Subprogramas: Funções e Procedimentos

1. Objetivos

- Apresentar os conceitos de Subprogramas
- Exemplificar a aplicação dos conceitos de Subprogramas.

2. Subprogramas

Quase todos os programas de computador que resolvem problemas do mundo real são muito maiores que os programas que já estamos acostumados a fazer. A melhor maneira de desenvolver programas grandes é dividi-los em partes menores, ou módulos ou subprogramas. Cada módulo ou subprograma, por ser uma pequena parte de um programa maior, tende ser mais fácil administrar. Na linguagem de programação C esses módulos podem ser chamados de procedimentos ou funções. Os programas escritos em C normalmente são escritos combinando-se funções pré-definidas pela linguagem, em sua biblioteca padrão, com funções escritas pelos programadores.

> "Subprogramas são trechos de programa que realizam uma tarefa específica. Podem ser chamados pelo nome a partir do programa principal ou de trechos de outros **subprogramas**, até mesmo ele próprio (chamada recursiva)." (LEITÃO, Helena C. G. Subprogramas. Disponível em: <http://www.ic.uff.br/~hcgl/subprograma.htm>. Acesso em: 19 abr. 2012)

Sendo assim, podemos definir como subprograma qualquer parte de um programa maior que seja responsável pela realização de um pedaço da solução do problema que o programa se propõe a resolver.

Os subprogramas possuem a mesma forma estrutural da função **main**, porém, ele não se encontra dentro dela, podendo ser especificado antes ou depois dela. No caso de uma especificação anterior ao **main**, deverá ser descrito o subprograma completo; já no caso de uma especificação posterior à função **main**, existe a necessidade de uma declaração prévia do subprograma, ou seja, acontecerá a declaração de todos os subprogramas descriminados no código, em seguida a função **main**, e posteriormente virá a descrição do subprograma.

Existem dois tipos distintos de subprogramas, os **procedimentos** e as **funções**, ambos são estruturados da mesma forma, a diferença entre eles, é que as **funções retornam um valor**, enquanto os **procedimentos não retornam valor algum**. Sendo assim, cabe ao programador decidir qual tipo de subprograma utilizar, sempre levando em consideração o problema que necessita solucionar.

A linguagem C possui apenas o subprograma do tipo **função**, sendo assim, caso exista a necessidade da utilização de um procedimento, é necessário que seja feita uma simulação, ou seja, deverá ser especificada uma função que não retorne valor, ou seja, deverá ser declarado o tipo de retorno como **void**, não necessitando colocar, no decorrer de sua implementação a instrução de retorno.

2.1. Declaração de um subprograma:

A regra geral para a declaração de um subprograma é a seguinte:

< tipo_do_valor_retornado > < nome_função > (< parametros_formais >)

Onde:

- tipo_do_valor_retornado: De acordo com o resultado que pretende-se obter da função, determina-se o tipo do valor que ela irá retornar. Uma função poderá retornar qualquer tipo de dado da linguagem C.
- nome_função: Nome que o programador deseja dar à função. É interessante que o nome seja sugestivo, permitindo que qualquer pessoa que manusear o código consiga compreender qual a finalidade dessa função.
- parametros_formais: São os dados de entrada da função, ou seja, é qualquer tipo de dado que o subprograma que invoca a função passa para ela a fim de que ela possa processá-los e devolver um resultado.

> "**Parâmetros formais** - são aqueles passados na declaração da função. É onde informamos quais as variáveis que a função irá receber quando chamada e quais os seus tipos (são informados como uma declaração de variáveis). Esses parâmetros são considerados como variáveis locais a função. Se a função não precisa receber nenhum parâmetro, colocamos entre os parênteses a palavra **void**." (LEITÃO, Helena C. G. Subprogramas. Disponível em: < http://www.ic.uff.br/~hcgl/subprograma.htm>. Acesso em: 19 abr. 2012)

Exemplificando com a linguagem C teremos:

float media (int nota1, int nota2);

Podemos entender a linha de código acima da seguinte forma: A função **media** recebe dois valores do tipo inteiro (**int**), que são **nota1** e **nota2**; após processar estes valores ela retornará um novo valor do tipo real (**float**).

2.2. Chamada de um subprograma:

Um subprograma pode ser chamado a partir de qualquer parte do programa principal, ou seja, pode ser invocado pela função main, por outros subprogramas, ou até por ele mesmo. Quando existe a necessidade de um subprograma invocar a si próprio acontece um fenômeno conhecido como recursividade.

