Aula 4 – Introdução ao C

Considere o nosso MSC. O Processador Central, “entende” o conjunto de instruções, leia, imprima, atribuição e condicional e com ela resolvemos vários problemas, construindo vários algoritmos.

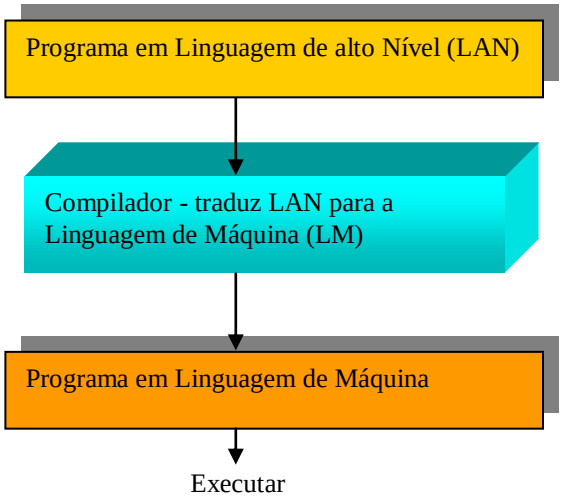
As instruções que definimos são de fato, instruções simbólicas (escritas em português), mas que seguem uma forma rígida de construção. Um computador real não “entende” direto uma linguagem simbólica como a que usamos. Além disso, os programas para serem executados precisam estar dentro da memória do computador, onde são armazenados apenas valores numéricos. A linguagem “entendida” pelos computadores reais são as linguagens de máquina.

A linguagem de máquina nada mais é que uma codificação numérica da linguagem simbólica.

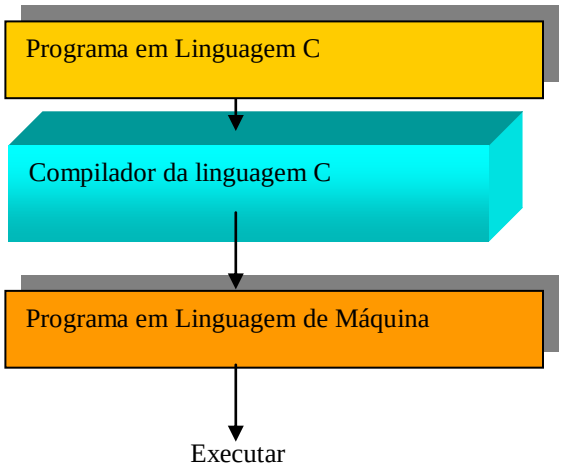
Linguagens simbólicas são chamadas de linguagem de alto nível. O computador não “entende” diretamente a linguagem de alto nível. Para que um programa ou um algoritmo possa ser executado pelo computador, este precisa ser traduzido para a linguagem de máquina.

A linguagem de alto nível, embora simbólica, possui um formato rígido (sintaxe). Daí poder ser traduzida automaticamente para a linguagem de máquina. Automaticamente quer dizer, ser traduzida por um programa que se chama COMPILADOR.

O processo de compilação



A função do compilador é, portanto traduzir um programa em LAN para LM. Cada LAN possui o seu compilador, que nada mais é que um programa tradutor desta LAN para a LM.
A linguagem C é uma LAN. Existem, portanto compiladores para a linguagem C.



Pequena história da linguagem C

A linguagem C foi criada por Denis Ritchie em 1972 no Bell Laboratories, o laboratório de pesquisas da empresa Bell nos EUA. Inicialmente, era utilizada como uma linguagem de programação de sistemas, isto é, uma linguagem usada para se programar os sistemas operacionais e softwares básicos em geral, principalmente ligados ao UNIX. Por suas características, logo começou a ser transformar numa linguagem de uso mais abrangente e por volta de 1980, já existiam várias versões comerciais de compiladores para o C.
O C, pertence a um conjunto de linguagens que derivaram do Algol e tem no C e Pascal os seus mais famosos descendentes. Na verdade a evolução do C, teve as seguintes linguagens:
ALGOL - 1960
CPL E BCPL - meados dos anos 60
B - 1970 (Ken Thompson, Bell Labs)
C - 1972 (Dennis Ritchie, Bell Labs)

Primeiros programas em C

Vejamos os comandos elementares em C

1) Entrada de números inteiros (`leia x`)

```
scanf("%d", &x)
```

A parte entre aspas é o formato e especifica que será lido um valor inteiro. Muitos outros tipos de formato existem que serão detalhados mais tarde. Observe que na frente do x tem o símbolo &. Isso significa "endereço de x" e será também melhor entendido futuramente.

