Universidade do Minho
Escola de Engenharia

Relatório:
Linguagem C

Disciplina: Linguagens de Programação
Curso: Engenharia e Gestão Industrial

Carolina Fernandes N.º40532
Guilherme Bacinello N.º42119
Diana Rocha N.º40536

Guimarães, 2004

Índice

Introdução

Origem da Linguagem C

A linguagem C foi criada por Dennis Ritchie, em 1972, no centro de Pesquisas da Bell Laboratories. A sua primeira utilização importante foi a reescrita do Sistema Operacional UNIX, que até então era escrito em Assembly.

Em meados de 1970 o UNIX saiu do laboratório para ser liberado para as Universidades. Foi o suficiente para que o sucesso da linguagem atingisse proporções tais que, por volta de 1980, já existiam várias versões de compiladores C oferecidas por várias empresas, não sendo mais restritas apenas ao ambiente UNIX, porém compatíveis com vários outros sistemas operacionais.

O C é uma linguagem de propósito geral, sendo adequada à programação estruturada. No entanto é mais utilizada escrever compiladores, analisadores léxicos, bancos de dados, editores de texto, etc..

A linguagem C pertence a uma família de linguagens cujas características são: portabilidade, modularidade, compilação separada, recursos de baixo nível, geração de código eficiente, regularidade, simplicidade e facilidade de uso.

Visão geral de um programa C

A geração do programa executável a partir do programa fonte obedece a uma sequência de operações antes de tornar-se um executável. Depois de escrever o módulo fonte em um editor de textos, o programador acciona o compilador que no UNIX é chamado pelo comando cc. Essa acção desencadeia uma sequência de etapas, cada qual traduzindo a codificação do usuário para uma forma de linguagem de nível inferior, que termina com o executável criado pelo lincador.

Editor (módulo fonte em C)

Pré-processador (novo fonte expandido) – itálico!!!

Compilador (arquivo objeto)

Lincador (executável)

Características do C

Entre as principais características do C, podemos citar:

- O C é uma linguagem de alto nível com uma sintaxe bastante estruturada e flexível tornando sua programação bastante simplificada;

- Programas em C são compilados, gerando programas executáveis;

- O C compartilha recursos tanto de alto quanto de baixo nível, pois permite acesso e programação directa do microprocessador;

  Com isto, rotinas cuja dependência do tempo é crítica, podem ser facilmente implementadas usando instruções em Assembly. Por esta razão o C é a linguagem preferida dos programadores de aplicativos.

- O C é uma linguagem estruturalmente simples e de grande portabilidade. O compilador C gera códigos mais enxutos e velozes do que muitas outras linguagens.

- Embora estruturalmente simples (poucas funções intrínsecas) o C não perde funcionalidade pois permite a inclusão de uma farta quantidade de rotinas do usuário. Os fabricantes de compiladores fornecem uma ampla variedade de rotinas pré-compiladas em bibliotecas.

A organização dos programas C

Um programa C é composto por **funções**, isto é, porções menores de código que realizam determinadas tarefas, e por **dados**, ou seja, variáveis ou tabelas que são inicializadas antes do início do programa. Existe uma função especial denominada **main**, onde a execução do programa se inicia. As funções são organizadas em módulos agrupados nos arquivos fonte. Em C, a organização do código em arquivos tem um significado semântico. O arquivo fonte principal, passado como argumento para o compilador, define uma unidade de compilação.

Uma unidade pode importar definições usando a diretiva #include ou apenas declarando algum identificador como externo.

C permite um modelo de compilação separado, isto é, você pode dividir o programa em várias unidades independentes que são compiladas separadamente e depois são encadeadas com o link editor para construir o programa final. Normalmente, cada módulo é escrito num arquivo texto separado que contém funções ou declarações de dados. As interfaces entre os módulos são escritas em "header files" (arquivos de cabeçalho) que descrevem tipos ou funções visíveis a vários módulos do programa. Estes arquivos possuem a extensão ".h" e são de dois tipos: privados, específicos da aplicação que está sendo elaborada, e para o sistema.