A regra geral para a chamada de um subprograma é a seguinte:

< variavel_armazenamento > = < nome_funcao > (< parametros_reais >)

Onde:

- variavel_armazenamento: É uma variável do mesmo tipo de retorno da função invocada, que servirá para armazenar o valor retornado pela função invocada. No caso de simulação de procedimentos, essa variável deverá ser suprimida.
- nome_funcao: Nome que o programador deu à função que está sendo chamada.
- parametros_reais: Parâmetros passados na chamada da função. Necessariamente devem ser do mesmo tipo dos parâmetros formais definidos na declaração da função.

> "Parâmetros reais - são aqueles passados na chamada da função. É quando informamos quais os valores que os parâmetros terão dentro da função. Se a função não espera receber nenhum parâmetro, na sua chamada colocamos o abre e fecha parênteses vazio." (LEITÃO, Helena C. G. Subprogramas. Disponível em: < http://www.ic.uff.br/~hcgl/subprograma.htm>. Acesso em: 19 abr. 2011)

Exemplificando com a linguagem C teremos:

M = media(8,10);

Podemos entender a linha de código acima da seguinte forma: A variável M, do tipo float, armazenará o valor retornado pela função media após o processamento dos valores 8 (nota1) e 10 (nota2), ambos do tipo inteiro.

2.3. Estruturação de um subprograma:

A regra geral para a estruturação de um subprograma é a seguinte:

< tipo_do_valor_retornado > < nome_função > (< parametros_formais >)

{

< tipo_do_valor_retornado > < nome_variavel_retorno >

< bloco_ comandos >

return(< nome_variavel_retorno >);

}

Onde:

- tipo_do_valor_retornado: De acordo com o resultado que pretende-se obter da função, determina-se o tipo do valor que ela irá retornar. Uma função poderá retornar qualquer tipo de dado da linguagem C.
- nome_função: Nome que o programador deseja dar à função. É interessante que o nome seja sugestivo, permitindo que qualquer pessoa que manusear o código consiga compreender qual a finalidade dessa função.
- parametros_formais: São os dados de entrada da função, ou seja, é qualquer tipo de dado que o subprograma que invoca a função passa para ela a fim de que ela possa processá-los e devolver um resultado.

- nome_variavel_retorno: Nome dado à variável que será retornada. A declaração dessa variável é feita localmente dentro do subprograma que a utilizará.
- bloco_comandos: Local onde qualquer tipo de processamento será feito utilizando, ou não, os parâmetros formais.
- return: Toda função deve retornar um valor, que será definido em sua declaração, porém, para que ela possa retornar esse valor, é necessário que seja utilizado ao final do processamento, o comando return.

Exemplificando com a linguagem C teremos:

```
float media(int nota1, int nota2)
{
float x;
x = (nota1+nota2) / 2;
return(x);
}
```

Podemos entender as linhas de código acima da seguinte forma: A função media recebe dois valores do tipo inteiro (int), que são nota1 e nota2; após processar estes valores ela retornará um novo valor do tipo real (float). O processamento dos valores recebidos pela função, ocorrem através da média aritmética entre a nota1 e nota2, o resultado desse calculo é armazenado na variável x, do tipo float, que será retornada pela função media.

1.4. Estruturação de um programa com a utilização de subprogramas:

A regra geral para a estruturação de um programa utilizando subprogramas é a seguinte:

```
//Declaração dos subprogramas
< tipo_do_valor_retornado > < nome_função > (< parametros_formais >);

//Função Main
int main()
{
< bloco_ comandos >
< variavel_armazenamento > = < nome_funcao > (< parametros_reais >);
< bloco_ comandos >
}

//Estruturação dos subprogramas
```

```
< tipo_do_valor_retornado > < nome_função > (< parametros_formais >)
{
< tipo_do_valor_retornado > < nome_variavel_retorno >
< bloco_ comandos >
return(< nome_variavel_retorno >);
}
```

Exemplificando com a linguagem C teremos:

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

float media(float nota1, float nota2);
int main(){
	float N1, N2, M;
	printf ("Digite o primeiro valor: ");
	scanf("%f", &N1);
	printf ("\nDigite o segundo valor: ");
	scanf("%f", &N2);

	M = media (N1,N2);
	printf("\n\nA media dos valores e: %.2f", M);

	printf("\n\n");
	system("Pause");
	return (0);
}
float media(float nota1, float nota2){
	float N;
	N = (nota1+nota2)/2;
	return(N);
}
```

2. Outras fontes de Pesquisa:

Capítulo 6: SCHILDT, Herbert. C Completo e Total. Editora: Makron Books, 1997.

Capítulo 7: ASCENCIO, A. F. G., CAMPOS, E. A. V. de. Fundamentos da Programação de Computadores: Algoritmos, Pascal e C/C++. São Paulo, Editora: Prentice Hall, 2002.