Quando um número é digitado no teclado, o comando que pega este número, ou seja, a instrução ou comando que pega esse número e o coloca dentro da variável lida é um **`scanf`** como acima.

Para ler mais de um como em `leia x,y,z`

```
scanf("%d%d%d", &x, &y, &z)
```

2) Saída de valores inteiros (`imprima x`)

```
printf("%d", x)
```

Note que aqui não tem & antes da variável.

Para imprimir mais de um como em `imprima x,y,z`

```
printf("%d%d%d", x, y, x)
```

3) Atribuição

Como no MSC, só que o símbolo ← (seta para esquerda) é substituído pelo símbolo = (igual).

```
x = y
```

O valor de `y` é atribuido a `x`.

```
x = y + z
```

Atribui o valor da soma `y` mais `z` à `x`.

Agora podemos resolver em C, os 4 primeiros problemas já resolvidos no MSC.

P1) Dados 2 números, calcular a soma.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
 int x, y, z;
 scanf("%d %d", &x, &y);
 z = x + y;
 printf("%d", z);
 system("PAUSE");
 return 0;
}
```

Observe a semelhança com o MSC. Colocamos em azul o que é essencial e em vermelho as coisas adicionais necessárias para completar o programa.

Vamos detalhar um pouco cada uma destas linhas.

`a) #include <stdio.h>`
Indica que um arquivo chamado `stdio.h` que contém informações sobre entrada e saída será incluído no programa. Esse arquivo é sempre necessário quando se usam as funções `printf` e `scanf`. Existem outros arquivos que podem ser incluídos, dependendo das funções que são usadas no programa como veremos mais tarde.

`b) #include <stdlib.h>`
Indica que um arquivo chamado `stdlib.h` que contém informações sobre funções padrões do sistema será incluído no programa. Neste caso, a função que está sendo usada é a `system("PAUSE")`.

`c) int main()`
Indica que esse é o programa principal. Como veremos mais tarde, podem existir o programa principal e os secundários. O `int` na frente do `main` indica o tipo do `main` (inteiro).

`d) int x, y, z;`
As variáveis de um programa em C, que conforme já vimos, correspondem às posições de memória que serão usadas precisam ser declaradas. A linha acima, diz que nesse programa serão usadas três variáveis de nomes

`x`, `y` e `z` e do tipo `int` (inteiro). Existem outros tipos de variáveis, mas por enquanto, só usaremos o tipo `int`.

`e) system("PAUSE");`
Espera que o usuário digite alguma coisa para continuar.
Esse comando é usado apenas para permitir que o usuário veja na tela o que o seu programa imprimiu.

`f) return 0;`
Sai do programa com o valor 0.
O valor de saída não é importante no programa principal. Pode ser qualquer valor em vez de zero. Será muito importante quando virmos funções.

`g) { e }` (abre e fecha chaves)
Indicam o início e o final do programa. Na verdade servem para delimitar um bloco, que detalharemos mais tarde.

`h) ;` (ponto e vírgula)
Usado no final de cada comando, indicando que o mesmo terminou. O formato é livre, isto é, podem-se colocar comandos na mesma linha, deixar espaços em branco separando os vários elementos do programa, etc. Desta forma é necessário um indicador que um determinado comando acabou que é o ponto e vírgula.

Compare agora o programa acima com o algoritmo em MSC da aula anterior:

```
leia x, y
z = x + y
imprima z
```

Existe uma correspondência um a um entre os comandos nos dois casos. Em C são necessárias algumas coisas ("perfumarias") adicionais.

P2) Dados 2 números, calcular a soma, subtração, multiplicação e divisão

```
#include <stdio.h>
#include <stdlib.h>
int main() {
 int x, y, z;
 scanf("%d %d", &x, &y);
 z = x + y;
 printf(" %d ", z);
 z = x - y;
 printf(" %d ", z);
 z = x * y;
 printf(" %d ", z);
 z = x / y;
 printf(" %d ", z);
 system ("PAUSE");
 return 0;
}
```

P3) Dados 2 números calcular a média
Fica como exercício.

P4) Dados 2 triplas de números a1,b1,c1 e a2, b2 e c2, determinar x e y onde:
a1x+b1y=c1
a2x+b2y=c2
isto é, a solução do sistema de 2 a 2. Supor que existe solução.

Lembrar que x=(c1.b2-c2.b1)/(a1.b2-a2.b1) e y = (a1.c2-a2.c1) / (a1.b2-a2.b1)
Fica como exercício.

Organizando a saída dos resultados

Voltemos agora ao P2. Vejamos o que será impresso no vídeo, depois que o problema for executado. Vamos supor que são digitados os valores 120 e 5 como x e y.