Toda função possui uma lista de parâmetros, um corpo e, eventualmente, um valor de retorno. O corpo pode conter declarações de variáveis locais, ou seja, variáveis que são activadas quando a execução alcançar o corpo da função.

Enquadramento teórico

Objectivos

- declarar, inicializar e utilizar correctamente variáveis dos diferentes tipos de dados
- reconhecer a sintaxe e aplicar correctamente as estruturas condicionais
- reconhecer a sintaxe e aplicar correctamente as várias estruturas cíclicas
- gerar e utilizar números aleatórios
- declarar, inicializar e utilizar correctamente vectores
- declarar, inicializar e utilizar correctamente matrizes
- declarar, inicializar e utilizar correctamente variáveis heterogéneas
- declarar strings e utilizar correctamente as funções básicas sobre strings
- declarar e utilizar funções
- declarar e utilizar funções recursivas
- saber editar, compilar e executar programas no TurboC

Programa

Enunciado 178
Fichas de jogadores

Muitas equipas de desportos profissionais utilizam um computador para auxiliar na análise de jogadores. Suponha que uma equipa profissional de "hockey" tenha um sistema deste tipo. Para cada jogador observado, é preparado um cartão com os seguintes dados:

nome do jogador, idade, altura (em cm), peso (em kg), golos no último campeonato, presenças no último campeonato, penalidades em minutos no último campeonato, factor da confederação (número real)

Os jogadores são avaliados segundo a seguinte fórmula:
(golos + presenças + (penalidades minutos)/4 + (altura+peso)/5 - idade)*factor de confederação

Preparar um algoritmo para ler o arquivo completo de jogadores em observação, listando para cada jogador as informações de seu cartão e a sua avaliação. No fim da listagem (indicada por um cartão especial com o nome de jogador 'FIM DE LISTAGEM'), dar o nome e a avaliação do jogador com o maior valor.

Resultados

Este é o nosso programa:

```
                    /* Avaliação de jogadores de hockey */
#include<stdio.h>
#include<conio.h>
main()
{
  struct joga
  {
    float idade, golos, peso, altura, pres, pen, factor;
    char nome[40];
  };
  struct joga pessoa[23];
  int k, j, i, x, cont;
  float maior, media[23];
  clrscr();
  printf("Vamos fazer a avaliação dos jogadores de hockey com no máximo 24
jogadores:\n");
  cont=1;
  x=-1;
  do
  {
    x=x+1;
    j=x+1;
    printf("Digite o nome do %i§ jogador: ",j);
    scanf("%s", &pessoa[x].nome);

    do
    {
      printf("Digite a idade do jogador:");
      scanf("%f", &pessoa[x].idade);
    }
    while(pessoa[x].idade<0);

    do
    {
      printf("Digite a altura(cm):");
      scanf("%f", &pessoa[x].altura);
    }
    while(pessoa[x].altura<0);

    do
```

```
    {
      printf("Digite peso(kg):");
      scanf("%f", &pessoa[x].peso);
    }
    while(pessoa[x].peso<0);

    do
    {
      printf("Digite os golos do ultimo campeonato :");
      scanf("%f", &pessoa[x].golos);
    }
    while(pessoa[x].golos<0);

    do
    {
      printf("Digite o numero de presenças no ultimo campeonato:");
      scanf("%f", &pessoa[x].pres);
    }
    while(pessoa[x].pres<0);

    do
    {
      printf("Digite as penalidades em minutos:");
      scanf("%f", &pessoa[x].pen);
    }
    while(pessoa[x].pen<0);

    printf("Digite o factor confederação:");
    scanf("%f", &pessoa[x].factor);

    printf("Se quiseres continuar prima 0 (zero)  se não prima qualquer outro
caracter: ");
    scanf("%i", &cont);

    media[x]= ( (pessoa[x].golos) + (pessoa[x].pres) + ((pessoa[x].pen)/4)+
((pessoa[x].altura)+(pessoa[x].peso))/5 -(pessoa[x].idade))*(pessoa[x].factor);

          if (x>23)
          {
                printf("J  atingiu o m ximo de jogadores(24) do programa.");
                cont=1;
          }
  }

  while(cont==0);
```

```
 i=0;

 maior=media[0];
 i=0;
 do
 {
  if (media[i]>=maior)
  {
    maior=media[i];
  }
 i=i+1;
 }
 while(i<=x);
 clrscr();
 i=0;
 do
 {

  if(media[i]==maior)

                printf("A maior m,dia , do %s com %6.2f.\n Tem%3.0f anos,
pesa%3.0f kilos, seu factor de confederação , %5.2f e no £ltimo campeonato
fez%3.0f golos e%3.0f penalidades, com%3.0f presenças."
    ,pessoa[i].nome, media[i], pessoa[i].idade, pessoa[i].peso, pessoa[i].factor,
pessoa[i].golos, pessoa[i].pen, pessoa[i].pres);
    i=i+1;
 }
 while(i<=x);
 }
```

