Sobre a Linguagem

- Surgiu na década de 70, criado por Dennis Ritchie.
- Uma das suas vantagens é possuir características tanto de "alto nível" quanto de "baixo nível".
- Muitos programas, ainda hoje, são desenvolvidos em C.
- O C é uma linguagem ESTRUTURADA!

O C é “case sensitive”!

Isso quer dizer que a linguagem C, diferencia letras maiúsculas e minúsculas, tanto para nome de funções, variáveis e comandos da linguagem; ou seja, ao declararmos as seguintes variáveis:

int Soma, SOMA, SoMa, soma;

Todas as variáveis acima serão diferentes para o C.

Isto também se aplica aos comandos de sintaxe do C, como, por exemplo, o “**for**” e “**if**” que se forem escritos em maiúsculas o compilador não poderá interpretá-las corretamente.

Palavras Reservadas

- Toda linguagem de programação possuí "palavras reservadas".
- Palavras reservadas não podem ser utilizadas a não ser por seus propósitos originais.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	fot	signed	void
default	goto	sizeof	volatile
do	if	static	while

Estrutura de um programa em C

- Um programa simples em C tem as seguintes declarações:

main() função de inicio de programa em C. Só' pode existe uma no programa.

{ chave de abertura indicando inicio da função

funções;

} chave de fechamento encerrando a função

- A função main() marca o ponto de inicio da execução do programa.
- Os parênteses apos o nome_da_funcao, no caso main(), indica que ela é uma função.
- O corpo de função em C é sempre definido entre um abre chave "{" e um fecha chave "}“ que definem o inicio e final da função.
- Cada linha de programa na linguagem C é denominada de "declaração".
- Toda declaração deves ser encerrada um ponto e virgula **";“**

Comentários

Como já falamos de boas práticas, seria impossível não falar de "comentários em C". Comentários são blocos de texto que são ignorados pelos compiladores.

No C teremos duas formas de comentários, são elas:

```
// Comentário de uma linha

/*
 Bloco de comentário
*/
```

Identação

Observe o bloco de código abaixo:

```
int main(){int check=1;if(check){printf("Hello
world!\n");}else{printf(“Goodbye world!\n");return 0;}
```

O código acima não está identado. Note como está complicado de ler, apesar de ser um código extremamente simples.

Identar um código nada mais é que separar os códigos em blocos através de tabulação.

Tipos de Dados

- A linguagem C possuí 5 (cinco) tipos básicos de dados: **char**, **int**, **float**, **void** e **double**.

- Para cada tipo de dado existem modificadores de tipo, estes são 4 (quatro): **signed**, **unsigned, long** e **short**.

- Lembre-se, para o **float** nenhum modificado pode ser aplicado; assim como para o **double** podemos aplicar apenas o **long**.

Tipos de Dados

Tipo	Número de bits	Formato de leitura com scanf	Intervalo	
			Inicio	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
signed char	8	%c	-128	127
int	16	%i	-32.768	32.767
unsigned int	16	%u	0	65.535
signed int	16	%i	-32.768	32.767
short int	16	%hi	-32.768	32.767
unsigned short int	16	%hu	0	65.535
signed short int	16	%hi	-32.768	32.767
long int	32	%li	-2.147.483.648	2.147.483.647
signed long int	32	%li	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
float	32	%f	3,4E-38	3,4E+38
double	64	%lf	1,7E-308	1,7E+308
long double	80	%Lf	3,4E-4932	3,4E+4932

Tipos de Dados

- Declaração de variável:

```
tipo_da_variavel nome_da_variavel = valor_inicial_da_variavel;
```

- Declaração de variáveis de um mesmo tipo:

```
tipo_da_variavel nome_var1 = valor1, nome_var2 = valor2;
```

Operadores

- Realizam funções aritméticas e lógicas.
- Possuem, como na matemática, regras de precedência.
- Podem ser classificados em 3 (três) categorias.

Aritméticos e de Atribuição

Operador	Ação
+	Soma (inteiro e ponto flutuante)
-	Subtração ou troca de sinal (inteiro e ponto flutuante)
*	Multiplicação (inteiro e ponto flutuante)
/	Divisão (inteiro e ponto flutuante)
%	Resto da divisão (inteiros)
++	Incremento (inteiros e ponto flutuante)
--	Decremento (inteiro e ponto flutuante)

Expressões

- São combinações de variáveis, constantes e operadores.

- Devemos levar em consideração a tabela de precedência ao montá-las.

