Faculdade de Computação

Programação Procedimental

1° Laboratório de Programação C

Prof. Cláudio C. Rodrigues

1. Introdução

O objetivo desta aula prática é exercitar o uso de variáveis de vários tipos e iniciar o estudo de comandos de entrada e saída simples a partir do console e expressões aritméticas. Lembre-se que os seus programas devem começar sempre com os comentários mostrados em 1.

Programa 1: Padrão para programas 1.

```
/*
Programa : NomeDoArquivoFonte.c
Autor: Aluno Programador
Data : dd/mm/aaaa
Descrição : Este programa faz algo muito importante.
*/
#include <stdio.h>
int main (void)
{
/* declaracoes das variaveis */

/* comandos do programa */
}
```

---

2. Declaração de variáveis

Todas as variáveis em C devem ser definidas (declaradas) antes de serem usadas. Esta definição deve ser dada no início do programa, segundo o padrão ANSI.

Os dados em C podem assumir cinco tipos básicos que são os seguintes:

tipo	descrição
char	O valor armazenado é um caractere
int	O valor armazenado é um número inteiro.
float	Número em ponto flutuante de precisão simples, normalmente 32 bits. São conhecidos como números reais.
double	Número em ponto flutuante de precisão dupla, com isto a precisão e as vezes a excursão dos números aumenta. Este tipo é armazenado em 64 bits.
void	Este tipo serve para indicar que um resultado não tem um tipo definido. Uma das aplicações deste tipo em C é criar um tipo vazio que pode posteriormente ser modificado para um dos tipos anteriores.

P1) Escreva o programa 2 e verifique o seu funcionamento. Observe que todas as variáveis foram definidas logo no início do programa.

```
Programa 2: Definição de variáveis 1.
/*
Programa : definicao.c
Autor: Aluno Programador
Data : dd/mm/ aaaa
Descrição : Este programa mostra exemplo de definicao de variaveis .
*/
```

```
#include <stdio.h>
int main (void)
{
    int i;
    float r;
    double dr;
    char c;
    i = 10;
    r = 1.0;
    dr = 3e3;
    c = ‘a’;
    printf ("i = %d\n", i);
    printf ("r = %f\n", r);
    printf ("df = %f\n", dr);
    printf ("c = %c\n", c);
    return 0;
}
```

P2) Determine o valor das seguintes expressões. Assuma a=7, b=5, c=9, d=2 e e=1.

- **a)** d % b == c % b
- **b)** a * c != d * b
- **c)** d * b == c * e
- **d)** !(a * b)
- **e)** !(a % b * c)
- **f)** !(c % b * a)
- **g)** b % c * a

P3) Usando parênteses, reescreva as seguintes operações para indicar corretamente a ordem de avaliação. Após, calcule o valor de cada expressão assumindo a=3, b=5 e c=1.

- **a)** a % b * c && c % b * a
- **b)** a % b * c || c % b * a
- **c)** b % c * a && a % c * b
- **d)** b % c * a || a % c * b

---

3. A Função printf

A função ***printf*** faz com que dados sejam escritos no dispositivo de saída padrão, que normalmente é a tela do computador.
O protótipo da função é:

int printf("formato", var1, var2, ...);

onde os argumentos var1, var2, ... são impressos de acordo com o formato indicado pela cadeia de caracteres que compõe formato.

Um exemplo simples pode tornar a explicação mais clara. O programa 3 imprime o valor das variáveis dia, mes, ano. Detalhando o comando printf temos primeiro entre aspas o seguinte texto:
Estamos na data: dia = %d, mes = %d e ano = %d.
Este texto contém as indicações do que e como deve ser impresso. Tudo que tiver precedido pelo caractere % é indicação do formato de saída dos dados. O resto vai direto para a saída. Por exemplo, neste texto %d indica que uma variável inteira vai ser impressa. Como temos três destes formatos devemos ter de imprimir três variáveis. Observe então que o comando printf termina com a lista das variáveis a serem impressas.

