Introdução à Programação

1.Ano LCC-MIERSI
DCC - FCUP

Nelma Moreira

Aula 5

Repitições controladas por contadores

Contar até 10

```
main() {
   int contador=0;
   while (++contador <= 10)
      printf("%d\n", contador);
}
```

Pseudo-código

1. Uma variável de controlo (`contador`)
2. Um valor inicial para essa variável
3. Uma condição que teste o valor final dessa variável (a partir do qual o ciclo deve terminar)
4. Um incremento (ou decremento) dessa variável antes de cada repetição do ciclo

Instrução `For`

```
for(Expr1; Expr2; Expr3) Inst
```

onde `Expr1`, `Expr2` e `Expr3` são expressões e `Inst` uma instrução

```
main() {
  int contador;
  for(contador = 1; contador <= 10; contador++)
    printf("%d\n", contador);
}
```

O que fazem?

```
for(i = 100; i >= 1; i++);

for(i = 7; i <=  77; i += 7);

for(i = 99; i >= 0; i -= 11);

for(j = 2; j <= 1000; j *= 2);
```

Numa instrução `for` as expressões podem ser de qualquer tipo (mesmo omitidas) e equivale a:

```
Expr1;
  while (Expr2)
    Inst
    Expr3;
  }
```

`Expr1` e `Expr3` podem ser expressões múltiplas, i.e várias expressões separadas pelo operador `,`

```
for(x = 0, y = 0; x + y <10; ++x);
for(i = 1,s = 5;i + s != 7;) i++;
s = 2; for(;;){s *= s; if (s>100) break;}
```

Instrução `do while`

`do Inst while ( Expr );`

1. A instrução `Inst` é executada.
2. A expressão `Expr` é avaliada. Se for não nula (verdadeira), volta a 1, se for nula (falsa) a instrução termina.

```
do {
     scanf("%d",&i);
  } while (i!=0);
```

Problema 1

Analisar sequência de inteiros (com terminador 0) lida da entrada padrão e indicar os dois últimos valores lidos distintos, sabendo a sequência tem sempre pelo menos dois valores distintos, além de 0.

Variáveis

* penúltimo valor
* último valor
* novo valor

Programa em C

```
#include <stdio.h>
int main() {
  int penultimo, ultimo,
novo;
  scanf("%d",&penultimo);
  do {
  scanf("%d",&ultimo);
  }
while (ultimo == penultimo);
scanf(“%d",&novo);
while(novo != 0) {
if  (novo != ultimo)
{penultimo = ultimo;
ultimo = novo;}
scanf("%d",&novo);}
printf("%d %d\n",penultimo,
ultimo);
return 0;
}
```

Instrução `break`

`break;`

A instrução `break` termina a execução do ciclo mais interno ou instrução de escolha a que pertence.

```
for(;;) {
 printf("Introduz um numero (0 para sair): ");
 scanf("%d",&n);
 if (n == 0) break;
 printf("%d ao cubo é %d", n, n*n*n);
}
```

Instrução `continue`

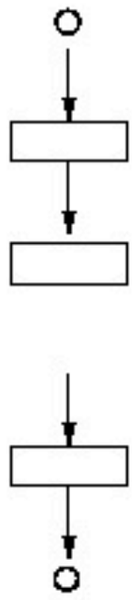
A instrução `continue` inicia uma nova iteração do ciclo mais interno a que pertence.

Nos ciclos `while` e `do while` obriga a que a `Expr` seja avaliada imediatamente;num ciclo `for` a expressão `Expr3` é avaliada.

```
for(i = 1; i < 10; i++) {
   if (i % 2 != 0) continue;
   printf("%d\n",i);
}
```