Exemplos de expressões:

```
Anos = Dias / 365.25;
i = i + 3;
c = a * b + b / e;
c = a * (b + d) / e;
```

Operadores Relacionais e Lógicos

Operador	Ação
>	Maior do que
>=	Maior ou igual a
<	Menor do que
<=	Menor ou igual a
==	Igual a
!=	Diferente de
&&	AND (E)
||	OR (OU)
!	NOT (NÃO)

Tabela de Precedência

Maior precedência
() [] ->
! ~ ++ -- . -(unário)
(cast) *(unário)
&(unário) sizeof
* / %
+ -
<< >>
<<= >>=
== !=
&
^
|
&&
||
?
= += -= *= /=
,
Menor precedência

Entrada

Sempre que falamos de entradas de dados, devemos considerar que essas entradas podem ocorrer de diversas formas, as mais comuns:

- Dados via teclado
- Dados recebidos através de scanners (leitores de código de barra)

Trabalharemos aqui apenas com dados recebidos através do teclado, para isso precisamos conhecer as funções básicas de entrada que o C nos fornece.

getch()

- É parte da biblioteca **conio.h**
- Utilizado para receber um único caractere
- Esta é uma função exclusiva para Windows

Formato:

```
variável_de_recebimento = getch();
```

getch()

Exemplo:

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

int main() {

    char Ch;
    Ch = getch();

    printf("Voce pressionou a tecla: %c", Ch);

    return 0;
}
```

scanf()

- É parte da **stdio.h**
- Utilizado para receber **string**s.
- É multiplataforma

Formato:

```
scanf(string_de_controle, lista_de_argumentos);
```

Comando scanf

- Este comando é utilizado para a leitura de caracteres formatados da entrada padrão. (no caso teclado)
- A forma geral da função scanf() e´:

scanf("<serie de controle>", lista de argumentos);

- A <serie de controle> usa-se apenas os códigos de formatação que especifica o tipo de informação definida na lista de argumentos(tipo da variável).
- Os códigos de formatação são os seguintes:

%c - leia um único caracter
%d - leia um inteiro
%f - leia um numero em ponto flutuante
%e - exibe um numero em notação cientifica
%g - utiliza o mais curto de %e ou %f,
%x - leia um inteiro hexadecimal
%o - leia um inteiro octal

- OBS.:O dado deve ser armazenada no endereço de memória que o computador alocou para a variável declarada. Por isso o operador de endereço **&** é utilizado nesta função, fazendo com que o valor digitado pelo usuário seja colocado no endereço correto de memória onde a variável foi criada pelo computador.

scanf()

Exemplo:

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

int main() {

   char Ch;
   scanf("%c", &Ch);

   printf("Voce pressionou a tecla: %c", Ch);

   return 0;
}
```

Comando printf

- Utiliza-se a função de biblioteca **printf**.

  printf() -> escreve caracteres formatados na saída padrão.(no caso vídeo)

- Formato Geral da função printf()

  printf("<serie de controle>", lista de argumentos);

- A <serie de controle> contem, tanto caracteres(letras, números, símbolos etc.) que serão impressos na tela, juntamente com os códigos de formatação que especificam como apresentar o conteúdo definido pelo tipo de informação declarada na lista de argumentos(tipo da variável).

- Os códigos de formatação são os seguintes:

  %c - exibe um único caracter
  %d - exibe um inteiro
  %f - exibe um número real em ponto flutuante
  %e - exibe um número real em ponto flutuante na notação científica
  %g - utiliza o mais curto de %e ou %f,
  %x - exibe um numero em notação hexadecimal
  %o - exibe um numero em notação octal
  %% - exibe um sinal de %

Comando if

O comando "if" representa uma tomada de decisão

- Avalia-se uma "expressão" para tomada de decisão e se a expressão é verdadeira executa a "declaração", se falsa não executa a "declaração".
- O corpo do "if" pode ter uma única declaração ou varias declarações entre um abre chave "{" e um fecha chave "}".

O formato geral quando se tem uma única declaração e':

```
if (expressão)

    declaração;
```

O formato geral quando se tem mais de uma declaração e':

```
if (expressão)
    {
    declaração;
    declaração;
    }
```

Observe que aqui tem que se usar o abre chave "{" e o fecha chave "}" para agrupar as declarações. Isto vale para todos os comando da linguagem C.

Comando If else

- Este comando "if-else" é uma expansão do comando "if".
- No comando "if" só' se executa a declaração ou declarações se a expressão for verdadeira(== 1).
- O comando "else" permite executar outra declaração ou declarações se a expressão for falsa(== 0).

O formato geral quando se tem uma única declaração é:

```
if (expressão)
    declaração 1;
 else
    declaração 2;
```

O formato geral quando se tem mais de uma declaração é:

```
if (expressão)
    {
    declaração 1;
    declaração 3;
    }
 else
    {
    declaração 2;
    declaração 4;
    }
```

Comando do - while

- Neste comando a avaliação da "condição lógica" que acontece após a execução da "declaração" ou "declarações“ pelo menos uma vez, pois a "condição" está no final do loop(apos o while).
- A "declaração" ou "declarações" são executadas pelo menos uma vez e depois é avaliada a "condição lógica". Se a "condição" for verdadeira (==1) a "declaração" ou "declarações" são executadas novamente.
- Se a "condição lógica" for falsa(==0) a execução do programa continua na "declaração" seguinte ao loop.
- Para este comando e' necessário sempre se utiliza o abre chave "{" e o fecha chave "}" entre o "do" e o "while".