P4) Escreva o programa 3 e verifique o seu funcionamento.
A tabela 1 mostra os códigos usados para saída de dados, note que para entrada de dados usam-se códigos um pouco diferentes.

```
Programa 3: Exemplo de impressão de resultados
#include <stdio.h>
int main (void)
{
    int ano = 1997 , dia = 29, mes = 12;
    /* Imprime o valor do ano */
    printf (" Estamos na data : dia = %d, mes = %d e ano = %d\n", dia
    , mes , ano );
    return 0;
}
```

Código	Comentário
%c	Caractere simples
%s	Cadeia Caracteres
%d ou %i	Inteiro (int) decimal com sinal
%ld ou %li	Inteiro (long int) decimal com sinal
%u	Inteiro decimal sem sinal
%f	Real em ponto flutuante
%Lf	Real em ponto flutuante formato longo (long double)
%E ou %e	Real (double) em notação científica com E ou e
%LE ou %Le	Real (long double) em notação científica com E ou e
%G	%E ou %f, o que for mais curto
%g	%g ou %f, o que for mais curto
%o	Inteiro em base octal
%x	Inteiro em base hexadecimal (letras minúsculas)
%X	Inteiro em base hexadecimal (letras maiúsculas)
%p	Endereço de memória
%%	Imprime o caractere %

Tabela 1: Códigos de Conversão para escrita de dados.

P5) Escreva um programa que defina uma variável do tipo inteiro, atribua um valor qualquer e imprima este valor na base ***decimal***, ***octal*** e ***hexadecimal***.
Dica: Procure na Tabela 1 o formato para impressão de valores em ***octal*** e ***hexadecimal***.

P6) Curiosidade: C permite que seja impresso o código inteiro usado pelo computador para representar um caractere. O programa 4 mostra um exemplo simples de impressão de caractere e seu código.
Digite este programa e verifique o resultado. Experimente com outros caracteres.

```
Programa 4: Exemplo de impressão codigos de caracteres
/*
Programa : codigo.c
Autor: Aluno Programador Brilhante
Data : dd/mm/aaaa
Descrição : Este programa imprime o codigo inteiro de um caractere
*/
#include <stdio.h>
int main (void)
{
    char c = ‘a’;
    printf("Caractere = %c, codigo inteiro do caractere = %d\n", c,
    c);
    return 0;
}
```

P7) Escreva um programa que defina variáveis dos tipos float, double e long double, atribua valores a estas variáveis e as imprima usando o ***formato normal*** e o de ***notação científica*** (Programa 5). Procure estes formatos na Tabela 2.

```
Programa 5: Exemplo de definicao de variaveis.
#include <stdio.h>
int main (void)
{
    float f;
    double d;
    long double ld;
    return 0;
}
```

P8) Um número **n** na faixa de **100** até **999** é um número de *Angstron* se o número **n** obedece a regra.

$$n = centenas^3 + dezenas^3 + unidades^3$$

Por exemplo, o número 153 é obedece a regra pois
$153 = 1^3 + 5^3 + 3^3$

Escreva um programa que defina uma variável inteira **n** ($100 \leq n \leq 999$), atribua um valor qualquer do intervalo, o decomponha em centenas, dezenas e unidades e imprima estes números, o resultado da fórmula acima e a mensagem se **n** é ou não um número ***Angstron***.

Dica importante: Em C o operador % fornece o resto da divisão de dois operandos inteiros. Considere, por exemplo, as variáveis inteiras int a, b, c. A expressão c = a % b; armazena em c o resto da divisão de a por b.
Exemplo de saída:
Número: 153
Centenas : 1
Dezenas : 5
Unidades : 3
Angstron : 153

4. A função scanf

Como escrevemos antes os formatos de entrada e saída de dados são um pouco diferentes. A tabela 2 mostra os códigos usados para entrada de dados.

Código	Comentário
%c	Caracter simples
%s	Cadeia Caracteres
%d ou %i	Inteiro (int) decimal com sinal
%ld ou %li	Inteiro (long int) decimal com sinal
%u	Inteiro decimal sem sinal
%f	Real em ponto flutuante
%lf	Real em ponto flutuante formato double
%Lf	Real em ponto flutuante formato long double
%E ou %e	Real (float) em notação científica com E ou e
%lE ou %le	Real (double) em notação científica com E ou e
%LE ou %Le	Real (long double) em notação científica com E ou e
%G	%E ou %f, o que for mais curto
%g	%g ou %f, o que for mais curto
%o	Inteiro em base octal
%x	Inteiro em base hexadecimal (letras minúsculas)
%X	Inteiro em base hexadecimal (letras maiúsculas)
%p	Endereço de memória
%%	Imprime o caractere %

Tabela 2: Códigos de Conversão para entrada de dados.

