Curso de C/C++ Avançado

Aula 3 – E/S, Tipos Avançados de Dados

Allan Lima – http://allanlima.wordpress.com

creative commons
COMMONS DEED

- ***Você pode:***
  - **copiar, distribuir, exibir e executar a obra**
  - **criar obras derivadas**
  - **fazer uso comercial da obra**
- ***Sob as seguintes condições:***
  - **Atribuição. Você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante.**
  - **Compartilhamento pela mesma Licença. Se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta.**
  - **Para cada novo uso ou distribuição, você deve deixar claro para outros os termos da licença desta obra.**
  - **Qualquer uma destas condições podem ser renunciadas, desde que Você obtenha permissão do autor.**
- ***Veja aqui a licença completa***

cit

A função printf()

- ***Protótipo:***
  - **int printf (char *str,...);**
  - ***str* é chamada de string de controle e possui dois tipos de componentes:**
    - Os caracteres a serem impressos
    - Os comandos de formato
  - **As reticências indicam que o número de parâmetros é variável**
  - **Retorna o número de argumentos escritos**

Códigos de Formato

Código	Formato
%c	Caractere
%d	Número decimal (inteiro)
%e	Número em notação científica com o ‘e’ minúsculo
%E	Número em notação científica com o ‘e’ maiúsculo
%f	Ponto flutuante
%g	Escolhe automaticamente o melhor entre %f e %e
%G	Escolhe automaticamente o melhor entre %f e %e
%o	Número octal
%s	String
%u	Decimal sem sinal (unsigned)
%x	Hexadecimal com letras minúsculas
%X	Hexadecimal com letras maiúsculas
%%	Imprime um %
%p	Ponteiro

Exemplos

```
printf("Um %%%c %s", 'c', "char"); // Um %c
    char
printf("%X", 107); // 6B
printf("%f", 49.67); // 49.67
printf("%e", 49.67); // 4.967e1
printf("%d %o",10,10); // 10 12
```

Formatando a saída

- ***A formatação dos tipos é indicada logo após '%' e antes do campo do tipo:***

Código	Formatação	Saída
printf("%4d", 45);	No mínimo 4 caracteres	| 45|
printf("%-4d", 45);	Anterior, mas alinhado a esq.	|45 |
printf("%04d", 45);	Completa com zeros	|0045|
printf("%.2f", 75.777);	2 casas decimais	|75,78|
printf("%8.2f\n", 75.777);	Anterior, mas com 8 carac.	| 75,78|
printf("%-8.2f\n", 75.777);	Anterior, mas alinhado a esq.	|75,78 |

Exemplo

- ***exemploFormatação.c***

A função scanf()

- ***Protótipo:***
  - **int scanf (char *str,...);**
  - ***str* é chamada de string de controle e possui dois tipos de componentes:**
    - Os caracteres
    - Os comandos de formato
  - **Os parâmetros da função devem ser sempre ponteiros**

sscanf() e sprintf()

- *Funcionam, respectivamente, de forma similar à scanf() e printf()*
- *int sprintf (char *destino, char *controle, ...);*
  - Ao invés de escrever na saída padrão escreve em *destino*
- *int sscanf (char *origem, char *controle, ...);*
  - Ao invés de ler da entrada padrão lê da string *origem*
- *São muito úteis para conversão de tipos*

Exemplo

```
#include <stdio.h>
int main() {
    int a = 10;
    float f;
    char s[20] = "15.05";

    sscanf(s, "%f", &f); // converte s para um float
    printf("%.2f\n", f);
    sprintf(s, "%d", a); // converte a para uma string
    printf("%s\n", s);
    // também podem escrever mais de um parâmetro
    sprintf(s, "%d %f", a, f); // apaga s para concatenar a e f
    printf("%s\n", s);
    return 0;
}
```

