{introcomp}

Working 03 : Conceitos Básicos I

Objetivos:

- Dominar os conceitos básicos da linguagem de programação C;
- Aprender a utilizar o compilador, identificando os erros de sintaxe do código fonte;

Prazo de Envio: segunda, 05/10, 18:00h.

1 INTRODUÇÃO

Agora sim vamos aprender a linguagem de programação C! Mas, saiba que sem a noção e domínio de algoritmos talvez você teria dificuldade de entender como funciona a linguagem de programação C. Como início, é importante entender que uma linguagem de programação obedece a regras rigorosas de escrita para que o compilador consiga traduzir seu código em linguagem de máquina. Portanto, é relevante entender bem o padrão de construção de programas em C e praticar muito pra não esquecer mais à frente um ';' no final das instruções ou um '&' no comando scanf. Além disso, lembre-se do compilador: ele é seu amigo, se você esqueceu um ';', ele vai te alertar, por isso, entenda bem o que ele está te alertando e mostre pra ele que você consertou o erro e o seu programa pode ser executado (:D)! Reveja com atenção o conteúdo passado em aula, assim como o conteúdo deste Working. Vamos começar!

2 COMPILADORES C PARA WINDOWS

Caso você não queira ou possa utilizar o Linux para a compilação de programas em C, é recomendável a utilização do **Codeblock** para a escrita e compilação dos códigos. Para compilar no Codeblock, basta clicar em build e, executar, em run. Além disso, você também pode escrever e compilar códigos online utilizando o site **Ideone**, escolhendo a linguagem C como opção, compilando e executando clicando no botão submit.

3 A FUNÇÂO MAIN

A função main() é onde você escreve o seu código em C. Mas você se lembra de como é formada?
Lembrando:

```
1 #include <stdio.h>
  int main(){
3 //programa
  return 0;
5 }
```

Observando o código, há no topo a inclusão das bibliotecas e logo abaixo a função main, que é por onde todos os comandos e instruções, ou seja, o algoritmo propriamente dito, são executados pelo computador. Lembrando também que as chaves '', '' tem como objetivo delimitar todo o código pertencente à main. É importantíssimo praticar para obter este escopo na mente, não tendo mais necessidade de buscá-la em livros, etc. Vamos falar mais de C!

4 VARIAVEIS E CELULAS DE MEMORIA

Quando os algoritmos são corretamente transcritos para uma linguagem de programação, os algoritmos são chamados de programas e podem ser executados pelo computador. É necessário, todavia, armazenar as informações utilizadas pelos programas em um local organizado

e seguro. Os computadores utilizam a Memória para armazenar os dados de um programa em execução.
A memória pode ser entendida como uma sequência de células, cada célula contém um número pré-definido de bits que varia de acordo com a arquitetura do computador podendo armazenar uma porção dos dados de um programa. Graças a essa ordenação sequencial, cada célula possui um número identificador chamado endereço, que está relacionado com a posição que a célula ocupa na sequência. Por meio dos endereços é possível acessar qualquer célula para ler ou alterar seu conteúdo.
Nos primórdios da programação, percebeu-se que acessar a memória por meio de endereços era trabalhoso demais e causa constante de erros. Isso porque o programador deveria escolher os endereços das células com as quais iria trabalhar tanto das células que teriam valores a serem lidos quanto das que seriam usadas para a escrita de resultados.
Para resolver essa questão, o conceito de variável foi criado. Uma variável nada mais é do que uma abstração para o endereço de memória. Com o emprego de variáveis, as células de memória são referenciadas nos programas por meio de rótulos, definidos com ajuda do bom-senso do programador. O compilador fica encarregado do trabalho de transformar rótulos em endereços para que as operações de acesso à memória sejam realizadas.
Apresentados os conceitos de variável e células de memória, é válido abordar um outro significado de programa. Trata-se de considerar os programas como sendo processos de mudança de estados.
Nessa abordagem, os dados de entrada, escritos em suas respectivas variáveis, são considerados o estado inicial do programa. A partir daí, realiza-se uma sequência de operações para chegar a um estado final. A cada operação, diz-se que o programa está em um novo estado intermediário. O estado final é atingido quando a tarefa em questão é considerada como realizada.

5 IDENTIFICADORES

Em geral, as linguagens de alto nível possuem elementos definidos pela própria linguagem – símbolos para operadores (+, -, *, /, <, >, && etc), nome de comandos (#include, #define, if, else, while), etc – e os elementos definidos pelo programador – identificadores, comentários, etc. Um identificador é um símbolo que pode representar alguma entidade criada pelo programador, como uma variável, por exemplo. Cada linguagem define uma regra para formação de identificadores. A regra de formação em C é a seguinte:

- Podem ser formados por letras, números, caracteres sublinhado e cifrão;
- Os números não podem aparecer na primeira posição;

Exemplos: cor, n1, n2, roda_1,roda_2.

