Programação Estruturada Aula 11 - Geração de números aleatórios

Prof. Luis Carlos Retondaro

Técnico em Telecomunicações
2^o Ano

CEFET/RJ - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

Campus Petrópolis

2017

Sumário

Valores máximos e mínimos

- Um tipo-de-dados (*data type*) é um conjunto de valores munido de um conjunto de operações;
- Alguns tipos de dado possuem o valor máximo e mínimo que suas variáveis podem assumir;
- **Exemplo para o tipo int:**

```
INT_MIN, ..., -2, -1, 0, 1, 2, ..., INT_MAX
```

Valores máximos e mínimos

- Os valores das constante *INT_MIN* e *INT_MAX* estão definidos na biblioteca limits.h;
- Outras constantes definidas em limits.h:

```
//inteiros
 #define INT_MIN   // (-2147483648)
 #define INT_MAX    // (2147483647)

//unsigned int
 #define UINT_MAX  // (4294967295)

 //short
 #define SHRT_MIN       //(-32768)
 #define SHRT_MAX       // (32767)
```

Valores máximos e mínimos

- Outras constantes definidas em limits.h:

```
//long
 #define LONG_MIN   // (-2147483648L)
 #define LONG_MAX   // (2147483647L)


//double
 #define DBL_MIN    // (2.2250738585072014E-308)
 #define DBL_MAX    // (1.7976931348623157E+308)


 //float
 #define FLT_MIN    // (1.175494351E-38F)
 #define FLT_MAX    // (3.402823466E+38F)
```

Geração de números aleatórios

- Sequências de números aleatórios são úteis em muitas aplicações;
- Números verdadeiramente aleatórios são muito difíceis de obter; por isso, devemos nos contentar com números pseudo-aleatórios, gerados por algoritmos;
- Um gerador de números pseudo-aleatórios é um programa computacional que gera uma sequência de números que não apresentam padrão, ou seja, aparentam ser aleatórios;
- Em computação, aplicações que requerem aleatoriedade envolvem simulações de sistemas reais, criptografia, jogos e amostragem estatística.

Geração de números aleatórios

- A função *rand()* (abreviatura de random), é utilizada para gerar números aleatórios e é definida na biblioteca *stdlib.h*;
- A função rand() produz um número aleatório no intervalo $[0, RAND\_MAX]$. Sintaxe:

```
rand();
```

- A constante *RAND_MAX* está definida no arquivo-interface da biblioteca stdlib.h e é tipicamente menor que *INT_MAX*;

Geração de números aleatórios

Exemplo:

```
// RandomInt.c
// Gera números inteiros aleatórios.

#include <stdio.h>
#include <stdlib.h>
void main(){

  //armazena cada inteiro aleatório gerado
  int num;
  int i;

  for(i=1; i<=20; i++){
      num = rand(); // gera o num
      printf("%d \t",num);
      if(i % 5 == 0)
        printf("\n");
  } // fim for
} // fim main
```

Geração de números aleatórios

Resultado da execução

```
1804289383      846930886       1681692777      1714636915      1957747793
424238335       719885386       1649760492      596516649       1189641421
1025202362      1350490027      783368690       1102520059      2044897763
1967513926      1365180540      1540383426      304089172       1303455736
```

Geração de números aleatórios

Problema: Como gerar números aleatórios entre 1 e 10?

- Dado que a função *rand()* devolve sempre um número entre os limites $[0, RAND\_MAX]$ então é necessário fazer com que o limite seja compreendido entre 1 e 10;
- O número em um determinado intervalo pode ser obtido aplicando o operador de resto, %;

```
num = 1 + rand % 10;
```

Geração de números aleatórios

Problema: Como simular o lançamento de uma moeda?

```c
#include <stdio.h>

void main(){

    int r, i, qtddCara = 0, qtddCoroa = 0;

    //simula o lançamento de uma moeda 20 vezes
    for(i=0; i< 20; i++){
      r = rand() % 2;
      if(r){
        printf("cara\n");
        qtddCara++;
      }
      else{
        printf("coroa\n");
        qtddCoroa++;
      }
    }
    printf("\nquantidade de vezes que saiu cara %d\n",qtddCara);
    printf("quantidade de vezes que saiu coroa %d\n",qtddCoroa);

}
```

Geração de números aleatórios

Problema: Como simular o lançamento de uma moeda?

Resultado da execução

```
cara
coroa
cara
cara
cara
cara
coroa
coroa
cara
cara
coroa
cara
coroa
cara
cara
coroa
coroa
coroa
coroa
coroa

quantidade de vezes que saiu cara 10
quantidade de vezes que saiu coroa 10
```

Geração de números aleatórios

Problema: O conjunto de números aleatórios gerados são sempre igual. Como solucionar?

- Por padrão a função *rand*() parte da mesma semente para realizar o cálculo dos números aleatórios;
- Dado que a semente é sempre a mesma então a toda execução os números gerados são iguais (como se a moeda fosse viciada);
- Para contornar esse problema é necessário modificar a semente a cada execução do programa;
- Então usa-se, um valor que é gerado através do relógio do computador.

Geração de números aleatórios

```
void inic_random(){
  long ultime;
  time(&ultime);
  srand((unsigned) ultime);
}
```

- A função inic_random() é a função responsável por iniciar a semente dos números aleatórios;
- A variável ultime é inicializada com o número de segundos desde 01/01/1970 e esse valor é passado como parâmetro para a função srand().

Geração de números aleatórios

```
void inic_random(){
  long ultime;
  time(&ultime);
  srand((unsigned) ultime);
}
```

- Dado que a função srand() espera um unsigned int então a solução é fazer um casting para promoção da variável;
- Includes necessários $< stdio.h >$, $< stdlib.h >$ e $< time.h >$.

Geração de números aleatórios

Problema: Como simular o lançamento de uma moeda não viciada?

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

void inic_random(){
  long ultime;
  time(&ultime);
  srand((unsigned) ultime);
  srand(time(NULL));
}

void main(){
  int r, i, qtddCara = 0, qtddCoroa = 0;

  inic_random();
  //simula o lançamento de uma moeda 20 vezes
  for(i=0; i< 5; i++){
    r = rand() % 2;
    if(r){
      printf("cara\n");
      qtddCara++;
    }
  //continua...
```

Geração de números aleatórios

Problema: Como simular o lançamento de uma moeda não viciada?

```
//continua...

    else{
      printf("coroa\n");
      qtddCoroa++;
    }
  }

  printf("\nquantidade de vezes que saiu cara %d\n",qtddCara);
  printf("quantidade de vezes que saiu coroa %d\n",qtddCoroa);

}
```

Geração de números aleatórios

Resultado da execução:

```
 //Execução 1
coroa
cara
cara
cara
cara

quantidade de vezes que saiu cara 4
quantidade de vezes que saiu coroa 1
```

```
 //Execução 2
cara
cara
coroa
coroa
cara

quantidade de vezes que saiu cara 3
quantidade de vezes que saiu coroa 2
```

Geração de números aleatórios

Revisão!!!

Próxima aula P_1

Referências

1. C Completo e Total, Herbert Schidt; Pearson Makron Books; 3a. Ed., 1997.

2. Linguagem C, Luís Damas, LTC, 10a. Ed.2014.