Introdução à Programação

1.Ano LCC-MIERSI
DCC - FCUP

Nelma Moreira

Aula 3

Tipos, Expressões,Operadores

* Tipos de dados
* Variáveis
* Constantes
* Constantes Simbólicas
* Expressões
* Operadores aritméticos, relacionais e lógicos
* Operadores de atribuição, incrementais, atribuição artméticos
* Expressões condicionais

Instruções (imperativas)

* Atribuições
* Instrução composta (sequêncial)
* Instruções de Controlo:
  * Condicionais
  * Ciclos

Tipos de dados simples

`int` números inteiros dentro de um certo intervalo (p.e de 32 bits).

`short,long` diferentes tamanhos

`unsigned` inteiros positivos

`float` números racionais em vírgula flutuante (precisão simples)

`double` números racionais em vírgula flutuante (precisão dupla)

`char` valores que representam caracteres, segundo um dado código (p.e código ASCII)

Código ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Constantes

Exemplo	Tipo
23444	Inteiro (base 10)
-233	Inteiro (base 10)
0x1f	Inteiro (base 16)
37	Inteiro (base 8)
-12.5622	Racional
'á'	caracter
"jsjdjdjdjd"	string
'\n'	caracter especial

Constantes Simbólicas

```
#define nome expressão
```

O pré-compilador substituí todas as ocorrências dessas constantes pelo respectivo valor.

* Problema:

  Tabelar a função sin(x) entre 0 e 90 graus

Tabela da função "seno" entre 0..90 graus

```
#include<stdio.h>

#define MIN      0
#define MAX      90
#define INCRE    5
#define PI       3.1415926
#define FCONV    PI/180
main() {
   int x = MIN;
      while(x<=MAX){
        printf("%3d %6.5f\n", x, sin(FCONV*x));
      x+= INCRE;
}
```

Variáveis

* Permitem guardar valores e são designadas por nomes.
* Os nomes começam por uma letra, podendo conter letras, dígitos e "_".
  Exemplos: `n, soma_10, TOTAL.`
* Antes de ser usada uma variável tem de ser declarada. Uma declaração permite:
  * Definir o nome da variável
  * Definir o seu tipo
  * Pode inicializar a variável

Exemplos

```
int soma;

int i = 0, n = MAX + 1;

float media;

char c = 'x';
```

Expressões

Contêm variáveis, operadores, constantes e chamadas a funções

```
55 + x

5

5 *(sin(x) + 1)

x = 7 * y + z/w + (z-1)

x + 1 != 4 || y > x + 2

x == 4 && y > 0

x >= 4 && !y > 0

x > 'a'
```

Operadores aritméticos

* Operadores binários: `+, -, *, /, %`
* Operadores unários: `-`

```
if (i % 3 != 0) printf( "%d \n",i);
```

Se os operandos são inteiros `/` efectua a divisão inteira

```
int n = 4, q, r;
q = 30 / n; r = 30 % n;
```

Os operadores têm a prioridade usual e são avaliados da esquerda para a direita.

Operadores relacionais

Operador	Semântica
==	Igualdade
!=	diferença
>	maior que
>=	maior ou igual
<	menor que
<=	menor ou igual

O valor duma expressão que envolve estes operadores ou é `0` (falso) ou `1` (verdade).

Não confundir `==` com `=` (atribuição)!

Operadores lógicos

	Operação	Semântica
!	Negação	1 se argumeto 0, senão 0
&&	Conjunção	1 se argumetos não 0, senão 0
||	Disjunção	1 se um dos argumentos 1, senão 0

Supondo `x` tem valor `5` e `y` tem valor `0`

Expressão	Valor
`!(x == 8)`	1
`x == 8 || y < 1`	1
`5 && 3`	1
`5 == 3 || 3 == 0`	0

Precedências e Negação

```
i < 10 && i != 3 || i >= 20
```

é equivalente a

```
((i < 10) && (i != 3) ) || (i >= 20)
```

```
if (!num) num=1;
```
é equivalente a

```
if (num == 0) num=1;
```

```
if (!(x==0)) x=0;
```
é equivalente a

```
if (x!=0) x=3;
```

Operador de Atribuição

```
Variável = Valor
```

```
x = y + 1
```

* A expressão da direita é calculada substituindo as variáveis pelos seus valores. Se `y` é `5`, fica `5 + 1` é `6`
* O seu valor é guardado na variável à esquerda, `x`

Pode ser usadas noutras expressões:

```
x = (y = 3) + (z = 4 * 6)
```

Atribui a `y` o valor `3` ,a `z` o valor `24` e a `x` o valor `27`

Operadores In(De)crementais

`++` `--`

* O operador `++` adiciona `1` ao seu operando
* O operador `--` subtraí `1` ao seu operando

`++n;` é o mesmo que `n = n + 1;`

`--n;` é o mesmo que `n = n - 1;`

Podem ser usados com prefixos ou sufixos:

`x= ++n;` ou `x= n++;` têm significados distintos:

é incrementada antes (depois) da atribuição ser avaliada. Se `n` vale `5`, `x` vale `6` (`5`).

