Linguagem C - Uma Breve Introdução

Prof. Antonio L. Apolinário Jr.

UFBA-IM-DCC 2015.2

Roteiro

- Conceitos básicos
- Tipos de Dados
- Entrada e Saída formatada
- Estruturas de Controle
- Exercícios

Conceitos Básicos

Linguagem C - Conceitos Básicos

- História
  - Criada entre 1969 e 1973
    - por Dennis Ritchie no *Bell Telephone Laboratories*
    - vinculada ao sistema operacional Unix
    - para o PDP-11
      - 24Kbytes de memoria
        - 12K para o SO !
      - Processador de 16-bits

Linguagem C - Conceitos Básicos

- Linguagem:
  - De propósito geral
  - Imperativa (*procedural*)
  - Voltada a desenvolvimento de sistemas computacionais
  - Portável
    - compiladores para praticamente todos os sistemas computacionais atuais

Linguagem C - Conceitos Básicos

- Um programa mínimo em C:

```
/* Programa Minimo*/
#include<stdio.h>
int main()
{
    printf("Alo Estrutura de Dados!!\n");
    // outros comandos
    return 0;
}
```

Comentários
Headers
Funções
Bloco de Comandos
Chamadas a Funções
Retorno de Funções

Linguagem C - Conceitos Básicos

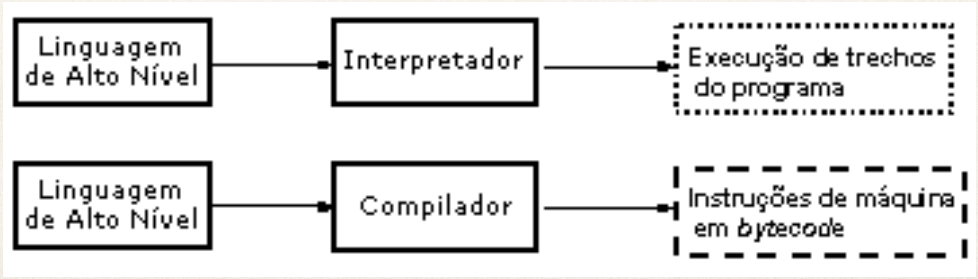
- Arquivos de cabeçalho (*header*):
  - Possuem a extensão `.h`
  - Contem a definição das funções utilizadas no módulo cujo código objeto vai mais tarde ser ligado.
  - São incluidos no código ANTES da compilação
    - Comando do pré-processador `#include`

Linguagem C - Conceitos Básicos

- Função `Main()`:
  - Ponto de entrada de um programa
  - Obrigatória em qualquer programa C
  - Pode receber parâmetros
    - `int main(int argc, char *argv[])`
    - durante a chamada do programa
      - linha de comando