E/S

- ***Funções:***
  - **int getche(void);**
    - Lê, imprime e retorna um caractere pressionado
  - **int getch(void);**
    - Lê e retorna um caractere pressionado
  - **int getchar(void);**
    - Lê e retorna um caractere após o usuário digitar ENTER
  - **int putchar(int c);**
    - Imprime um caractere e retorna o caractere impresso

Arquivos

- ***Arquivos em C podem representar tanto arquivos do sistema quanto dispositivos periféricos***
- ***Podemos criar ponteiros para arquivos:***
  - **FILE *f;**
  - **FILE foi definido usando o comando *typedef***
- ***Arquivos pré-definidos:***

Arquivo	Descrição
stdin	Dispositivo de entrada padrão (normalmente o teclado)
stdout	Dispositivo de saída padrão (normalmente o monitor)
stderr	Dispositivo de saída de erro padrão (normalmente o monitor)
stdaux	Dispositivo de saída auxiliar (normalmente a porta serial)
stdprn	Dispositivo de impressão padrão (normalmente a porta paralela)

Abrindo e Fechando Arquivos

- *FILE *fopen(char *nomeDoArquivo, char *modo);*
  - Abre um arquivo e retorna um ponteiro para ele
  - Exemplo: FILE *fp = fopen(“L1QD.in”, “r”);
  - Se algum erro ocorrer a constante NULL é retornada
- *int fclose(FILE *fp);*
  - Fecha o arquivo passado como parâmetro
  - Retorna 0 se o arquivo foi fechado corretamente
  - Retorna a constante EOF se algum erro ocorreu
  - É muito importante fechar todos os arquivos que o seu programa abriu, isto evita a perda dos dados armazenados no *buffer*

Modos de Abertura

String	Modo
“r”	Abre um arquivo de texto para leitura. O arquivo deve existir antes de ser aberto.
“w”	Abre um arquivo de texto para escrita. Se o arquivo não existir ele será criado, mas se existir o seu conteúdo será apagado.
“a”	Similar ao anterior, porém se o arquivo já existir, os dados escritos serão adicionados no fim do arquivo
“r+”	Abre um arquivo para leitura a gravação.
“w+”	Cria um arquivo de texto para leitura e gravação. Se o arquivo já existir, o conteúdo anterior será destruído. Se não existir, ele será criado.
“a+”	Abre um arquivo de texto para leitura e gravação. Se o arquivo já existir, os dados serão adicionados ao seu fim. Se não existir, será criado um arquivo vazio.

Modos de Abertura

- ***O modo de abertura de um arquivo pode ser binário ou texto (padrão)***

string	modo
“t”	Abre o arquivo no modo de texto (O caractere Ctrl-Z indica o fim do arquivo)
“b”	Abre o arquivo em modo binário

- Exemplos:

  "rb", "wt", "r+t", "w+b“, "r+", "wb+"

ES em Arquivos

- *int getc(FILE *fp);*
  - Lê um caractere de fp
  - Retorna EOF se houver erro
- *int putc(int c, FILE *fp);*
  - Escreve um caractere em fp
  - Retorna EOF se houver erro
- *int feof(FILE *fp);*
  - Indica o fim de um arquivo
  - Retorna não-zero se o arquivo chegou ao fim e zero caso contrário

Exemplo

- ***exemplosArquivos1.c***

ES em Arquivos

- *char *fgets(char *str, int tamanho, FILE *fp);*
  - Lê no máximo *tamanho* caracteres de *fp* e guarda em *str*
  - Se a função encontrar '\n' ou EOF ela para
- *char *fputs(char *str, FILE *fp);*
  - Imprime uma string no arquivo
- *int ferror(FILE *fp);*
  - Retorna zero se não houver erros na última operação realizada com um arquivo
  - Caso contrário retorna não-zero