Operadores de atribuição aritméticos

`var op= e1`

```
x += 1
```

```
x = x + 1
```

São equivalentes. Assim como

```
x *= y + 1
```

```
x = x * (y + 1)
```

Expressões Condicionais

`e1 ? e2 : e3`

Se `e1` é `0` então o valor é `e3` senão `e2`

```
z = (a>b)? a:b;
```

É equivalente a

```
if (a>b) z = a; else z = b;
```

Exemplos:

```
passo = (nota >= 10:1 : 0
nota>=10? printf("Passo"):printf("Não");
```

Intervalo

ss

---

10 minutos

Instruções

* **Atribuição**: associção de valores a variáveis
* **Chamada de função** `fact(4)`
* **Regresso de função** `return`
* **Instrução composta:** sequência de instruções
* **Condicional** : execução de instruções condicionada pela verificação duma condição. Ex. `if ...;else...`

Instruções

* **Ciclo**: repetição (iteração) de instruções com abse numa condição. Ex. `while`

* **Escolha** de instruções com base em valores de um parâmetro. Ex. `switch`

* **Transferência de controlo** (saltos) execução duma instrução noutro ponto do programa.

Atribuição

```
Variavel op_atrib Expressão;
```

* `op_atrib` pode ser `=, +=,-=,*=,/=, %=`
* Seja `x` com valor `7` e `y` com valor `12`. Determinar o valor das variáveis no final da execução de:

1. `t=x; x=y; y=t;`
2. `x=t; y=x; t=x;`
3. `x=x; y=y; t=t;`

Instrução Composta

* Sequência de instruções delimitadas por chavetas.

```
{
   t=5;
   x=y;
   calcula(t,x);
   if(1) printf(``Ola'');
}
```

Instruções Condicionais

if

* `if ( expressao ) instrucao`

  A expressão `expressao` é avaliada; se tiver um valor não nulo (verdade) então a instrução `instrucao` é executada; se for falsa, não faz nada.

```
if (n == 9) printf("%d",n);
```

* `if ( expressao ) inst1 else inst2`
* Neste caso se `expressao` é nula a instrução é `inst2` executada

```
if (n == 9) printf("%d",n); else printf("nao");
```

Problema 1

* Se `x` é par, soma `1` e se é ímpar subtraí `1`
* Estará bem?

```
if (x %2 == 0) x = x + 1;
if (x %2 != 0) x = x - 1;
```

* Certo

```
if (x %2 == 0) x = x + 1; else x = x - 1;
```

Ordenar 3 inteiros

```
if(x>=y && x>=z){                /* x é o maior */
      if(y>=z) printf("%d %d %d\n",x,y,z);
      else printf("%d %d %d\n",x,z,y);
      }
   else if(y>=x && y>=z){              /* y é o maior */
      if(x>=z) printf("%d %d %d\n",y,x,z);
      else printf("%d %d %d\n",y,z,x);
      }
   else {                           /* z e' o maior */
      if(x>=y) printf("%d %d %d\n",z,x,y);
      else printf("%d %d %d\n",z,y,x);
      }
```

Instruções de Ciclo

`while (expressao) instrucao`

* Funcionamento:

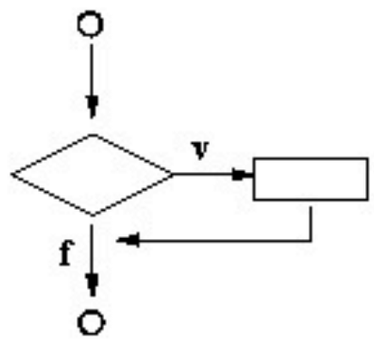
1. A expressão `expressao` é avaliada.
2. Se o resultado for `0` (falsa), a instrução termina.
3. Se o resultado for não nulo (verdade), é executado `instrucao` e volta para o passo 1.