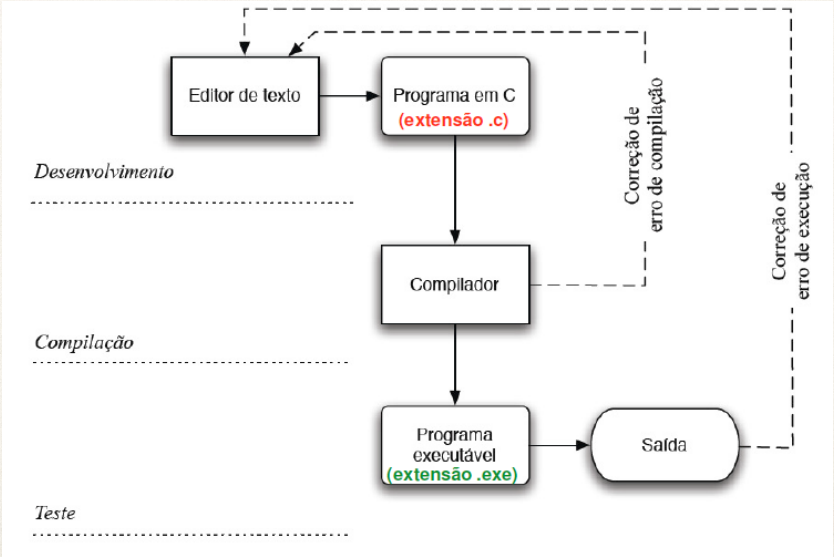
Linguagem C - Conceitos Básicos

- Tradução de um programa em linguagem de alto nível para linguagem de máquina
  - Duas abordagens
    - Interpretação
      - Python, Perl, JavaScript, PHP, Lua, Ruby, etc.
    - Compilação
      - C, C++, Pascal, Cobol, Fortran, Objective-C



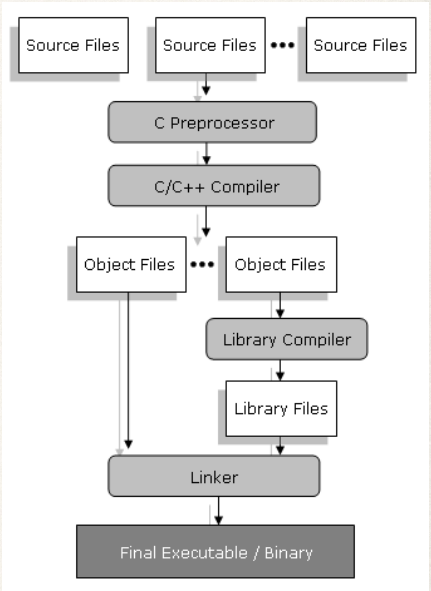
Linguagem C - Conceitos Básicos

- Linguagem de programação compilada:
  - Ciclo de Desenvolvimento



Linguagem C - Conceitos Básicos

- Compilação:
  - `gcc`
    - Portável
    - Gratuito
    - Não proprietário
  - Duas etapas:
    - Compilação
      - `gcc -c ProgramaMinimo.c`
    - Linkedição
      - `gcc -o ProgramaMinimo ProgramaMinimo.o`



Linguagem C - Conceitos Básicos

- Compilação:
  - Utilitário Make:
    - Automatização de processos de compilação, instalação e configuração em ambientes *Unix-like*
  - Exemplos de uso:
    - `make`
      - interpreta e executa os comandos do arquivo **`Makefile`** no diretório corrente
    - `make -f Makefile.MacOs`
      - interpreta e executa os comandos do arquivo **`Makefile.MacOs`** do diretório corrente

Linguagem C - Conceitos Básicos

- Exemplo de `makefile` simples:

```
.c.o:    $*.h
   gcc -c $*.c

.cpp.o:    $*.h
   g++-c $*.cpp
```

```
all:   ProgramaMinimo
```

```
ProgramaMinimo:  ProgramaMinimo.o
   gcc -o $@ $^
```

```
clean:
   rm *.o ProgramaMinimo
```

Regras de compilação
Regra *default*
Regras de geração dos executáveis
Regras específicas

Tipos de Dados

Tipos de Dados em Linguagem C

- Tipos básicos compatíveis com os tipos nativos do processador
  - Tradução natural para o *assembly*

Tipo	Tamanho	Menor valor	Maior valor
`char`	1 byte	-128	+127
`unsigned char`	1 byte	0	+255
`short int (short)`	2 bytes	-32.768	+32.767
`unsigned short int`	2 bytes	0	+65.535
`int (*)`	4 bytes	-2.147.483.648	+2.147.483.647
`long int (long)`	4 bytes	-2.147.483.648	+2.147.483.647
`unsigned long int`	4 bytes	0	+4.294.967.295
`float`	4 bytes	-10^{38}	$+10^{38}$
`double`	8 bytes	-10^{308}	$+10^{308}$

Variáveis em Linguagem C

- Declaradas a partir de um tipo de dados e um identificador
  - Regras usuais para o identificador:
    - Não pode ser uma palavra reservada
    - Iniciar com um letra ou *underscore*(_)
    - Pode conter letras, números e *underscore*
    - Nenhum outro caractere especial pode ser usado
    - Diferencia maiúsculas de minúsculas