P9) Escreva um programa que calcule a nota de um aluno de uma disciplina cuja fórmula é a seguinte:

$$notaFinal = 0.8 \times prova + 0.2 \times \frac{\sum_{i=1}^{n} teste_i}{n} \quad (1)$$

Considere que o número de testes é igual a 3.
Exemplos de saída:
Prova: 8.0
Teste 1: 8.0
Teste 2: 10.0
Teste 3: 3.0
Nota final 7.8

Programa 6 mostra como este programa poderia ser escrito.

```
Programa 6: Exemplo de leitura de dados.
#include <stdio.h>
int main (void)
{
    float prova;
    float teste1 , teste2 , teste3;
    float notaFinal ;
    printf (" Prova: ");
    scanf("%f", & prova );
    printf (" Teste 1: ");
    scanf("%f", & teste1 );
```

```
    printf (" Teste 2: ");
    scanf("%f", & teste2 );
    printf (" Teste 3: ");
    scanf("%f", & teste3 );
    notaFinal = 0.8 * prova + 0.2 * ( teste1 + teste2 + teste3 ) / 3 ;
    printf ("Nota final %0.2 f\n", notaFinal );
    return 0;
}
```

P10) Acrescente ao programa anterior a impressão do resultado da seguinte expressão booleana **`notaFinal ≥ 5.0`** (use o formato inteiro para imprimir este resultado). Utilize o seguinte comando:

```
printf("%d\n", notaFinal >= 5.0);
```

Verifique o que é impresso quando a **`notaFinal`** é maior do que 5.0 e quando é menor.

P11) É possível na linguagem C usar caracteres em expressões aritméticas. Por exemplo, é possível escrever o seguinte código:

```
char c = 'a';
c = c + 1;
printf("%c\n", c); /* imprime o caractere seguinte ao 'a' */
```

Escreva um programa que defina uma variável do tipo caractere, atribua um caractere qualquer a variável e imprima o caractere seguinte e o anterior. Verifique qual é o caractere anterior ao 'a' e qual é o seguinte ao 'z'. Faça o mesmo para letras maiúsculas.

P12) ***Desafio***: Escreva, compile e execute o programa apresentado no *Programa 7*. Faça uma análise do código e gere uma sequência de operações que permita a execução da mensagem "Sucesso! Você conseguiu!" Caso tenha alcançado êxito no desafio, explique ou justifique os resultados obtidos:

```
Programa 7: Definição de variáveis 1.
/* Descrição : Este programa explora bugs na representação inteira. */
#include <stdio.h>
void printInt(unsigned int i){
    if(i > 100) {
        printf("Sucesso ! Voce conseguiu!\nValor do Int : %u",i);
        return (0);
    }
}

int main() {
    int i=0;

    scanf("%d",&i);

    if(i > 100) {
        return(-1);
    }
    printInt(i);
    return (0);
}
```

P13) Considere a ***massa*** da terra (**M**) igual a **5,9 x 10^{24} kg** e um satélite de massa **m** em órbita circular de raio **R** em torno da terra. Escreva um programa que leia do dispositivo padrão de entrada (teclado) o valor do ***raio*** e, determine e escreva no dispositivo padrão de saída (tela) a velocidade escalar **V** do movimento orbital do satélite e o período **T** do movimento orbital.

$V = \sqrt[2]{\frac{(G * M)}{R}}$	$T = 2 \times Pi \times \sqrt[2]{\frac{R^3}{(G * M)}}$
G = 6,7*10^{-11} Nm2/kg2	M ? massa do planeta (kg)
R ? raio da órbita (metros)	T ? Período (segundos)
V ? Velocidade escalar (m/s)	

O algoritmo deve estar contido no arquivo "orbita.c"

P14) Escreva um programa em linguagem C que leia do dispositivo padrão de entrada (teclado) as coordenadas (x,y) de três pontos (P1, P2 e P3) no plano cartesiano. O programa deve calcular a distância entre os três pontos, que definem o comprimento dos lados (**a**, **b**, **c**) de um triângulo. Verificar se os comprimentos dos lados formam um triângulo.
Distância entre os pontos i e j:

$$\boldsymbol{d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}$$

O programa deve calcular a área do triangulo. A área de qualquer triângulo de lados **a**, **b** e **c** pode ser calculada pela fórmula abaixo:
$\boldsymbol{A = \sqrt{s(s-a)(s-b)(s-c)}}$, onde $\boldsymbol{s = (a+b+c)/2}$

O algoritmo deve estar contido no arquivo "triangulo.c"

P15) Realize as transformações de mudança de base de representação numérica

1. 1990_{10} → X_2
2. 10101010_2 → X_{10} , X_8 , X_{16}
3. $AB2C_{16}$ → X_{10} , X_8
4. 10011_2 → X_8 , X_{16}
5. $54,75_{10}$ → X_2
6. $F8,A_{16}$ → X_8
7. $110,111_2 + 728_{10}$ → X_{10}
8. $AF,4_{16} – 26_8$ → X_{10}
9. $270,1_{10} – 110_2$ → X_{16}
10. 100_2 x 14_{16} → X_{10}

Obs: para conferir o resultado, vocês podem fazer a conversão de volta.