *(p+1), *(p+2) );
}
```

```
15 20 25
```

Ponteiros

- Reservar espaço para 10 inteiros
  - código equivalente ...

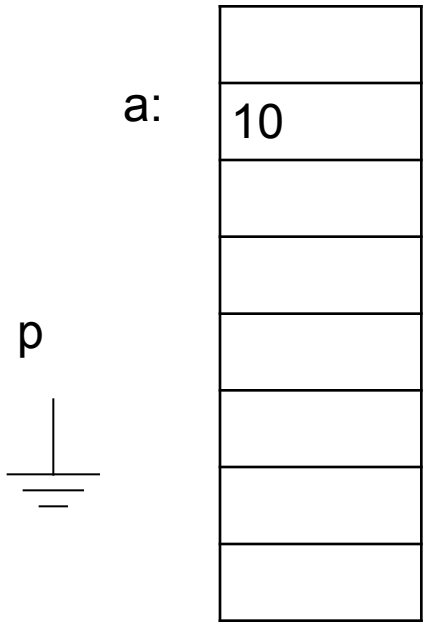
```c
#include <stdio.h>
#include <stdlib.h>

#define TAMANHO 10
main() {
    int a, *p = NULL;
    p = malloc ( sizeof(int) * TAMANHO );
    *p = 15;
    *(p+1) = 20;
    *(p+2) = *p + 10;
    printf ( "%d %d %d\n", p[0], p[1], p[2] );
}
```

```
15 20 25
```

Ponteiros

- As duas variáveis apontam para a mesma posição de memória

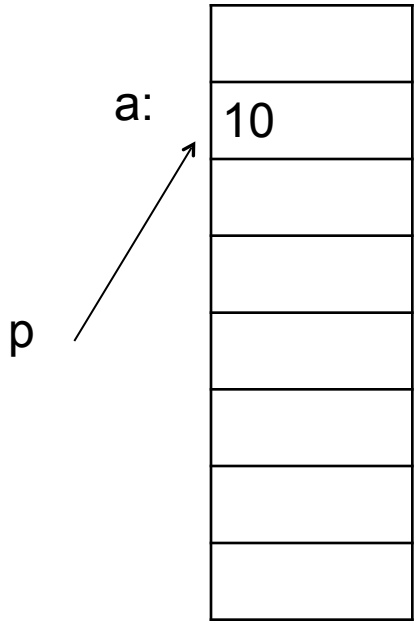
```
#include <stdio.h>
#include <stdlib.h>
main() {
    int a, *p = NULL;
    a = 10;
    p = &a;
    printf ( "%d %d\n", a, *p );
    a = 66;
    printf ( "%d %d\n", a, *p );
}
```



Ponteiros

- As duas variáveis apontam para a mesma posição de memória

```
#include <stdio.h>
#include <stdlib.h>
main() {
    int a, *p = NULL;
    a = 10;
    p = &a;
    printf ( "%d %d\n", a, *p );
    a = 66;
    printf ( "%d %d\n", a, *p );
}
```



Ponteiros

- As duas variáveis apontam para a mesma posição de memória

```c
#include <stdio.h>
#include <stdlib.h>
main() {
    int a, *p = NULL;
    a = 10;
    p = &a;
    printf ( "%d %d\n", a, *p );
    a = 66;
    printf ( "%d %d\n", a, *p );
}
```

```
10 10
66 66
```