Variáveis em Linguagem C

- Exemplos com tipos básicos:
  - `int i;`
  - `int *j, k=30;`
  - `unsigned char *ch;`
  - `double x=12.0;`
  - `long double a, b, c;`

Operadores em Linguagem C

Operador	C
Atribuição	=
Aritméticos	+ - * / %
Relacionais	< > <= >= == !=
Lógicos	&& || !
Bit-a-bit	& |
Incremento/Decremento	++ --
Atribuição Composta	+= *= -= /= %= &= |=
Deslocamento	<< >>

Operadores em Linguagem C

- Operadores e os tipos envolvidos:
  - Qual o resultado armazenado na variável `c`?

```
int a = 5;
int b = 2;
float c;
c = a / b;
```

- Conversão forçada de tipo (*type cast*):

```
int a = 5;
int b = 2;
float c;
c = (float)a / (float)b;
```

Entrada e Saída formatada

Entrada e Saída em C

- Funções da biblioteca padrão do sistema
  - **`#include <stdio.h>`**
- Duas funções básicas:
  - Entrada:
    - **`int scanf ( const char * format, ... );`**
  - Saída:
    - **`int printf ( const char * format, ... );`**
  - onde:
    - **`format`** é uma string contendo a descrição da entrada ou da saída;
    - ... representa a lista de variáveis que serão lidas ou escritas
  - Por *default* a entrada padrão é o teclado e a saída padrão o console (terminal).

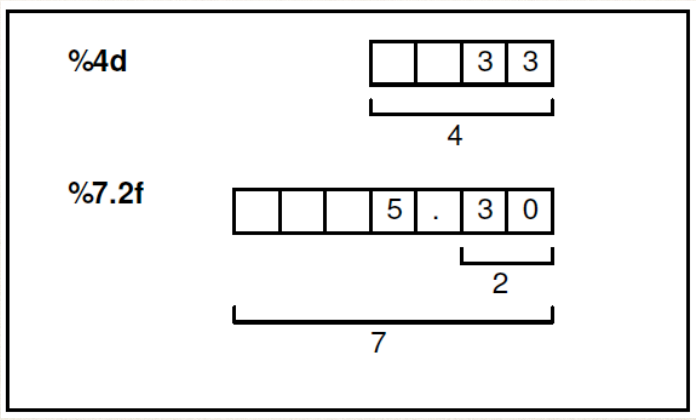
Entrada e Saída em C

- *Flags* de formatação da entrada/saída

specifier	Output	Example
`d` *or* `i`	Signed decimal integer	`392`
`u`	Unsigned decimal integer	`7235`
`o`	Unsigned octal	`610`
`x`	Unsigned hexadecimal integer	`7fa`
`X`	Unsigned hexadecimal integer (uppercase)	`7FA`
`f`	Decimal floating point, lowercase	`392.65`
`F`	Decimal floating point, uppercase	`392.65`
`e`	Scientific notation (mantissa/exponent), lowercase	`3.9265e+2`
`E`	Scientific notation (mantissa/exponent), uppercase	`3.9265E+2`
`g`	Use the shortest representation: `%e` or `%f`	`392.65`
`G`	Use the shortest representation: `%E` or `%F`	`392.65`
`a`	Hexadecimal floating point, lowercase	`-0xc.90fep-2`
`A`	Hexadecimal floating point, uppercase	`-0XC.90FEP-2`
`c`	Character	`a`
`s`	String of characters	`sample`
`p`	Pointer address	`b8000000`
`n`	Nothing printed. The corresponding argument must be a pointer to a `signed int`. The number of characters written so far is stored in the pointed location.	
`%`	A `%` followed by another `%` character will write a single `%` to the stream.	`%`