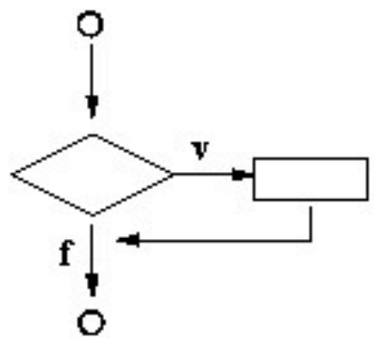
Instruções

Sequencial

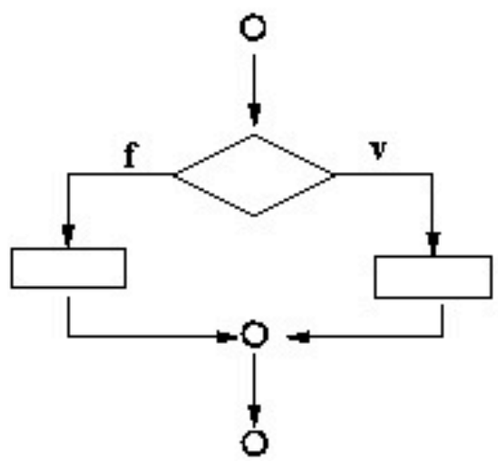


Condicionais

if (expr) inst

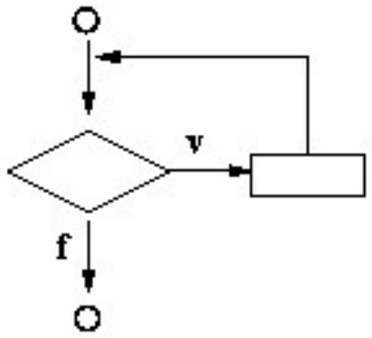


if (expr) inst1 else inst2

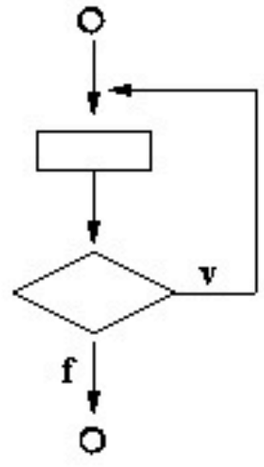


Repiticao (ciclos)

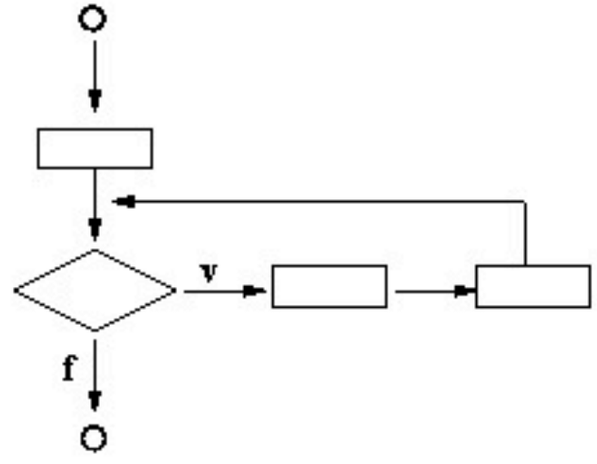
while(expr)inst



do inst while (expr)



for(exp1;exp2;exp3)inst



Intervalo

5 minutos

O factorial, outra vez

$$fact(0)=1$$
$$fact(x)=x \times fact(x-1)$$

```
#include<stdio.h>
   int fact(int);

   main(){
     int n;
     scanf("%d",&n);
     printf("Factorial de %d = %d\n",n,fact(n));
   }

   /*  definição da função  */
   int fact(int x){
     if(x==0) return(1);
     return(x*fact(x-1));
  }
```

Funções

* As funções em C estruturam um programa, permitindo modelaridade, mais legibilidade e que um mesmo conjunto de instruções seja repetido em vários pontos.

* Ao contrário das funções matemáticas:
  * Podem ter ou não argumentos
  * Podem calcular (retornar) ou não um valor

Problema 2: Potênciação

Tabelar as potências x^n para 1 <= n <= 5 e 2<= x <= 10

```
int potencia(int x, int n) {
  int i, p = 1;
  for(i = 1; i <= n; i++)  p = p*x;
  return p;
}
main(){
  int n, x;
  for(n = 1; n <= 5; n++) {
    for(x = 2; x <= 10; x++) printf("%d ",potencia(x,n));
    printf("\n");}
}
```

Melhorar o resultado

```
main() { /* potências inteiras */
   int i, x;
   printf("\t\t\tTabela das potencias\n");
   printf("\t\t\t----------------------\n\n");
   for(i = 1; i <= 5; i++){
      printf("%3d |",i);
      for(x=2;x<=10;x++) printf("%6d ", potencia(x,i));
      printf("\n");

   }

}
```

Execução

```
$ gcc pot.c -o pot
$pot
```

```
                    Tabela das potencias
                    --------------------

  1 |   2     3     4     5     6     7     8     9     10
  2 |   4     9    16    25    36    49    64    81    100
  3 |   8    27    64   125   216   343   512   729   1000
  4 |  16    81   256   625  1296  2401  4096  6561  10000
  5 |  32   243  1024  3125  7776 16807 32768 59049 100000
```

Definição de função

```
tipo_do_resultado
nome_da_funcao(declaracao parametros)
{
   declaracoes
   instrucoes
}
```

As variáveis declaradas dentro duma função e os parâmetros são locais, isto é, só são conhecidos dentro da função.

Chamada de função

```
nome_da_funcao(argumentos);
```

onde `argumentos` é uma lista de expressões, cada um correspondendo a um parâmetro da definição da função. Os argumentos são passados por valor, isto é, o valor duma variável passada como argumento não é alterado pela execução da função!

```
n = 3;  m = 5;
p = potencia(n,m);
```

Regresso de função

```
return Expr;
return;
```

Estas instruções podem ocorrer em qualquer parte duma função e causam a sua terminação, sendo `Expr`, se existir, o valor retornado.

```
int max(int x,int y) {
 if (x>y)
   return x;
 else
   return y;
}
```

Protótipos de função

```
tipo_do_resultado nome_da_funcao(declaracao parametros);
```

Se uma função for usada antes de ser definida (ou se é definida noutro ficheiro), os protótipos indicam qual o tipo de resultado e o tipo de cada parâmetro.

```
#include <stdio.h>
int max(int,int);
main() {
int n, m;
scanf("%d %d",&n,&m);
printf("maximo: %d\n",maximo(n,m));
}
int max(int x,int y) {
if (x>y) return x; else return y;}
```

Problema 3: Primos de Mersenne

Tabelar os primos de Mersenne , 2p - 1, para p primo inferior a 32.

