Universidade Federal do Espirito Santo
Centro Tecnologico

{introcomp}

Working 02 : 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;

PET
Engenharia de Computação

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 FUNÇÃO MAIN

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

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

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 VARIÁVEIS E CÉLULAS DE MEMÓRIA

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:

```
double a;
int b;
char c = 'b';
a = c; //neste caso, a recebe 98, valor numerico de 'b'
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.
```

7 CONSTANTES

Em algumas situações surge a necessidade de se utilizar um determinado valor constante em diversas partes do código. Para simplificar a alteração dessas constantes e facilitar a leitura do código, é possível definir um nome significativo para elas de maneira simples, por meio de uma estrutura específica.
Essa estrutura é iniciada pela diretiva #define, seguida pelo identificador da constante e pelo seu respectivo valor, todos separados por um espaço em branco. #define <identificador><valor> Observe que não é acrescentado o ';' no final. Geralmente, para diferenciar as constantes no código, opta-se por escrever seus nomes com todas as letras maiúsculas.
Exemplos:

```
#define TRUE 1
#define FALSE 0 //agora, a palavra FALSE esta vinculada ao valor 0
```

8 EXPRESSÕES

As variáveis e constantes podem ser combinadas com os operadores associados a cada tipo de dados, gerando expressões.

8.1 Expressões Aritméticas

A expressão aritmética é aquela cujos operadores e operandos são de tipos numéricos (int ou float, por exemplo). Nas expressões aritméticas, é guardada sempre a seguinte relação de prioridade:

1. Multiplicação, divisão e resto de divisão (*,/,%, respectivamente);
2. Adição e subtração (+,-, respectivamente).

Além das operações básicas citadas acima, é possível usar, nas expressões aritméticas, outras operações, bastante comuns na matemática, definidas dentro da linguagem na forma de funções. A sintaxe geral para o uso dessas funções é a seguinte: <nome da função>(<valor>);
Algumas das funções matemáticas existentes são: sin (seno de um ângulo em radianos), cos (cosseno), pow (potenciação), sqrt (raiz quadrada de um número) etc. Para saber mais, verifique no livro texto.
Para a utilização dessas funções é necessário a inclusão da biblioteca math.h, além de ser necessário compilar o código adicionando a tag '-lm'.
Exemplo:

```
gcc programa.c -o prog -lm
```

9 COMANDOS DE ENTRADA DE DADOS

O comando de entrada de dados serve para captar do usuário do programa um ou mais valores necessários para a execução das tarefas. Na verdade, o que se faz é ler os dados de uma fonte externa, normalmente do teclado, para depois processar os dados e obter uma saída. Por meio desse comando de leitura de dados, uma ou mais variáveis podem ter seus valores lidos durante a execução do programa. Essa característica permite criar programas mais flexíveis.
Na linguagem C, a sintaxe para o uso do comando é a seguinte:

```
1 scanf("<formato1><formato2> ... <formatoN>", &var1, &var2, ..., &varN);
```

Essa estrutura é dividida em duas partes distintas. A primeira, colocada entre aspas duplas, contém os formatos, os quais são relacionados diretamente com os tipos das variáveis a serem lidas. A segunda parte é uma lista dos nomes dessas variáveis, com uma relação direta entre a posição nessa lista e o respectivo formato descrito na primeira parte do comando. Conforme mostra a descrição da sintaxe, os nomes das variáveis devem ser precedidos pelo

caractere '&'.

Os tipos de formatos mais utilizados são:

- %c – lê um único caractere;
- %d – lê um inteiro;
- %lf – lê um número em ponto flutuante (real).

É importante lembrar que o comando scanf é oriundo da biblioteca stdio.h.
Exemplos:

```
double a, n1;
int b, n2;
char c = 'b';
scanf ("%d %lf",&b,&a);
scanf ("%d %lf",&n1,&n2); //observe que esta incorreto, pois %d e o formato para numeros
    inteiros sendo que n1 e do tipo float. Assim como para o n2
```

10 COMANDO DE SAÍDA DE DADOS

Até agora, os exemplos de códigos e os conceitos estudados não mostravam como os programas comunicavam seus resultados ao usuário no final do programa, as variáveis armazenavam os resultados, mas o usuário não tinha como acessá-los. De fato, esse é o papel do comando de saída de dados.
O comando de saída de dados serve para escrever mensagens e exibir valores de expressões, proporcionando mais informação e legibilidade tanto para o usuário quanto para o próprio programador.
Na linguagem C, a sintaxe para o uso do comando de saída é mostrada a seguir.

```
printf("<formato1><formato2> ... <formatoN>", exp1, exp2, ..., expN);
```