Conclusão

Na ciranda das linguagens de programação, a linguagem C permanece firme. Concluímos então, que a linguagem C não é uma linguagem orientada a objecto, mas pode-se fazer uma programação orientada a objecto se assim o desejar. Não é uma linguagem funcional, mas pode-se programar funcionalmente se quiser. Deste modo, a C não impõe qualquer ponto de vista. Além disso, possui todas as características de uma linguagem de programação de uso geral, como recursividade, procedimentos como tipos de dados de primeira classe e muito mais.

No âmbito do nosso trabalho concluímos, também, que não só existem muitas formas de programar, como existem várias soluções para o mesmo problema. Assim, diferentes algoritmos de programas podem solucionar o mesmo problema.

Referências Bibliográficas

Webliografia

- www.dsi.uminho.pt/disciplinas/PLP
- www.ceop.com.br/articles
- www.ead.cee.ufmg/cuso/c
- www.portalc.acif.com.br

Anexos

Anexo 1

Estrutura básica de um programa C

-

Conjunto de caracteres do C

A Linguagem C distingue as letras maiúsculas das minúsculas. Os caracteres utilizados são os seguintes:

- A - Z
- a -z
- 0 - 9
- space . , : ; ' $ "
- # % & ! _ {} [] () < > |
- + - / * =

Formato do programa C

Todo programa C consiste de uma ou mais funções. Estas funções são semelhantes aos procedimentos em Pascal. Existe uma função principal, obrigatória em qualquer problema chamada *main()*. Esta é a primeira função que será executada em um programa. Contém, em essência, um esboço de mais alto nível do que o programa faz. A partir dela outras funções serão chamadas.

Além das funções, um programa C deverá ter declarações de variáveis. Todas as variáveis devem ser alocadas antes da sua utilização.

Com funções, variáveis e comandos próprios da linguagem (palavras reservadas são todas em letras minúsculas) pode-se construir qualquer programa. Entretanto, a definição de funções permite que estas sejam reutilizadas em situações semelhantes em novos programas. A partir daí existe uma biblioteca padrão de funções que realizam as tarefas mais comuns (por exemplo - entrada e saída). Além das funções padrão que estão implementadas em qualquer compilador, existem funções específicas de cada ambiente. O código destas funções será adicionado ao programa no momento de "linkar".

A seguir é apresentado o formato geral de um programa C.

```
Diretivas de pré-processamento
Declarações globais
main()
{
   variáveis locais à função main ;
   instruções da função main ;
}
```

```
f1()
{
   variáveis locais à função f1 ;
   instruções da função f1 ;
}
f2()
{
   variáveis locais à função f2 ;
   instruções da função f2 ;
}
.
.
.
etc
```

Comentário

Os comentários podem estar em qualquer parte do programa. Iniciarão sempre após os símbolos "/*" e terminarão com os símbolos "*/".

Directiva de pré-compilação - #include

Esta directiva inclui um arquivo de cabeçalho, que tem sempre a extensão ".h". Este arquivo apresenta definições utilizadas pelas funções de uma determinada biblioteca que será usada. No caso, o arquivo *stdio.h* é necessário para se utilizar a função *printf()*, que escreve na tela.

