Linguagem C - Resumão

Profª. Rose Yuri Shimizu

Programas C - Estrutura básica

TODA INSTRUÇÃO TERMINA COM ";"

```c
/*
    Estrutura básica de um programa em C
    stdio.h : cabeçalho das funções de entrada e saída
*/
#include <stdio.h> //inclusão da biblioteca

//função principal
int main() {
    //declaração de variáveis
    int a; //toda instrução termina com ;
    int A; //case sensitive
    char letra;
    float real1;
    double real2;

    return 0; //padrao C: return 0 sucesso e nao-zero erro.
}
```

Declaração de variáveis - Tipos de dados - Intervalos

Tipo	Tamanho	Intervalo $\approx$
char	8 bits	-128 $\rightarrow$ 127
short	16 bits	-32 mil $\rightarrow$ 32 mil
unsigned short	16	0 $\rightarrow$ 65 mil
int	32 bits	-2 bilhões $\rightarrow$ 2 bilhões
unsigned int	32	0 $\rightarrow$ 4 bilhões
long	64 bits	-9 quintilhões $\rightarrow$ 9 quintilhões
long long	64 bits	-9 quintilhões $\rightarrow$ 9 quintilhões
unsigned long	64	0 $\rightarrow$ 18 quintilhões
float	32 bits	$-3.4x10^{38} \rightarrow 3.4x10^{38}$
double	64 bits	$-1.7x10^{308} \rightarrow 1.7x10^{308}$

- float: 6 dígitos de precisão decimal (restante é aproximação)
- double: 15 dígitos de precisão decimal (restante é aproximação)
- Exemplo:
  - float 1**123456**123456123456 = 1**123456**104810938368
  - double 1**1234561234561234**56 = 1**1234561234561234**392