Os formatos são os mesmos utilizados no scanf. Além disso, há um especficador '\n' que pula uma linha, evitando assim que o especificador do terminal e a saída do programa fiquem na mesma linha.
Exemplos de uso do printf:

```
double altura, peso;
printf ("Digite sua altura e peso:");
scanf ("%lf %lf",&altura", &peso);
printf ("Sua altura e peso sao: %f m %f kg\n",altura,peso);
```

Para especificar limites para casas decimais em números reais, basta acrescentar entre o % e lf um ponto (.), mais a quantidade de casas decimais que você deseja utilizar.
Exemplo:

```
double altura, peso; // supondo a leitura de 2.1234 e 4.322123
scanf ("%lf %lf",&altura, &peso);
printf ("Sua altura e peso sao: %.2f m %.4f kg\n",altura,peso); //saida: 2.12 e 4.3221
```

11 COMPILANDO E EXECUTANDO

O compilador é um programa que transforma o código-fonte escrito em linguagem de alto nível, para um executável em linguagem de máquina.
No sistema operacional Linux, para compilar um código-fonte utilize os seguintes comandos:

```
gcc <codigo_fonte>.c ou gcc  <codigo_fonte> -o <nome_do_executavel>
```

Lembrando que utilizando o primeiro comando, o compilador irá criar um executável com o nome padrão a.out. Caso queira dar um nome para o executável, utilize o comando com o parâmetro -o <nome_do_executavel>.
Para executar, basta utilizar o seguinte comando:

```
./<nome_do_executavel>
```

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) Faça um programa que leia 2 números inteiros do teclado, e imprima a soma deles.

```c
#include <stdio.h>
int main()
{
  int soma, num1, num2; //declaracao das variaveis
  //utilize a funcao scanf para ler os numeros do teclado
  //some os numeros lidos do teclado
  //imprima o resultado da soma com o comando printf
  return 0;
}

```

4. Crie um programa em C que leia duas variaveis inteiras do teclado e imprima o conteudo delas.

5. Passe para C os algoritmos a seguir:

   (a) Algoritmo 1

```
inicio {Algoritmo para o calculo da Media Final }
real : P1,P2,P3,MF;
imprima("Entre com o valor das Parciais P1, P2, P3");
leia (P1,P2,P3);
MF <- (P1+P2+P3)/3.0;
imprima ("A Media Final e: ", MF);
fim
```

   (b) Algoritmo 2

```
inicio {Algoritmo para o calculo da area de um retangulo}
real : AREA, L1, L2;
imprima("Entre com o valor da Altura e da Largura");
leia (L1, L2);
AREA <- L1*L2;
imprima ("A area do retangulo de lados L1 e L2 e: AREA");
fim
```

   (c) Algoritmo 3

```
inicio {Algoritmo para a troca dos valores de duas variaveis}
inteiro : A, B, AUX;
imprima("Entre com o valor de A e de B");
leia (A, B);
AUX <- A;
A <- B;
B <- AUX;
imprima ("A e B");
fim
```

6. Sobre o seguinte codigo mostrado abaixo, aponte sete erros identificando a linha e qual erro em específico foi cometido.

```c
#include <std.h>
#define PI 3.14;
int main()
{
    double Area,raio;
    printf("Entre com o valor do raio da circunferencia:);
    scanf("%f",raio);
    area=pi*raio*raio;
    printf("A area da circunferencia e igual a: %d\n",area);
}
```