Execução do programa

O programa inicia pela primeira linha da função *main()* e termina na última linha da função *main()*. Portanto, a função printf() é executada e o programa termina. A função printf() é responsável por pegar os caracteres que estão delimitados por aspas e colocá-los na tela. Como ela faz isto não interessa ao programador. Este apenas precisa saber como passar os dados que quer jogar na tela para a função. O símbolo "\n" indica que a função deve pular passar para a próxima linha ao final da frase.

4. Variáveis

Como todas as linguagens de programação, C utiliza variáveis para armazenar valores. Todas as variáveis de um programa C devem ser declaradas antes de serem usadas. Isto é necessário para que seja alocada memória para as mesmas. Existem diferentes tipos de variáveis em C e o tamanho destes tipos podem variar de acordo com o processador e a implementação do compilador.

Tipos básicos

- char -1 byte. Este tipo é utilizado para se guardar valores definidos dentro da tabela ASCII (-127 a 127).
- int - número inteiro. Geralmente apresenta o tamanho natural da palavra do processador.
- float - O tipo float permite representar números com ponto flutuante utilizando 32 bits.
- double - Permite representar números com ponto flutuante utilizando 64 bits.
- void - Este tipo declara explicitamente que uma função não retorna nenhum valor.

Tipos modificados

- unsigned - a palavra reservada unsigned indica que não haverá representação de sinal do valor (todos os valores tornam-se positivos).
- long - este modificador aumenta a capacidade de um tipo. Os tipos abaixo permitem as seguintes representações:
- short - inteiro curto

Tipo	Num de bits	Formato para leitura com scanf	Intervalo	
			Inicio	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
signed char	8	%c	-128	127
int	16	%i	-32.768	32.767
unsigned int	16	%u	0	65.535
signed int	16	%i	-32.768	32.767
short int	16	%hi	-32.768	32.767
unsigned short int	16	%hu	0	65.535
signed short int	16	%hi	-32.768	32.767
long int	32	%li	-2.147.483.648	2.147.483.647
signed long int	32	%li	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
float	32	%f	3,4E-38	3.4E+38
double	64	%lf	1,7E-308	1,7E+308
long double	80	%Lf	3,4E-4932	3,4E+4932

Nomes de varáveis devem seguir algumas regras:

- podem-se utilizar letras e dígitos;
- o primeiro caractere deve ser uma letra ou "_";
- letras maiúsculas são diferentes de minúsculas;
- apenas 6 caracteres são significativos;
- deve ser diferente de uma palavra reservada.

Declaração de variáveis

tipo lista-de-variáveis;

Exemplos:

int i, j, k;

int x;

char letra;

char número;

As variáveis podem ser:

a) Variáveis locais

Estas variáveis são declaradas dentro de uma função. Também são conhecidas como variáveis automáticas. Estas variáveis podem ser manipuladas apenas dentro da função onde foram declaradas e existem apenas enquanto a função a que pertencem estiver sendo executada. A menos que seja especificado, o valor da variável é perdido entre cada chamada da função.

b) Parâmetros

Quando uma função usa argumentos, deve declarar as variáveis que receberão os valores dos argumentos. Essas variáveis se comportam como variáveis locais. A declaração destes argumentos ocorrem depois do nome da função dentro dos parênteses. Estas variáveis recebem os parâmetros "por valor".

c) Variáveis globais

Estas variáveis são reconhecidas pelo programa inteiro. São declaradas fora de qualquer função.

Especificadores de tipo de classe

Existem palavras reservadas que informam ao compilador como uma variável deve ser armazenada. São eles: *extern*, *static*, *register* e *auto*.

a) *extern*

Quando uma variável global tiver sido declarada em um módulo arquivo de um programa e for utilizada em um módulo que esteja em outro arquivo. A declaração no segundo deve ser feita utilizando-se o especificador extern. Isto avisa o compilador que esta variável está declarada em outro módulo que será "linkado" ao programa.

b) *static*

Este especificador é utilizado em variáveis locais para avisar que uma determinada variável é permanente apesar de ser reconhecida apenas dentro da função ou bloco em que foi declarada. No caso de variáveis globais, significa que a variável não é reconhecida em outro arquivo.

c) *register*

Aqui é especificado que o acesso à variável é o mais rápido possível. Só pode ser aplicado a variáveis locais e parâmetros de funções. Entretanto, não existe endereço de uma variável do tipo register.

