Métodos e Técnicas de Programação
.:: Avaliação continuada P2 ::.

Prof. Igor Peretta

ENTREGA: até 19/set/2018

Contents

1 Introdução

1.1 Bases numéricas

Fontes: `http://www.dainf.cefetpr.br/~robson/prof/aulas/common/bases.htm`; `https://pt.wikipedia.org/wiki/Convers~ao_de_base_numérica`

Podemos considerar, a fim de simplificação, que base numérica é um conjunto de símbolos (ou algarismos) com o qual podemos representar uma certa quantidade ou número.

No dia a dia costuma-se utilizar a base dez, ou base decimal, que como o próprio nome já diz é composta por 10 algarismos diferentes: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.

Conversão de base numérica é o nome dado à passagem de um valor de uma base para outra mantendo o valor quantitativo, mas alterando a simbologia para se adequar à nova base.

As bases numéricas que serão exploradas aqui são:

- Decimal (10) {0,1,2,3,4,5,6,7,8,9};
- Binária (2) {0,1};
- Octal (8) {0,1,2,3,4,5,6,7};
- Hexadecimal (16) {0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f}.

Uma dada base N pode ser descrita como uma contagem modular ou contagem módulo N (operador `%` em C).

1.2 Usando o scanf

O termo *buffer* pode ser definido como uma memória temporária utilizada para escrita e leitura de dados. Quando digitamos alguma coisa no teclado do computador, os códigos binários referentes a todas as teclas digitadas são armazenados em um buffer de entrada (em C, identificado como `stdin`).

Quando utilizado o `scanf()` para captura de dados alfanuméricos, dependendo do tipo de dado capturado, a máquina de estados implementada em seu núcleo não consome os códigos armazenados no buffer que não compõe a informação esperado, por exemplo o código da tecla `ENTER`.

Portanto, após cada `scanf()` você precisa definir um tratamento para esvaziar o buffer de teclado. Teste o seguinte programa:

```
#include<stdio.h>
int main() {
    int i; char c;
    printf("Informe um inteiro\n");
    scanf("%d", &i);
    printf("Informe um caractere\n");
    scanf("%c", &c);
    printf("Voce informou: \"%d\" e \"%c\" "
   "(codigo %hhu)!\n", i, c, c);
    return 0;
}
```

Digite `42` e tecle `ENTER`; depois digite `A` e tecle `ENTER`. Você verá que a segunda informação foi perdida porque o segundo `scanf()` capturou como caractere o `ENTER` (código 10) que não foi consumido do buffer pelo primeiro `scanf()`.

Para adaptar essa característica do `scanf()` à lógica desejada pelo programa, basta consumir a tecla `ENTER` com `getchar()`, por exemplo, logo após cada `scanf()`.

Teste então o programa com as seguintes alterações:

```
#include<stdio.h>
int main() {
    int i; char c;
    printf("Informe um inteiro\n");
    scanf("%d", &i); getchar(); // <-------
    printf("Informe um caractere\n");
    scanf("%c", &c); getchar(); // <-------
    printf("Voce informou: \"%d\" e \"%c\" "
   "(codigo %hhu)!\n", i, c, c);
    return 0;
}
```

Digite `42` e tecle `ENTER`; depois digite `A` e tecle `ENTER`. Agora o programa se comporta como o esperado pelo usuário.

2 Programas a serem entregues

Os programas a serem entregues precisam seguir o nome da seção em que são descritos, não sendo aceitos programas com outros nomes.

2.1 P2.c

Implemente um programa de conversão entre bases numéricas com o seguinte menu de opções:

1. Binário para Decimal
2. Binário para Hexadecimal
3. Hexadecimal para Decimal
4. Hexadecimal para Binário
5. Decimal para Binário
6. Decimal para Hexadecimal
7. Octal para Decimal

8. Decimal para Octal

Em seguida, capture o valor na base numérica de origem e imprima na tela o resultado da conversão para a base numérica de destino.

2.1.1 Dicas

1. Tente usar as estratégias de captura que você usou no primeiro programa P1
2. Para a captura de bits, crie uma variável do tipo vetor de caracteres (*string*) para armazenar a sequência de bits (ex. `char bits[256];`) e use `scanf()` com o especificador próprio para *strings*; use o fato de que o último caractere válido de uma string é `'\0'` para criar o laço de análise (ex. `while(bits[i] != '\0') {}`)
3. Caso você esteja com dificuldades com alguns algoritmos de conversão, com excessão da base binária, o próprio `scanf()` e o `printf()` podem ser utilizados para capturar ou imprimir valores em bases numéricas distintas, com o auxílio dos especificadores `%d`, `%x` (`%X`) ou `%o` (use esse atalho apenas em último caso)
4. A **única** biblioteca que deverá ser usada é a `stdio.h`
5. Uma maneira mais simples de descobrir quantos bits são necessários (`nb`) para se representar um número (`numero`) em binário:

```
for(nb = 0; numero >= (1 << nb); nb++);
```

Na linha seguinte, a variável `nb` terá essa informação. Por que será? Tente descobrir, tem a ver com o operador `«` das operações *bitwise* em C.

2.1.2 Testes

- "1" + "001111011011" retorna "987"
- "2" + "001111011011" retorna "3db" ou "3DB"
- "3" + "4A0F" retorna "18959"
- "4" + "4A0F" retorna "100101000001111" (para 15 bits)
- "5" + "987" retorna "1111011011" (para 10 bits)

- "6" + "18959" retorna "4A0F"
- "7" + "761" retorna "497"
- "8" + "497" retorna "761"

3 Informações importantes

É necessário criar em sua conta do github um repositório com o nome 'MTP-2018-1'. É nesse repositório que você dar *upload* do(s) seu(s) código-fonte(s) (ex. arquivos `P1.c`, `P2.c` etc.), não sendo desejado nenhum executável ou arquivo de apoio de projetos.

Em todo programa que você fizer, comece com seu nome e matrícula como comentários. Se não constar essas informações nos arquivos enviados para seu repositório no Github, os programas serão **desconsiderados**.

Mantenha seu código limpo. Não use comandos como `system(pause)` ou `#include<conio.h>` pois são específicos do sistema operacional Windows. Se usá-los, seu código-fonte poderá não compilar, invalidando sua entrega.