```
120 5
 125  115  600  24
```

Observe que na linha seguinte a da digitação de 120 e 5 (em azul) foram impressos os resultados (em vermelho) 125 (em ver (soma), 115 (subtração), 600 (multiplicação) e 24 (divisão). Imagine agora um programa onde são digitados e impressos mais valores. A saída no vídeo ficaria confusa. Seria bom que pudéssemos imprimir no vídeo algumas frases que indicassem tanto o que deve ser digitado quanto o que foi impresso. Vejamos como fazer, dando outra solução ao P2.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
 int x, y, z;

 printf("entre com o primeiro:");
 scanf("%d", &x);

 printf("entre com o segundo:");
 scanf("%d", &y);

 z = x + y;
 printf("\n valor da soma:%d ", z);

 z = x - y;
 printf("\n valor da diferença:%d ", z);

 z = x * y;
 printf("\n valor do produto:%d ", z);

 z = x / y;
 printf("\n valor do quociente:%d ", z);

 system("PAUSE");
 return 0;
}
```

Nesse caso a saída ficaria:

```
entre com o primeiro:120
entre com o segundo:5
 valor da soma: 125
 valor da diferença:  115
 valor do produto: 600
 valor do quociente: 24
```

A saída ficou muito mais clara. O único elemento novo foi o \n, cuja função é apenas mudar de linha.

Agora vamos ao comando condicional em C:

P5) Dados 2 números (supor distintos) calcular o maior.

```
leia a, b
se a > b
   então imprima a
   senão imprima b
```

Em C:

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

 printf("entre com o primeiro:");
 scanf("%d", &a);

 printf("entre com o segundo:");
 scanf("%d", &b);

 if (a > b) printf("\n o maior é:%d ", a);
 else printf("\n o maior é:%d ", b);

 system("PAUSE");
 return 0;
}
```

Outra forma:

```
leia a, b
se a > b
   então maior ← a
   senão maior ← b
imprima maior
```

Em C:

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

 printf("entre com o primeiro:");
 scanf("%d", &a);

 printf("entre com o segundo:");
 scanf("%d", &b);

 if (a > b) maior = a;
 else maior = b;

 printf("\n o maior é:%d ", maior);

 system("PAUSE");
 return 0;
}
```

P6) Idem, supondo que podem ser iguais. Neste caso imprimir "iguais"

```
leia a, b
se a > b
   então imprima a
   senão se b > a
         então imprima b
         senão imprima "iguais"
```

Em C:

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

 printf("entre com o primeiro:");
 scanf("%d", &a);

 printf("entre com o segundo:");
 scanf("%d", &b);

 if (a > b) printf("\n o maior é:%d ", a);
 else if (b > a) printf("\n o maior é:%d ", b);
      else printf("\n são iguais");

 system("PAUSE");
 return 0;
}
```

Outra forma:

```
leia a, b
se a = b
   então imprima "iguais"
   senão se a > b
         então imprima a
         senão imprima b
```

Em C:

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

 printf("entre com o primeiro:");
 scanf("%d", &a);

 printf("entre com o segundo:");
 scanf("%d", &b);

 if (a == b) printf("\n são iguais");
 else if (a > b) printf("\n o maior é:%d ", a);
      else printf("\n o maior é:%d ", b);
```

```
 system("PAUSE");
 return 0;
}
```

Os exercícios abaixo, já foram resolvidos usando o MSC. Escreva agora os algoritmos em C.

P7) Dados 2 números imprimi-los em ordem crescente (o primeiro é menor ou igual ao segundo).

P8) Idem, verificando se os números são iguais e imprimindo "iguais" neste caso.

P9) Dados 3 números imprimir o maior. Supor distintos.

P9a) Idem determinando o menor

P10) Dados 3 números imprimi-los em ordem crescente (o primeiro menor ou ao segundo que é menor ou igual ao terceiro).

P10a) Idem imprimindo em ordem decrescente.

P11) Dados 3 números positivos verificar se são lados de um triângulo retângulo.

P12) Idem, verificando se são lados de algum triângulo, isto é, se o maior é menor que a soma dos outros dois.