Declaração de variáveis

Em geral, as variáveis podem ser inicializadas nas suas declarações. O formato da inicialização é o seguinte:

tipo nome_da_variável = constante ;

tipo nome_da_variável ;

tipo nome1, nome2, nome3, ...;

Exemplos:

int x = 0;

int total = 20;

```
char c = ‘A’;

        float k;

        double num;

        int a,b;
```

Atribuição de valores à variáveis

O símbolo utilizado para atribuir-se uma valor a uma variável é "=". O formato da instrução de atribuição é o seguinte:

Variável = Expressão;

A expressão pode ser outra variável, uma constante ou uma operação tal como soma, multiplicação, comparação, etc.

Exemplos:

```
a = b + c;

total = soma1 + soma2;

x = 10;
```

5. Constantes

Constantes são valores que são mantidos fixos pelo compilador.

Tipo de Dado	Exemplos de Constantes
char	'b' '\n' '\0'
int	2 32000 -130
long int	100000 -467
short int	100 -30
unsigned int	50000 35678
float	0.0 23.7 -12.3e-10
double	12546354334.0 -0.0000034236556

Numéricas

- decimal inteiros: 10, -10, -23
- decimais de ponto flutuante: 1.234, 4.3e-3F

- inteiros longos: 10000L, -34L
- hexadecimal: 0x10
- octal (037)

Caractere

Esta constante é representada entre apóstrofes: 'A', ‘%’. Caracteres especiais são representados utilizando-se a barra \. Alguns deles são os seguintes:

BS - ‘\b’

FF - ‘\f’

LF - ‘\n’

Nulo - ‘\0’

Cadeia de caracteres (string)

Esta constante é representada entre aspas: "Otávio".

6. Operadores

Atribuição

nome_da_variável = expressão ;

Em C ocorre a conversão automática de tipos.

Aritméticos

+ : adição

- : subtração

* : multiplicação

/ : divisão

% : módulo da divisão (resto)

Incremento ou decremento

Os operadores ++ e -- favorecem a produção de um código mais rápido. Entretanto deve-se tomar cuidado para as duas formas de posicionamento do operador.

- Pré-fixados: incrementa antes de usar.
- Pós-fixados: incrementa após usar.

Relacionais e lógicos

Em C, verdadeiro corresponde a qualquer valor diferente de 0. As operações relacionais devolvem valores 0 para falso e 1 para verdadeiro.

Operadores relacionais

- >: maior que
- >= : maior ou igual que
- < : menor que
- <= : menor ou igual que
- == : igual
- != : diferente

A precedência destes operadores é menor do que a dos operadores aritméticos.

Operadores lógicos

&& : AND

|| : OR

! : NOT

Bit-a-bit

Estas operações podem ser utilizadas apenas em variáveis dos tipos *char* e *int*.

& : AND

| : OR

^ : XOR

<< : deslocamento à esquerda

>> : deslocamento à direita

~ : complemento de um

7. Função printf()

printf(*string , lista_de_argumentos*)

Esta função coloca dados formatados na tela. O protótipo está em stdio.h e a função devolve o número de caracteres escritos, ou um valor negativo se ocorrer um erro.

A string de controle apresenta caracteres que serão colocados na tela e comandos de formato que definem a maneira como os argumentos serão escritos. Estes comandos começam sempre por % (porcento) e são seguidos pelos códigos dos formatos.

Símbolo	**Substituição**
%c	caractere
%d	Inteiro decimal com sinal
%i	Inteiro decimais com sinal
%e	Notação científica
%E	Notação científica
%f	Ponto flutuante decimal
%g	Usa %e ou %f (o mais curto)
%G	Usa %E ou %f (o mais curto)
%o	Octal
%s	String
%u	Inteiro decimal sem sinal
%x	Hexadecimal sem sinal (minúsculas)
%X	Hexadecimal sem sinal (maiúsculas)
%p	Apresenta um ponteiro
%n	Guarda o número de caracteres escritos
%ld, %lu, %lo, %lx, %li	Inteiro longo
%%	Escreve símbolo % (porcento)