Declaração de variáveis - Tipos de dados - Memória

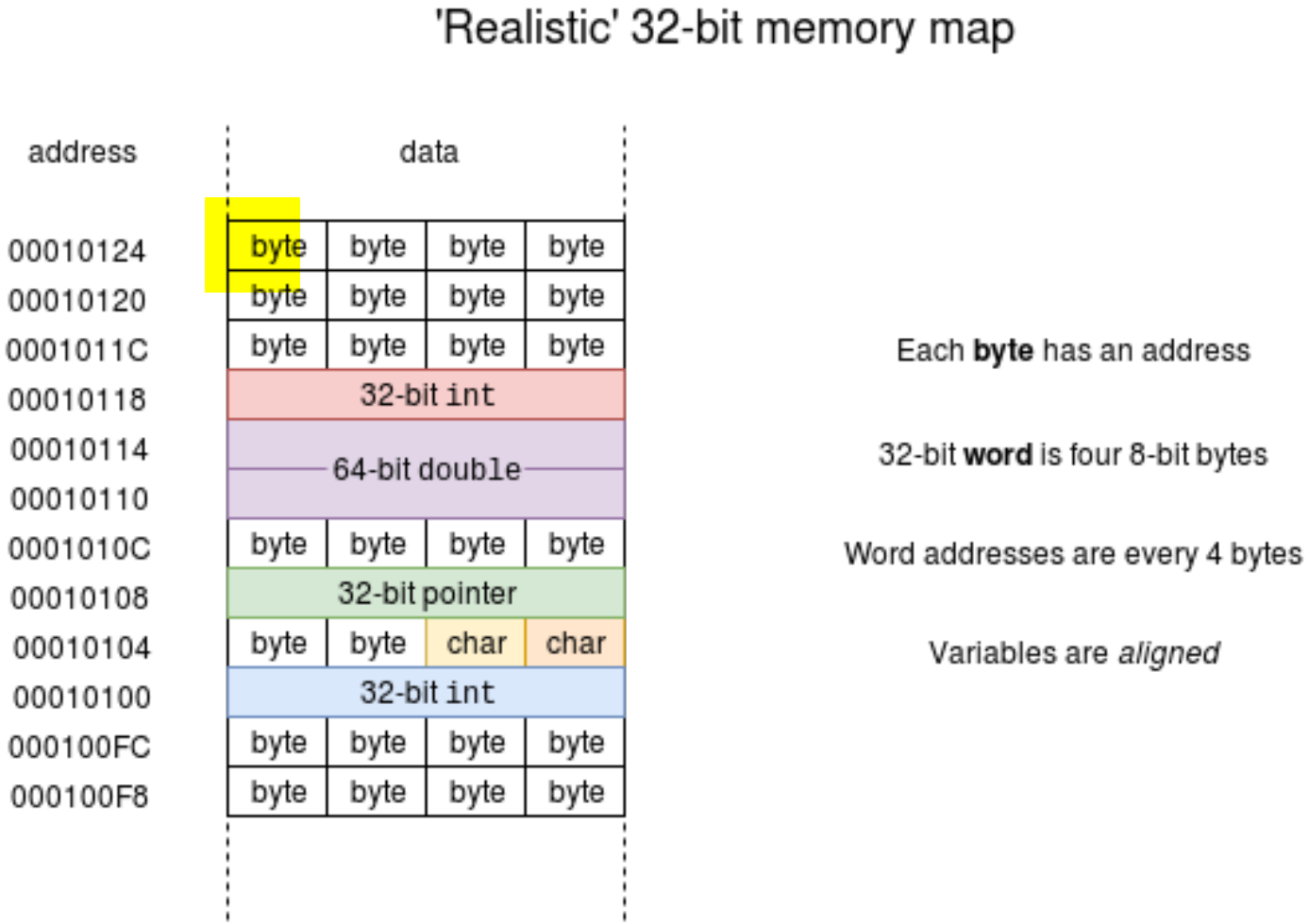

Figura: https://xerxes.cs.manchester.ac.uk/comp251/kb/images/variables.png

Inicialização de variáveis (atribuição)

```c
int main()
{
    //Declaração
    int pera;
    char qualidade;
    float peso;

    //Inicialização
    pera = 3;
    qualidade = 'A';
    peso = 0.653;

    //Inicialização na declaração
    int maca = 1;

    //Várias declarações
    int a, b=9, c;
}
```

Função **printf()**

- Imprimir mensagens e valores de variáveis

```
1 int X = 3;
2 int Y = 2;
3
4 //                  |------------------|
5 printf("X vale %d e Y vale %d\n", X, Y);
6 //                                |--------|
7
8 printf("Olá Mundo!\n");
```

- Tudo que está entre aspas dupla será impresso na tela
- **Especificadores de formato**: tipo do dado que será substituído pelo valor da variável
- **man 3 printf**

Tipo	Especificador
int	%d
long	%ld
char	%c
float	%f
double	%lf

Função **scanf()**

```c
#include <stdio.h>

int main(){
    //Declaração das variáveis
    float y;
    int x;

    //Inicialização pela entrada padrão (teclado)
    scanf("%f %d", &y, &x); //& : endereço da variável

    //Saída
    printf("Você digitou %f %d\n", y, x);
    return 0;
}
```

```c
scanf("%d %d",&a,&b);
          //^ nao pode ter espaco entre o ultima entrada e o
     fecha aspas
```

Operadores Aritméticos

- Operador de Atribuição =:
  - (soma = a + b) $\neq$ (a + b = soma)
- Operadores Aritméticos:

Operador	Exemplo
+ (adição)	x = y + 1
- (subtração)	x = y - 1
/ (divisão)	x = y/z
* (multiplicação)	x = y*10
% (resto)	x = y%4
++ (mais um)	++x $\rightarrow$ x = x + 1
−− (menos um)	−−x $\rightarrow$ x = x - 1

Operadores Relacionais

- Compara a relação entre dois valores ou expressões;
- Tipos
  - $<$ Menor
  - $>$ Maior
  - $<=$ Menor ou igual
  - $>=$ Maior ou igual
  - $==$ Igual
  - $!=$ Diferente
- Em C: $==$ é diferente de $=$
  - $=$ é atribuição, $==$ é comparação
- O resultado é 1 (verdade) ou 0 (falso)

Operadores Lógicos

- Compara a relação entre dois valores ou expressões
- Lógica proposicional através da álgebra booleana (George Boole, matemático inglês)
- Operadores lógicos em C
  - && E : "todos"
  - || OU : "algum"
  - ! Não : inverso

Operadores Lógicos

- P: Está chovendo
- Q: Solo está seco
- Equivalência lógica em C:

```c
int chuva = 1;
int seco = 1;

int ativar_sensor = !chuva && seco;
```