O formato geral que é igual tanto para uma como para mais declarações e':

```
do
        {
        declaração 1;
        declaração 2;
        }while (condição);
```

Comando while

- Usa-se este comando quando queremos que uma tarefa seja executada enquanto a "condição lógica" for verdadeira(==1).
- Quando esta é falsa (==0) executa-se "declaração" seguinte após o laço.
- O teste da "condição lógica" é executado antes de entrar no laço.
- Portanto se a condição for falsa(==0) no primeiro teste a "declaração" ou "declarações" que estão dentro do laço não serão executadas nenhuma vez.
- O formato geral quando se tem uma única declaração é:

```
while (condicao)
        declaracao;
```

- O formato geral quando se tem mais de uma declaração e':

```
while (condicao)
        {
        declaracao;
        declaracao;
        }
```

Exemplos

Exemplo 1

Este programa tem como objetivo receber dois números quaisquer e exibir a soma entre eles.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b, soma;
    printf("Este programa tem como objetivo  receber dois números quaisquer e exibir a soma entre eles.\n");
    printf("Digite  o primeiro número \n");
    scanf("%d", &a);
    printf("Digite  o segundo número \n");
    scanf("%d", &b);
    soma=a+b;
    printf("O resultado da soma:   %d \n",soma );
    printf("Pressione uma tecla para finalizar \n");
    getch();
}
```

Exemplo 2

```
#include <stdio.h>
main()
{
   //VARIÁVEIS COM LETRAS MAIÚSCULAS
   float NUM1,NUM2=0,D;

   printf("COMANDO LIMPA TELA");
   system("cls");
   printf("Este programa tem como objetivo calcular e exibir a divisão \n");
   printf("do primeiro pelo segundo numero. \n");
   printf("Digite  um numero qualquer. \n");
   scanf("%f",&NUM1);
   while (NUM2==0)
      {
      printf("Digite um numero diferente de 0. \n");
      scanf("%f",&NUM2);
      if (NUM2 == 0)
         printf("Numero invalido. \n");

   }
   D=NUM1/NUM2;
   printf("Usando \t para exibir todos os números que compoem a variavel real D \n");
   printf("O resultado da divisão  %f /%f =%f\t. \n",NUM1,NUM2,D);
}
```

Exemplo 3

```
#include <stdio.h>
main()
{
    int a=0,b=0,s;
    printf("Receba dois numeros.\n");
    printf("O primeiro numero deverá ser maior que zero e menor que dez.\n");
    printf("O segundo deve ser maior que 5 e menor ou igual a 15.\n");
    printf("Exiba a soma entre eles observando as seguines condições:\n");
    printf("Se a soma for inferior a 15 soma 50, caso contrario some 10.\n");
    printf("\n");
    printf("\n");
   while (a<=0 || a>=10)
    {
        printf("Digite um numero maior que zero e menor que dez.\n");
        scanf("%d",&a);
        if ((a<=0) ||(a>=10))
            printf("quantidade inválida.\n ");

    }
    while (b<=5 || b>15)
    {
        printf("Digite um numero maior que cinco e menor ou igual a quinze.\n");
        scanf("%d",&b);
        if ((b<=5) || (b>15))
            printf("quantidade inválida.\n ");

    }
    s=a+b;
    if (s<15)
    {
        printf("A soma foi inferior a 15. \n");
        s=s+50;
    }
    else
    {
        printf("A soma foi superior a 15. \n");
        s=s+10;
    }
    printf("A soma é %d : \n",s);
}
```

Exemplo 4

```
#include <stdio.h>
main()
{
   float num1,num2,d;
//COMANDO LIMPA TELA
   system("cls");
   printf("Este programa tem como objetivo calcular e exibir a divisão \n");
   printf("do primeiro pelo segundo numero. \n");
   printf("Digite  um numero qualquer. \n");
   scanf("%f",&num1);
   do
   {
      printf("Digite um numero diferente de 0. \n");
      scanf("%f",&num2);
      if (num2 == 0)
         printf("Numero invalido. \n");

   }
   while (num2==0);
   d=num1/num2;
   printf("O resultado da divisão  %f /%f =%f. \n",num1,num2,d);
}
```

Bibliografia

- http://linguagemc.com.br/string-em-c-vetor-de-caracteres/
- http://mtm.ufsc.br/~azeredo/cursoC/aulas/c530.html
- http://www.cprogressivo.net/2013/03/Lendro-e-Escrevendo-Strings-em-C.html
- http://www.inf.pucrs.br/~pinho/LaproI/Funcoes/AulaDeFuncoes.htm