6 TIPOS DE DADOS E DECLARAÇÃO DE VARIÁVEIS

Um tipo de dados delimita o conjunto de valores possíveis que uma determinada variável pode representar. Além disso, define as operações básicas possíveis para suas variáveis. Os tipos de dados simples utilizados em diversas linguagens de programação são o tipo: inteiro,

ponto flutuante, caractere e lógico (booleano).
Beleza, até agora sabemos sobre variáveis e sobre os tipos das mesmas, mas como começar a utilizá-las? Pois então, na linguagem C, para utilizar as variáveis, antes de tudo, é necessário declará-las. A seguir os tipos de dados simples e como declarar as variáveis com os seguintes tipos de dados. Mas antes de tudo, nunca se esqueça de inicializar uma variável com um valor!

6.1 Tipo Inteiro

O tipo inteiro pode armazenar números que vão da faixa de -2.147.483.648 até 2.147.483.647 (um total de 2$\hat{3}$2 números, ou 4.294.967.296). Você pode modificar esta faixa de valores usando as palavras reservadas signed, unsigned, short e long.
Exemplos de declaração:

```
int n1, n2 = 3, n3; //declarando variaveis do tipo int
long int l1;
unsigned int u1; 04 short int s1=3;
```

6.2 Tipo Ponto Flutuante

Representa números fracionários e números reais (o que inclui os números inteiros). A faixa de valores varia de 1,2e-38 até 3,4e+38. A declaração é feita através da palavra reservada float.
Há também um também, igualmente ao tipo float, utilizado para números fracionários e reais, porém com precisão ainda maior. A faixa de valores varia de de 2,2e-308 até 1,8e+308. A declaração é com a palavra reservada Double.
Exemplos de declaração:

```
float n1, n2 = 3.12; // declarando variaveis do tipo float
double d1 = 123.123;
```

6.3 Tipo Caractere

O tipo de dados caractere é utilizado para armazenar letras, números e outros caracteres especiais. A declaração do tipo dá-se através da palavra reservada char. Ao total, são 256 caracteres que podem ser armazenados, dos quais os 128 primeiros fazem parte do conjunto. Você pode declarar um caractere de duas formas: com ou sem aspas. Ao declarar um caractere sem as aspas simples, você deverá saber qual é o decimal correspondente. Por exemplo, se você digitar char caractere = 97, o console imprimirá a letra 'a'. Logicamente, você poderia ter declarado esta variavel com char caractere = 'a'.
Exemplos de declaração:

```
char c1='a', c2='s', n3; //declarando variaveis do tipo char
char c = 97; //pela tabela ASCII, 97 e o caractere 'a'
char c = 'fg' /*ATENCAO: observe que foi atribuido dois caracteres, mas a variavel char so
    armazena um por isso essa atribuicao esta incorreta.*/
```

6.4 Tipo Lógico (booleano)

O tipo booleano é o tipo de dado mais simples. Como já ensinado, ele possui apenas dois valores possíveis: verdadeiro e falso. É usado principalmente quando se precisam verificar condições no programa, em expressões lógicas e relacionais.
Na linguagem C, não há uma representação específica para esse tipo de dados e são utilizados valores inteiros para codifica-lo. Desta forma, todo valor inteiro diferente de zero é considerado valor verdadeiro e o valor zero é considerado falso.

6.5 Conversões de Tipos

O que aconteceria se eu atribuir 1.34 a uma variável do tipo inteiro? Um erro não será gerado pelo compilador, pois a linguagem C realiza algumas conversões de tipos de dados de maneira implícita.
Conversão implícita é aquela realizada pela linguagem sem que o programador tenha explicitamente solicitado que ela a fizesse (entretanto, fica subentendido que você deseja fazer a conversão, por isso ela é chamada de implícita). C converte implicitamente a maioria dos tipos de dados. Os principais são:

- De inteiros para caracteres e ponto flutuante;
- De ponto flutuante para inteiro e caracter;
- De caracter para inteiro e ponto flutuante.

Exemplos:

```
1 double a;
  int b;
3 char c = 'b';
  a = c; //neste caso, a recebe 98, valor numerico de 'b'
5 a = 1.23; //atribuindo um novo valor a variavel a
  b = a; //neste caso, b recebera o valor 1 e nao 1.23, observe portanto que ocorre um corte e
       nao arredondamento.
```

Praticando

Agora vamos praticar! Para todos os praticando em que é pedido que se escreva um programa, escreva o código do seu programa e nos envie o .c (código fonte) correspondente.

1. Quais dos seguintes nomes de variaveis são inválidos? Por quê?
   (a) int
   (b) char
   (c) 6_05
   (d) _var1
   (e) massa muscular
   (f) Z
   (g) alfa_beta_rotina
   (h) f#
   (i) _1312
   (j) Reinicializa
   (k) A$
2. Que tipo de variavel se usa para guardar os seguintes valores:
   (a) 2000
   (b) 'a'
   (c) 2.3
   (d) 23.432
3. Acrescente as partes necessárias, e complete os códigos de acordo com o que cada enunciado pede.
   (a) Faça um programa com a seguinte declaração de variáveis em C :
   inteiro : NFILHOS, IDADE;
   caractere: LETRA;
   real : VALOR, PESO;

```c
#include <stdio.h>
int main()
{
//declare as variaveis do tipo inteiro nessa linha
//declare a variavel do tipo caractere nessa linha
//declare as variaveis do tipo real nessa linha
  return 0;
}
```

   (b) Atribua valores à sua escolha às variáveis criadas no item (a). Atente-se ao tipo de cada variável.