Entrada e Saída em C

- Formatação da entrada/saída
  - Mascaras indicando tipo e formato do dado:
    - `%[flags][width][.precision][length]specifier`



Entrada e Saída em C

- Exemplo:

```
#include <stdio.h>

int main (void) {
   int numero1, numero2, diferenca;
   printf (“Digite o primeiro número inteiro: “);
   scanf (“%d”, &numero1);
   printf (“Digite o segundo número inteiro: “);
   scanf (“%d”, &numero2);
   diferenca = numero1 - numero2;
   printf (“Resultado da diferenca = %d”, diferenca);
   return 0;
   }
```

Estruturas de Controle

Estruturas de Controle em C

- Fluxo de execução de um programa é sempre sequencial
  - PC = PC + 1;
- Estruturas de controle alteram essa execução sequencial:
  - Condicional
  - Estruturas de repetição
    - Incondicional
    - condicional

Estruturas Condicionais em C

- Se... então... senão

```
if (expr-cond)
        comando
else
        comando
```

```
if (expr-cond) {
        comandos
        }
else {
        comandos
        }
```

Opcionais

Estruturas Condicionais em C

- Exemplo:

```
float nota;
(....)
if (nota < 3.0)
printf("Reprovado");
else
if (nota >= 5.0)
   printf("Aprovado");
else
   printf("Prova final");
```

- Expressão Relacional retorna um valor booleano:
  - 0 significa falso
  - qualquer outro valor diferente de zero significa verdadeiro.
- Erro comum:

```
if (nota = 10.0) // sempre V
   printf("Genio!!");
```

Estruturas Condicionais em C

- Caso...

```
switch (expr-ord) {
        case <valor-ord> : comandos
                break;
        case <valor-ord> : comandos
                break;
        .....
        default: comandos
                break;
        }
```

Estruturas Condicionais em C

- Exemplo:

```
#include <stdio.h>
int main(void) {
   int valor;
   scanf("%d", &valor);
   switch(valor){
          case 0:
             printf("Valor e igual a 0");
             break;
          case 1:
             printf("Valor e igual a 1");
             break;
          case 2:
             printf("Valor e igual a 2");
             break;
          default:
             printf("Nenhuma das anteriores");
          }
   printf("\n\n");
   return 0;
}
```

Estruturas de Repetição em C

- Incondicional

```
for (expr-inic ; expr-cond ; expr-inc) {
      comandos
      }
```

Estruturas de Repetição em C

- Exemplo:

```
#include <stdio.h>
int main(void) {
int x;
      for(x=0 ; x<100 ; x++){
        printf("%d\n", x);
        }
      return 0;
      }
```

Estruturas de Repetição em C

- Condicional

```
while (expr-cond) {
      comandos
      }
```

de 0 a n repetições

```
do {
      comandos
} while (expr-cond)
```

de 1 a n repetições

Estruturas de Repetição em C

- Exemplos:

```
#include <stdio.h>
int main(void) {
   int x = 0;
   while(x < 100){
     printf("%d\n", x);
     x++;
     }
   return 0;
}
```

Estruturas de Repetição em C

- Exemplos:

```
#include <stdio.h>
int main(void) {
   int x = 0;
   while(x < 100){
     printf("%d\n", x);
     x++;
     }
   return 0;
}
```

Exercícios

Exercícios

1. Faça um programa em linguagem C que leia 2 números inteiros e calcule o MDC utilizando o ***Algoritmo de Euclides***. Veja o exemplo abaixo:

Passo	x	y	r
1	42	24	18
2	24	18	6
3	18	$\underline{6}$	0

2. Faça um programa em linguagem C que leia um número inteiro positivo é indique se ele é ou não primo. Considere que um número é dito primo se for divisível apenas pelo número 1 e pelo próprio número, sendo que 1 não é primo (2 é o primeiro número primo).
3. Faça um programa em linguagem C que leia um número inteiro positivo **`n`** e imprima em seguida a ***Série de Fibonacci*** até o seu n-ésimo termo.