- Tabela verdade (2^2 linhas)

p	q	$\neg p \wedge q$
V	V	F
V	F	F
F	V	V
F	F	F

Expressão aritmética, relacional e lógica: precedências

1. ()
2. !
3. * / %
4. + −
5. < > <= >=
6. == ! =
7. &&
8. ||
9. =

- a || ! (b && c)

Controle de fluxo : IF

```c
#include <stdio.h>
int main(){
  char c;
  scanf("%c",&c);

  if(c=='a'|| c=='A'|| c=='e'|| c=='E'||
     c=='i'|| c=='I'|| c=='o'|| c=='O'||
     c=='u'|| c=='U')
  {
    if(c>='A' && c<='U') //intervalo: tabela ASCII
        printf("%c:vogal maiúscula\n",c);
    else
        printf("%c:vogal minúscula\n",c);
  }
  else if(c>='0' && c<='9')
  {
    printf("%c:digito\n",c);
  }
  else
  {
    printf("%c:consoante\n",c);
  }
  return 0;
}
```

Estrutura de Repetição: WHILE e FOR

```c
//n! = n*(n-1)*(n-2)*(n-3)*...*3*2*1
//declarar
int fat, n, i;

//inicializar
scanf("%d", &n);
fat = 1;
i = n;

//repetir - verificar
while(n >= 1) {
    fat = fat * n;

    //atualizar
    n--;
}

printf("%d! = %d\n", i, fat);

for(fat=1; i >= 1; i--) {
    fat = fat * i;
}
```

Estrutura de Repetição: WHILE e FOR

```
inicializações
while(condições)
{
    ...
    atualizações
}
```

```
for( inicializações; condições; atualizações )
{
    ...
}
```

Estrutura de Repetição: leituras seguidas

```c
scanf("%d", &a);
while(a > 0){
    printf("%d\n", a);
    scanf("%d", &a);
}
```

```c
while(scanf("%d", &a), a > 0){
    printf("%d\n", a);
}
```

```c
while(scanf("%d", &a) && a > 0){
    printf("%d\n", a);
}
```

```c
          // | espaço para leituras seguidas de caracteres
while(scanf(" %c", &a)){
    printf("%c\n", a);
}
```

Função e Procedimento

- Definido por um conjunto de instruções que fazem uma tarefa específica
- Somente é executada ao ser utilizada/chamada/invocada
- Funções podem utilizar outras funções
- Exemplos: printf, scanf, strlen, strcmp, etc.
- Componentes:

```
1 tipo_retorno nome(parametros)
2 {
3     ...
4     tipo_retorno a;
5     ...
6     return a;
7 }
```

  - **tipo_retorno**: o que a função devolve para o código que chamou:
    - ★ int, double, float, char;
    - ★ void (vazio, nada);
  - **nome_funcao**: o nome utilizado para chamar a função;
  - **parametros**: lista de declarações de variáveis separadas por vírgulas;
  - **return**: palavra reservada que indica o que será retornado;

Retorno de funções

Retorno do scanf

- A função **scanf** retorna:
  - man scanf
  - **Número de itens** de entrada **combinados e atribuído** com sucesso;
  - O valor **EOF** é retornado se o final da entrada é alcançado antes da primeira leitura ou falha de correspondência ou erro de leitura.

```
1 while(scanf("%d", &d) != EOF) { } //ctrl+d = end of file
2 while(scanf("%d", &d) == 1)   { }
3
```

```
1 int d;
2 while(scanf("%d", &d) != EOF) //ctrl+d = end of file
3 {
4     printf("%d\n", d);
5 }
```

Retorno de funções

Retorno do printf

- A função **printf** retorna:
  - man 3 printf
  - Sucesso: **número de caracteres impressos**
  - Erro: **número negativo**

```
1 int a = printf("alo\n");
2 printf("%d\n", a);
3 /*
4 Saida
5 alo
6 4
7 */
8
```

```c
int somas() {
    int a, s=0;
    while(scanf("%d", &a) != EOF)
        s=s+a;
    return s;
}

void imprime() {
    printf("Ola mundo!\n");
}

float obtem_valor() {
    float valor;
    printf("Entre um valor:");
    scanf("%f", &valor);
    return valor;
}

void imprime_olas() {
    int a;
    while(scanf("%d", &a) == 1 && a>0){
        if(a>1000) return;
        while(a-->0)
            printf("Ola mundo!\n");
    }
}
```