Definir uma função `int primo(int p)` que retorne 1 ou 0 e usar a função `int potencia(int,int)` anterior.

```
int primo(int n) {
 int i, md;
 if (n < 1) return 0;
 if(n == 2 || n == 3) return 1;
 if(n%2 == 0) return 0;
 else {
   md = sqrt(n); i=3;
   while (n%i != 0 &&  i <= md) i += 2;
   if (i > md) return 1;
   else return 0;
 }
}
```

```
#include <math.h>
#include <stdio.h>
int primo(int);
int potencia(int,int);
main() {
  int m = 3, mer;
  printf("\n Primos de Mersenne\n====================\n\n");
  printf("|  p  |     2^p-1    |\n\n");
  printf("| %3d | %12d |\n",2,3);
  while (m < 32) {
    if (primo(m)) {
      mer=potencia(2,m)-1;
      if (primo(mer)) printf("| %3d | %12d |\n",m,mer);
    }
    m+=2;
  }
}
```

Execução

```
% gcc mersenne.c -omersenne -lm
% mersenne
 Primos de Mersenne
====================
|   p |       2^p-1     |

|   2 |               3 |
|   3 |               7 |
|   5 |              31 |
|   7 |             127 |
|  13 |            8191 |
|  17 |          131071 |
|  19 |          524287 |
|  31 |      2147483647 |
%
```

Estrutura de um Programa em C

* Comandos para inclusão de ficheiros com declarações
* Definições de constantes
* Declarações de variáveis
* Declarações de funções
* Definições de funções (sub-programas
* Definição da função `main` (programa principal)

Biblioteca de funções matemáticas

Incluir o ficheiro de cabeçalhos `#include <math.h>`
Todas retornam e aceitam como parâmetros valores de tipo `double`

Função	Descrição	Exemplo
`sqrt(x)`	raiz quadrada de x	`sqrt(9) é 3`
`exp(x)`	função exponêncial base e	`exp(1) é 2.7182`
`log(x)`	logaritmo base e	`log(2.171) é 1`
`ceil(x)`	menor inteiro não menor do que x	`ceil(9.2) é 10`
`floor(x)`	maior inteiro que não excede x	`floor(9.2) é 9`
`pow(x,y)`	potência y de x	`pow(2,7) é 128`
`sin(x)`	seno de x em radianos	`sin(0.0) é 0`
`cos(x)`	coseno de x em radianos	`cos(0.0) é 1`
`tan(x)`	tangente de x em radianos	`tan(0.0) é 0`

Biblioteca de funções de entrada e saída

Incluir o ficheiro de cabeçalhos `stdio.h` .

A função `printf()` permite a escrita formatada

A forma geral é:

```
printf(<formato>,<arg1>,...,<argn>)
```

```
printf("valorn= %3d valorx= %f \n",n,x);
```

O formato é uma sequência de caracteres que serão escritos, sendo cada especificação iniciada por `%`, substituída pelo valor argumento correspondente depois de convertido.

Caracteres

caracter	Tipo	valor
d	int	escrito comando
o	int	inteiro positivo octal
x	int	inteiro positivo hexadecimal
c	int	caracter
s	char *	sequência de caracteres
f	float	número em vírgula flutuante
g	double	número em vírgula flutuante
e	float	número em vírgula flutuante em notação científica

Formatação entre % e caracter

* Um - indicando ajustamento à esquerda: `%-6d`
* Um inteiro que especifica o tamanho do campo. Ex: `%6d`, escreve um inteiro ocupando 6 caracteres, se necessário com espaços à esquerda.
* um . que separa o tamanho, da precisão
* um número, que indica ou
  * máximo de caracteres a serem escritos.
  * dígitos depois do . em números em vírgula flutuante. Ex. `%5.2f`
  * mínimo de dígitos de um inteiro. Ex. `%10x`

Caracteres especiais

Caracter	Significado
\b	backspace
\n	mudança de linha
\t	tabulação
\\	\
\'	'
\"	"
\0	0

Função `scanf()`

Lê caracteres da entrada, e interpreta-os de acordo com o formato e guarda o resultado nos argumentos:

```
scanf(<formato>, <arg1>,...<arg2>)
```

```
scanf("%d %f",&n,&x);
```

`%d` indica que `n` é inteiro e que a sequência de caracteres deve ser interpretada como a representação do inteiro na base 10

`%f` indica que `x` é do tipo float

`&` indica que nos referimos ao endereço das variáveis `x` e `n` em vez do seu valor.