10. Subrotinas

DIM0321

2015.1

Outline

Motivação

Observação

- Os programas escritos até então estão inseridos na função principal `main`
- Já usamos recursos externos (parte da biblioteca padrão / arquivos cabeçalhos)
- Recursos externos = componentes do código

Modularização

Numa aplicação real, tarefas repetitivas ou que podem ser reusadas são organizadas em módulos

- inverter um matriz
- ordenar um vetor (de nome, de número)

Sub-rotinas

Objetivo

- propõem a prestação de um serviço preciso
- são relativamente independentes do resto do programa

Estruturar o código

- Subrotinas encontram-se num ponto pré-definido do programa
- Podem ser chamadas a partir de vários pontos do programa
- Podem ser modificadas de maneira relativamente independente, desde que o serviço prestado continue o mesmo

Elas são uma maneira de introduzir uma separação lógica entre a implementação/algoritmo usado (o corpo da sub-rotina) e o resultado.

Elementos de sub-rotina

Partes

- Nome da subrotina
- Tipo de retorno
- Parâmetros
- Corpo
- Valor de retorno

Funções

- Em C, toda subrotina é uma função
- Procedimento ≈ função com tipo de retorno `void`

Declaração

Definição (Declaração)

- A interface = declaração de prestação de serviços
- Descreve como a sub-rotina deve ser chamada. Ela é **necessária**.

```
1 tipo_retorno nome_funcao(tipo1 par1, tipo2 par2, ..., tipon parn);
```

Exemplo

```
1 dummy(); // declaração mínima
2
3 double pow(double x, double y);
4 int printf(const char *format, ...);
```

Definição

Definição (Definição)

- A definição da sub-rotina é a etapa de **implementação**
- Definimos o corpo da sub-rotina
- Pode ocorrer em qualquer lugar do código apos a declaração

```
tipo_retorno nome_funcao(tipo1 par1, tipo2 par2, ..., tipon parn)
{
    // declaracoes e comandos
}
```

Elementos

- Repetição da declaração
- Variáveis locais
- Instruções de retorno (se necessário)

Exemplos

V1

```c
#include <stdio.h>

void print_int(int d)
{
     printf("Int: %d\n", d);
}

int square_int(int d);

int main(void)
{
     int n;
     printf("Enter an int: ");
     scanf("%d", &n);
     print_int(square_int(n));
     return 0;
}

int square_int(int d)
{
     int sq;
     sq = d * d;
     return sq;
}
```

V2

```c
#include <stdio.h>

void print_int(int d)
{
     printf("Int: %d\n", d);
}

int square_int(int d)
{
     return d * d;
}


int main(void)
{
     int n;
     printf("Enter an int: ");
     scanf("%d", &n);
     print_int(square_int(n));
     return 0;
}
```


Variáveis globais

- Até agora, só usamos variáveis definidas no início de um bloco ou dentro de um laço `for`.
- Podemos também compartilhar variáveis entre componentes, através variáveis globais declaradas **no mesmo nível** que as funções.

Getter/setter

```c
int glob;
void setglob(int v);
int getglob(void);

void setglob(int v) { glob = v; }
int getglob(void) { return glob; }
```

Aviso

- Não é aconselhado usar variáveis globais **sem necessidade**, pois elas quebram as separações entre componentes.

Exemplo (K & R)

```c
#define BUFSIZE 100

char buf[BUFSIZE];
int bufp = 0;

/* buffer for ungetch */
/* next free position in buf */
int getch(void) /* get a (possibly pushed-back) character */
{
     return (bufp > 0) ? buf[--bufp] : getchar();
}


void ungetch(int c)
/* push character back on input */
{
     if (bufp >= BUFSIZE)
  printf("ungetch: too many characters\n");
     else
  buf[bufp++] = c;
}
```

Escopo

Definição

O **escopo** de um nome é a parte do código na qual ele pode ser usado.

Exemplo

- Para variáveis locais, é o corpo da função.
- Para variáveis declaradas no início de um bloco, é o bloco mesmo.

Variáveis globais : declaração e definição

Declaração + definição

```
1 int sp;
2 double val[MAXVAL];
```

- A variável é criada e inicializada
- O espaço de memória é alocado

Declaração só

```
1 extern int sp;
2 extern double val[MAXVAL];
```

- A variável é definida fora (por exemplo, num outro arquivo).
- Ela não é criada
- O espaço de memória é alocado fora.

Exemplo

- Qual é o resultado retornado por essa função ?

```
1 int x = 5;
2 int f() {
3   int x = 3;
4   {
5     extern int x;
6     return x;
7   }
8 }
```


Referências

Precisões

[KR88] Seções 4.1 – 4.5

[Bac13] 9.1

André Backes, *Linguagem C completa e descomplicada*, Elsevier, 2013.

Brian W. Kernighan and Dennis M. Ritchie, *The (ANSI) C Programming Language*, 2nd ed., Prentice Hall Professional Technical Reference, 1988.

Perguntas ?

http://dimap.ufrn.br/~richard/dim0321