Programação / Programação I
LEI/1, LTSI/1, LMAT/1

Hugo Pedro Proença

Universidade da Beira Interior
Departamento de Informática

Resumo

- Conceito de Função
  - Sintaxe
  - Exemplos
- Variáveis
  - Locais
  - Globais
- Parãmetros
  - Referência
  - Valor

Funções

- Como a maioria das linguagens estruturadas, o C permite a organização do código em unidades lógicas designadas de funções.
  - Um programa poderá ser composto por um conjunto de funções e respectivas chamadas.

Funções

- Tal como o nome indica, a utilidade / objectivo de cada função deverá estar claramente definido.

- Exemplos:
  - Função para receber um conjunto de valores do utilizador.
  - Função para mostrar um menu de ajuda.
  - Função para contar o numero de elementos positivos de um conjunto.
  - ...

Funções : Sintaxe

Protótipo da função (1ª linha, terminada com ;)

```
<tipo_valor_retorno> <identificador> (<parâmetro 1>, <parâmetro 2>){

        //código da função

        return <valor_retorno>;
}
```

Output (ponto de saída / retorno da função. O valor retornado terá que respeitar o tipo indicado no protótipo da função)

Input (dados de entrada na função, necessários para a sua execução)

Funções : Exemplo

- **Função para mostrar um menú ao utilizador:**

```
void mostra_menu(){
        printf("1- Inserir Cliente\n");
        printf("2- Remover Cliente\n");
        printf("3- Sair\n");
        return;
}
```

Funções : Exemplo

- Função para calcular o maior elemento de um vector:

```
float maximo_vector(int v[100]){
        float max=v[0];
        int ic;
        for (ic=1; ic<100; ic++){
                if (v[ic]>max)
                        max=v[ic];
        }
        return max;
}
```

Funções : Valor de Retorno

- Cada função poderá (no máximo) retornar um valor após a sua execução
  - Tipos básicos:
    - int, char, float, void
  - Tipos compostos:
    - pessoa, cliente, produto, ...
- O valor de retorno será devolvido para o local onde foi efectuada a chamada à função.

Funções : Chamada

- A chamada de uma função faz-se através do seu identificador, conjuntamente com o respectivo conjunto de parãmetros de entrada:

```
for (i=0; i<10; i++){
        mostra_menu();
}
```

- Quando o tipo de retorno for dieferente de “void”, deverá sempre receber-se o valor que a função retorna:

```
if (opcao==7)
        m=maximo_vector(m);
else
        printf(“O maximo é %f\n”,maximo_vector(m));
```

Funções : Chamada

- É imprescindível respeitar o número e tipo de parâmetros de entrada:

```
int func1(int a, float b){
    return(a*2);
}
void main(){
    int v1=2, v2=6, va;
    float v3=1.4, v4=3.2, vb;
    func1();            // s/ parâmetros e s/ valor retorno
    va=func1();         // s/parâmetros
    va=func1(v1, v2);       //tipos errados
    va=func1(v1);           //parâmetros insuficientes
    va=func1(v1, v4);       //OK
}
```

Funções : Execução

```
#include <stdio.h>
void f1(){
        linha1;
        linha2;
        linha3;
}
void f2(){
        linha4;
        f1();
        linha5;
}
void f3(){
        linha6;
        f2();
        linha7;
        f1();
}
main(){
        linha8;
        f1();
        linha9;
        f3();
}
```

Ao efectuar uma chamada a uma função a execução é transferida para dentro dessa função. Após o seu término, a execução regressa ao local onde foi efectuada a chamada

Exemplo (ordem de execução):

```
linha8;
linha1;
linha2;
linha3;
linha9;
linha6;
linha4;
linha1;
linha2;
linha3;
linha5;
linha7;
linha1;
linha2;
linha3;
```

Funções : Variáveis Locais

- Por regra, todas as variáveis declaradas dentro de uma função são locais a essa função.
- Apenas têm alcance dentro da respectiva função:

```
void func1(){
    int v1;
    scanf("%d",&v1);
}

void func2(){
    int v2;
    v2=v1+10;          // X: A variável "v1" não tem alcance nesta função
}
```

Funções : Variáveis Locais

- Devido ao facto de serem variáveis locais, não existe qualquer incoveniente em definir o mesmo nome para variáveis em diferentes funções:

```
void func1(){
    int v1;
    scanf("%d",&v1);
}

void func2(){
    int v1;
    v1=10;
}
```

Funções : Variáveis Globais

- Uma variável global é declarada fora das funções e tem alcance em qualquer local do programa.

```
int v1;                                  //variável global

void func1(){
        scanf("%d",&v1);
}

void func2(){
        v1=10;
}

main(){
        f1();
        printf("O valor é %d\n",v1);
}
```

- Regra: <u>Nunca se devem usar variáveis globais.</u>
  - Propiciam erros de programação.
  - Tornam o código menos portável e reutilizável.

Funções : Variáveis Globais / Locais

- O alcance da variável global só poderá ser afectado por declarações de variáveis locais com o mesmo nome nas respectivas funções:

```
int v1;

void func1(){
        int v1;
        scanf("%d",&v1);                // O "v1" que é lido é o da variável local
}

void func2(){
        v1=10;
}

main(){
        f1();
        printf("O valor é %d\n",v1);            //Erro. "v1" não tem valor
}
```

Funções : Passagem de Parãmetros

```
int func1(int a, int b){
        a=a+1;
        return(a+b);
}
main(){
        int v1=10, v2=5, v3;
        v3=func1(v1,v2);
        printf("v1=%d, v2=%d e v3=%d\n",v1,v2,v3);
}
```

Neste caso o valor da variável "v1" é copiado para o parãmetro "a" e o valor da variável "v2" é copiado para o parãmetro "b".

- Efectuou-se uma passagem de parãmetros por valor.
- O valor de "a+2" retorna da função e copiado para a variável "v3".

- O valor de "v1" e "v2" não sofrem qualquer tipo de alteração, uma vez que os parãmetros são variáveis-cópias de "v1" e "v2".

Funções : Passagem de Parãmetros

- Na linguagem C, os vectores e matrizes são passados através de parãmetros por referência.
  - Não são criadas cópias das variáveis originais, mas sim uma simples referência para elas.
    - Potencial elevado volume de dados.
  - Implica que os valores alterados dentro de uma função têm repercussão global.

Funções : Passagem de Parãmetros

```
void f1(int v[10], int x){
        int ic;
        x=x+1;
        for (ic=0;ic<10;ic++)
                v[ic]=v[ic]+1;
}

main(){
        int ic, ix=9;
        int vector[10]={1,2,3,4,5,6,7,8,9,10};
        f1(vector,ix);
                        //Passagem por referência. "vector" é alterado
                        //Passagem por valor.  "ix" não é alterado
        for (ic=0;ic<10;ic++)
                printf("%d\n",vector[ic]);
        printf("%d\m",ix);
}
```

Funções : Exercício

Suponha que é necessário registar as classificações dos alunos de uma turma do ensino básico, composta por 20 alunos. Para tal foi decidido utilizar um vector com 15 posições.

Codifique funções em C que:

a. Receba o conjunto de 20 classificações para o vector. Tenha em atenção que deve ser garantido que as notas introduzidas são válidas ([0,20]).

b. Devolva o total de alunos com classificações positivas.

c. Devolva a classificação média dos alunos aprovados.

d. Verifique se existem notas repetidas entre os alunos da turma. A função deve devolver "1" em caso afirmativo, ou "0" em caso contrário."

Implemente o programa principal que recebe o conjunto de elementos do vector e fornece a informação correspondente às funções implementadas.