Exemplo

- ***exemplosArquivos2.c***

ES em Arquivos

- ***int fwrite(void *buffer, int numBytes, int qtd, FILE *pf);***
  - ***buffer* é o local que contém os dados a serem gravados**
  - ***numBytes* é o tamanho da unidade a ser gravada**
  - ***qtd* é o número de unidades a serem gravadas**
  - ***pf* é o arquivos do no qual se deseja gravar os dados**
  - **O retorno é o número de unidades efetivamente gravadas**
  - **Exemplo:**

```
FILE *pf = fopen("meuArquivo", "w");
int mat[] = { 1, 2, 3, 4 };
fwrite(mat, sizeof(int), 4, pf);
```

ES em Arquivos

- ***int fread(void *buffer, int numBytes, int qtd, FILE *pf);***
  - ***buffer*** **é o local aonde os dados lidos do arquivo serão guardados**
  - ***numBytes*** **é o tamanho da unidade a ser lida**
  - ***qtd*** **é o número máximo de unidades a serem lidas**
  - ***pf*** **é o arquivo do qual se deseja ler**
  - **O retorno é o número de unidades efetivamente lidas**
  - **Exemplo:**

```
FILE *pf = fopen("meuArquivo", "r");
int mat[12];
fread(mat, sizeof(int), 10, pf);
```

Exemplo

- ***exemploArquivos3.c***

ES em Arquivos

- *int fseek (FILE *pf, long numBytes, int ori);*
  - Move a posição corrente de leitura ou escrita em um arquivo
  - pf = o arquivo que se deseja percorrer
  - numBytes = O número de bytes a serem movidos
  - ori = O ponto de partida:

Nome	Valor	Significado
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Ponto corrente
SEEK_END	2	Fim do arquivo

ES em arquivos

- *void rewind(FILE *fp);*
  - Move o cursor para o inicio do arquivo
- *int remove(char *nomeDoArquivo);*
  - Apaga um arquivo
- *int fprintf(FILE *pf, char *str, ...);*
  - Similar a printf() porém escreve em um arquivo
- *int fscanf(FILE *pf, char *str, ...);*
  - Similar a scanf() porém lê de um arquivo

ES em Arquivos

- ***FILE *freopen(***

  const char *nomeArquivo,

  const char *modo,

  FILE *fp

  );

  - **Fecha *fp***
  - **E abre um arquivo e o associa ao ponteiro passado como parâmetro**

Exemplo

- *exemploFreopen.c*

Estruturas

- *Agrupam diversas variáveis em um único tipo*
- *São tipos de dados criados pelo programador*
- *Sintaxe:*

```
struct nomeEstrutura {
        tipo1 nome1:numeroBits1;
        ...
        tipoN nomeN:numeroBits1;
} listaDeVariáveis;
```

Exemplos

```
struct Endereco {
    char rua[50];
    int numero;
    char bairro[20];
    char *cidade;
    char siglaEstado[3];
    long int cep;
};
```

Os atributos não podem ser inicializados dentro da própria estrutura:

```
struct S {
   float f = 2.5f; // ERRADO!!!
};
```

Estruturas

- ***Podemos restringir o número exato de bits que cada campo possui***
- ***Exemplo:***

```
struct Exemplo {
    int a:3;
    char c:4;
    long long l:40;
};
```

Ponteiros e Estruturas

- ***Estruturas podem conter ponteiros para outras estruturas***
- ***Exemplo:***

```
struct Pessoa {
    char nome[100];
    char telefone[14];
    struct Endereco *endereco;
};
```

Exemplos

```
struct Cidade {
    char *nome;
    long populacao;
};
```

```
struct Estado {
    char *nome;
    char sigla[3];
    struct Cidade *cidades;
    int numeroCidades;
} pernambuco; // variável
```

Manipulação

- ***O acesso aos membros de uma estrutura é feito através dos operadores:***

. (Ponto)

```
struct Cidade c;
c.populacao = 10;
```

->

```
struct Cidade *c = malloc(sizeof(struct Cidade));
c->populacao = 10;
// equivalente a  (*c).populacao
```

Exemplo

- *exemploStruct1.c*
- *exemploStruct2.c*

Cópias de Estruturas

```
struct Cidade C1, C2;

C1.nome = "Sao Paulo";
C1.populacao = 15000000;

// cuidado ao fazer cópias de estruturas
C2 = C1;
```

Cópias de Estruturas

C1	
População	15000000
Nome	

C2	
População	15000000
Nome	

São Paulo

O que acontece quando fazemos:
strcpy(C1.nome, “Recife”);
???