Instruções

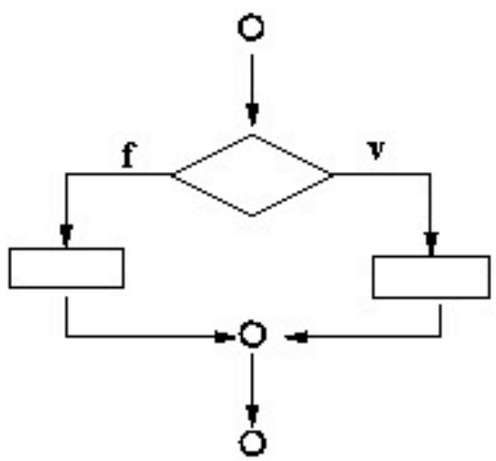
Sequencial

Condicionais

if (expr) inst

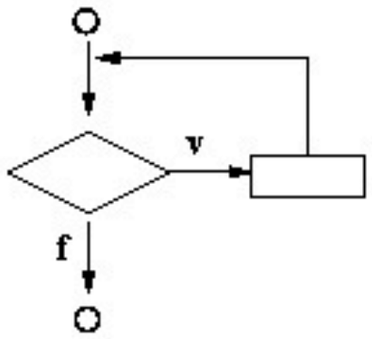


if (expr) inst1 else inst2

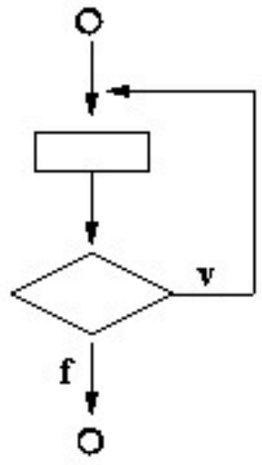


Repiticao (ciclos)

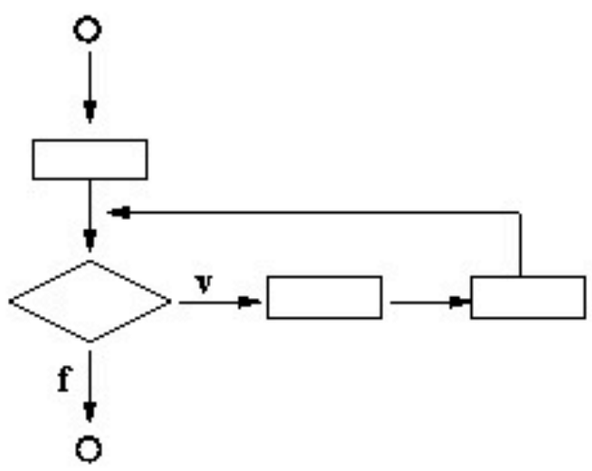
while(expr)inst



do inst while (expr)



for(exp1;exp2;exp3)inst



Problema

* Dado um inteiro positivo n determina o seu factorial n!, isto é o produto de todos os inteiros inferiores ou iguais a n (0! = 1, n! = 1×2...×n).

Ideia da resolução

* Começar por valores pequenos: 2!, 3!, 4! Para 4: multiplicar 1 por 2 e guardar o resultado, multiplicar o resultado por 3 e guardar o resultado, multiplicar o resultado por 4 e guardar o resultado. Analogamente para n.
* Mas, como gerar os inteiros? Como guardar os valores?

Algoritmo	Programa C

Ler n
fact = 1
i = 1
enquanto i <=n fazer
  fact = fact * i
  i = i + 1
	main() {
	  int n,i=1,fact=1;
	  scanf("%d",&n);
	  while(i <=n) {
	  fact = fact * i;
	  i = i + 1;
	  }
	  printf("fact(%d) = %d
	\n",n,fact);
	}

O que faz?

```
main(){
     int x,i,p;
     x=2;
     while(x!=0){
        scanf("%d",&x);
        if(x<0) break;
        p=1; i=1;
        while(i<=x){
           p*=i;
           i++;
        }
        printf("Factorial(%d)= %5d\n",x,p);
     }
}
```

Problema 3

* Determinar o maior inteiro duma sequência de n inteiros, lidos um a um.

Ideia da resolução:

* Considerar alguns valores 30, 45, 40, 38, 49, 55, 40, 34 e resolver o problema à mão. Descrever o que foi feito:

  Considerar o primeiro o maior e ir sucessivamente comparando como os restantes. Sempre que um valor for superior ao maior (até então), o maior passa a ser esse valor.

* Como representar o sucessivamente? Usando um contador, que pode ser o próprio n...

Variáveis

* **Importante**: em cada momento é só necessário o último número lido e o máximo até aí.

* `n`: número de números ainda a ler
* `i` cada valor lido
* `max` em cada momento, guarda o maior valor encontrado até aí

Algoritmo

```
Ler n
Se n < 1 termina
Ler i
max=i
n=n-1
Enquanto n > 0 faça:
   Ler i
   Se i > max então max=i
   n= n - 1
Escreve max
```

Programa C

```
main() {
    int n,i,max;
    printf("Quantos? ");
    scanf("%d",&n);
    if (n <= 0) return 0;
    printf("\n Primeiro: ");
    scanf("%d",&i);
    max = i; --n;
    while(n>0) {
      printf("\n Outro: ");
      scanf("%d",&i);
      if (i > max) max = i;
      --n;
    }
    printf(" o maximo e %d
\n",max);
  }
```