Empreendendo

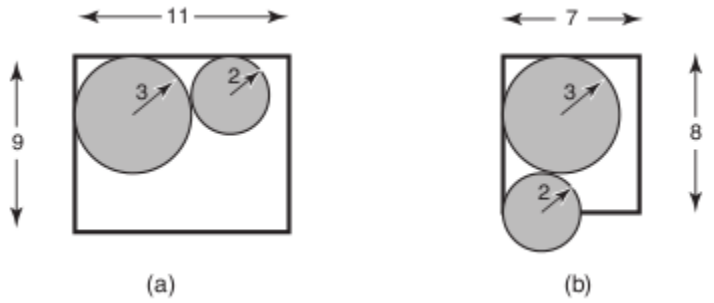
No mundo da tecnologia o que você não pode é ficar parado, ocioso no tempo. Você deve correr atrás, buscar a informação, o conhecimento. Empreender é isso, é correr atrás. Não é fundar uma empresa, ser um grande empresário e blablabla. É simplesmente correr atrás dos seus planos e sonhos e não deixar o tempo te pegar. Ás vezes, ao nosso redor talvez não tenhamos exemplos de pessoas que são assim, querendo e buscando sempre mais do que é possível. Mas o que importa mesmo é explorarmos nossa capacidade e fazer aquilo que gostamos. Se você está gostando de programar, de pensar algoritmicamente, se você não vê a hora de produzir tecnologia, sei lá, um aplicativo de celular, uma solução na web, vocêestá no caminho certo. Mas, entenda, em nada você terá sucesso sem esforço, sem ultrapassar barreiras insuportáveis, desde que essas barreiras façam parte dos seus objetivos. O Introcomp está aí pra isso, testar sua capacidade, sua vontade de fazer a diferença. Faça os praticando, curta os Workings, pergunte suas dúvidas, faça mais do que é pedido, busque mais na internet sobre programação de computadores, aplicações etc. Não é proibido que você saiba o que á uma struct ou até mesmo uma função recursiva antes de abordarmos estes tópicos com você (;D). Vá além! Os desafiso a seguir é proposto para que você possa ir além de seus limites, e desenvolver ainda mais sua capacidade de programação em C! Utilize todos os conceitos aprendidos em sala de aula para a compreensão do enunciado destes desafios, e preste muita atenção nos dados de entrada e saída. Explique o raciocínio utilizado para resolver o problema, e comente o código. Não esqueça de utilizar as boas práticas de programação, como indentação do código fonte, e atribuição de nomes de variáveis coerentes com o programa a ser desenvolvido.

Desafios

1. Elabore um programa, usando os comandos printf e scanf, que leia os valores do peso e da altura de uma pessoa, e ao final exiba na tela o valor do Índice de Massa Corporal (IMC) dessa pessoa. Observações:
   - Considere que o peso e a altura sejam dados em quilograma (kg) e metros (m), respectivamente.
   - IMC = peso / altura*altura;
   - Atente-se ao tipo de dado necessário das variáveis.
2. Escreva um programa em C que leia uma temperatura em graus celsius e imprima na tela a temperatura equivalente em farenheit. Mais uma vez, atente-se pelo tipo de dados

necessário para o problema.
Dado: F = 9C/5+32;

3. **Dica:** Pesquise sobre estruturas condicionais e sobre o comando return.
A FCC (Fábrica de Cilindros de Carbono) fabrica vários tipos de cilindros de carbono. A FCC está instalada no décimo andar de um prédio, e utiliza os vários elevadores do préio para transportar os cilindros. Por questão de segurança, os cilindros devem ser transportados na posição vertical; como são pesados, no máximo dois cilindros podem ser transportados em uma única viagem de elevador. Os elevadores têm formato de paralelepípedo e sempre têm altura maior que a altura dos cilindros.
Para minimizar o número de viagens de elevador para transportar os cilindros, a FCC quer, sempre que possível, colocar dois cilindros no elevador. A figura abaixo ilustra, esquematicamente (vista superior), um caso em que isto é possíbel (a), e um caso em que isto não é possível (b):



Como existe uma quantidade muito grande de elevadores e de tipos de cilindros, a FCC quer que você escreva um programa que, dadas as dimensões do elevador e dos dois cilindros, determine se é possível colocar os dois cilindros no elevador. **Entrada**
A entrada contém apenas um caso de teste. A primeira e única linha de cada caso de teste contém quatro números inteiros L, C, R1 e R2, separados por espaços em branco, indicando respectivamente a largura do elevador (1 <= L <= 100), o comprimento do elevador (1 <= C <= 100), e os raios dos cilindros (1 <= R1, R2 <= 100). Todos os dados de entrada são do tipo inteiro. Para o caso de teste, o seu programa deve imprimir uma única linha com um único caractere: 'S' se for possível colocar os dois cilindros no elevador e 'N' caso contrário. **Exemplo**

```
Entrada:        Saida:
11 9 2 3          S
7 8 3 2           N
10 15 3 7         N
8 9 3 2           S

```