Parâmetros: passando valores para as funções

- Passar argumentos (valores) através dos parâmetros (variáveis);

```
1  int acumulador(int n, int m) { //parâmetros
2      int s=0, i;
3      while(n){
4          scanf("%d", &i);
5          if(i<m)
6              s+=i; //s=s+i;
7          n--;
8      }
9      return s;
10 }
11
12 int funcao() {
13     int n, max;
14     scanf("%d %d", &n, &max); //entrada 10 100
15     return acumulador(n, max); //argumentos n e max
16                                //valor de n no retorno?
17 }
```

Passagem de argumentos por valor

- **Cópia** do valor original
- Não altera a variável original
- Variável da função $\neq$ variável original

```c
void teste (char k) {
    printf("%c\n", k); //podemos utilizar o valor de k
    k='a';             //podemos alterar o valor de k
    printf("%c\n", k); //a
}

int main() {
    teste('p'); //passando o valor direto

    char k = 'b';

    teste(k); //passando uma copia do valor de k
    printf("%c\n", k); //b - nao altera o valor de k

    return 0;
}
```

Passagem de argumentos por referência

- **Passar o endereço (referência) de uma variável**
- As **alterações** são feitas **diretamente no endereço** da variável original
- Ponteiro:
  - Variável especial que armazena endereços
  - Identificado pelo * (asterisco)
  - Armazena o endereço do local onde está o conteúdo

```
1  void f1(int x, int *i){
2     printf("%d %d\n", x, *i); //1 0
3     *i = 4; //alterando o conteúdo no endereco apontado por i
4     x = 5;  //alterando o conteúdo da variável x
5  }
6
7  int main(){
8     int a=1, c=0;
9
10    f1(a, &c); //conteudo de a e endereco de c
11    printf("%d %d\n", a, c); //1 4
12    return 0;
13 }
```

Função e Procedimento

```c
//função
void soma(int a, int b)
{
    int c = a+b;
    printf("%d\n", c);
}

//procedimento
int menu(){
    char ch;
    printf("Escolha Sim (s) ou Não (n):\n");
    scanf("%c", &ch);
    return ch;
}
```

Definindo a função: antes da "invocadora"

- Dever ser definida/criada antes de quem invoca a função

```
1  float f1() {
2      float a, b;
3      scanf("%f %f", &a, &b);
4      if(b!=0) return a/b;
5      else return 0;
6  }
7
8  int main() {
9      char x;
10     scanf("%c", &x);
11
12     while(x!='s') {
13         printf("%f\n", f1());
14         scanf(" %c", &x);
15     }
16     return 0;
17 }
```

Definindo a função: depois da "invocadora"

- "Tudo que usa, tem que declarar";
- Onde declarar?
  - Antes da função (antes da main) ou em arquivos cabeçalho .h
- Como declarar? Através de seu protótipo:

```c
#include <stdio.h>
//prototipo da funcao soma()
//tipo_retorno nome_funcao();
float soma(float, float); //ponto e virgula no final

int main() {
    printf("%f\n", soma(1.2, 3.0));
    return 0;
}
float soma(float a, float b) {
    return a+b;
}
```

Definindo a função: depois da "invocadora"

```c
//prototipos
void imprime();            //procedimento
void soma(int a, int b); //void soma(int, int);
int quad(int n);           //int quad(int);

int main() {
    imprime();
    quad(10);

    return 0;
}

int quad(int n) {
    return n*n;
}

void imprime() {
    printf("Ola mundo!\n");
}

void soma(int a, int b) {
    int c=a+b;
    printf("%d\n", c);
}
```

Exemplo

Faça uma função que leia, repetidamente até EOF, a quantidade de pessoas que entra e sai, respectivamente, de um elevador e imprima quantas pessoas restaram.

```c
#include <stdio.h>
//declaracao do prototipo
void elevador();

int main() {
    elevador();
    return 0;
}

//definicao da funcao elevador
void elevador() {
    int entra, sai, e=0;
    while( scanf("%d%d", &entra, &sai) != EOF ) {
        e = e + entra - sai;
    }
    printf("%d\n", e);
}
```