Cópias de Estruturas

C1	
População	15000000
Nome	

C2	
População	15000000
Nome	

Recife

C2.nome também é alterado!

Uniões

- *Permitem criar uma única localização de memória onde podem estar armazenadas variáveis diferentes*
- *Sintaxe:*

```
union nomeDaUniao {
    tipo1 nome1;
    ...
    tipoN nomeN;
} listaDeVariaveis;
```

Exemplo

```
#include <stdio.h>

union MinhaUniao {
    long i;
    float l;
};

int main() {
    union MinhaUniao
     uniao;

    uniao.i = 10;
    printf("%d\n", uniao.i);
    uniao.l = 45.4f;
    printf("%f\n", uniao.l);
    return 0;
}
```

O compilador aloca o tamanho do maior componente da união para permitir o uso de qualquer um destes

Enumerações

- *Permitem ao programador restringir os valores de uma variável pode assumir*
- *Sintaxe:*

  enum nome { listaDeValores } listaDeVariáveis;

- *Exemplo:*

```
enum tiposTelefone {
    celular, residencial, comercial
};
```

Exemplo

```
#include <stdio.h>

enum Linguagens {
    Java, C, Pascal, Assembly, Haskell
} minhaLinguagem;

int main() {
    enum Linguagens outraLinguagem = Java;
    minhaLinguagem = C;
    printf("Valor de Java: %d\n", outraLinguagem); // Imprime 0
    printf("Valor de C: %d\n", minhaLinguagem); // Imprime 1

    return 0;
}
```

Emumerações

- *A cada valor da enumeração é atribuído um número*
- *Por default o 1° é 0, o 2° é 1, ...*
- *Mas podemos modificar isto!*
- *Exemplo:*

```
enum Linguagens {
    Java = 0, C = 10000, Pascal = 10, Assembly = 5, Haskell = 9
};
```

typedef

- ***Permite ao programador definir um novo nome para um tipo criado anteriormente***
- ***Sintaxe:***

  typedef nomeAtual novoNome

- ***Exemplo:***

  exemploTypedef.c

Exercícios

1) Implemente um programa que escreve o seu nome completo em um arquivo e depois o lê do mesmo arquivo.

Exercícios

2) Crie um programa que lê os nomes dos alunos e as suas notas de um arquivo de texto, os valores lidos devem ser mostrados em forma de tabela. Os nomes e notas devem ser guardados em uma matriz alocada dinamicamente. Exemplo para o arquivo de entrada:

```
4
Allan Lima
8.5
Fulano Silva
6.7777
Cicrado Torres
9.2
Beltrano Oliveira
6.3
```

Exercícios

3) Crie as seguintes estruturas:

```
struct Node { // Nó da pilha
    int value;
    struct Node *next;
};

struct Stack { // Pilha
    struct Node *first;
};
```

Exercícios

- ***Defina também as funções:***
  - **void push(struct Stack *stack, struct Node *node);**
    - insere *node* no topo de *stack*
  - **struct Node *pop(struct Stack *stack);**
    - Remove e retorna o primeiro elemento de stack
- ***Não se esqueça de desalocar a memória no final do programa***
- ***Sinta-se à vontade para criar funções auxiliares***

Referências

- ***Matos, P. A. & Carvalho, G. H. P. - A Linguagem de Programação C***
- ***Curso de C da UFMG***
  - http://ead1.eee.ufmg.br/cursos/C/
- ***Algumas Notas sobre Programação em C***
  - http://paginas.fe.up.pt/~apm/C_tut/Cap_7.htm
- ***Standart C***
  - http://www.ccs.ucsd.edu/c/
- ***Slides de Gustavo (ghcp@cin.ufpe.br) do Curso C/C++***