CENTRO DE CIÊNCIA E TECNOLOGIA
LABORATÓRIO DE CIÊNCIAS MATEMÁTICAS
UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE

Aplicação de Listas: Uma Lista de Filmes

Disciplina: Estrutura de Dados I

Prof. Fermín Alfredo Tang Montané

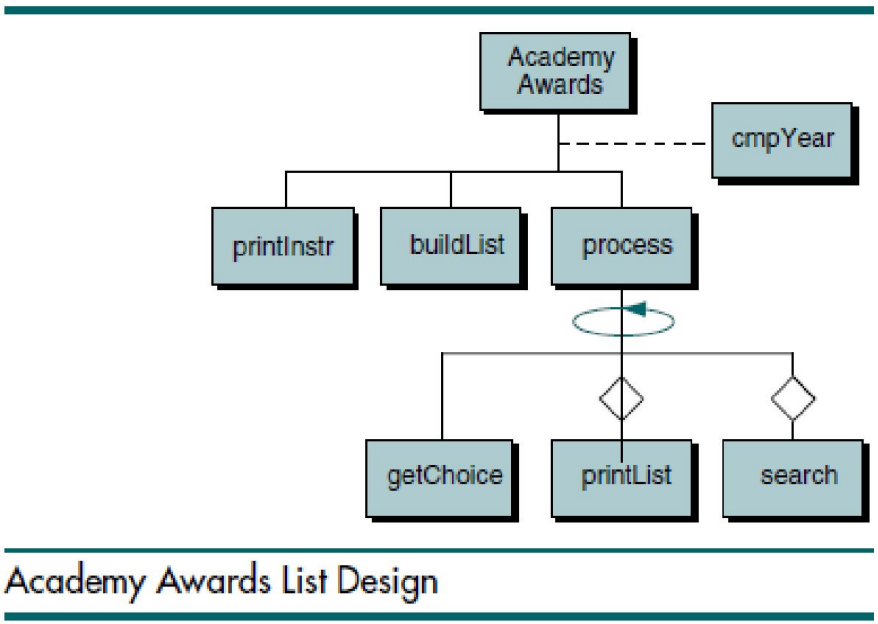
Curso: Ciência da Computação

Uma Lista de Filmes
Descrição

- Descreve-se uma aplicação simples para a construção de uma lista que armazene os dados de filmes ganhadores do premio Oscar.
- Em cada nó da lista armazena-se os seguintes dados: ano, nome do filme e nome do diretor.
- A lista é criada a partir da leitura de um arquivo e pode ser consultada pelo usuário.
- A lista usa uma função de comparação com base no ano do filme.

Listas Aplicação
Dependência entre as funções

- A dependência entre as funções da aplicação para a criação e consulta de uma lista de filmes é mostrada na figura.



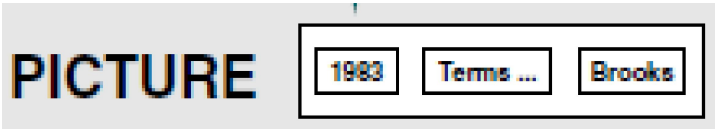
Academy Awards List Design

Listas Aplicação
Estrutura

- A aplicação de lista de filmes premiados utiliza uma estrutura que armazena os seguintes dados:
  - year (ano);
  - picture (nome do filme);
  - director (nome do diretor).
- Define o tipo PICTURE para denotar o tipo dessa estrutura.

P5-16.h – Estrutura para armazenar dados do filme

```
1  /*Data Structure for Academy Awards
2       Written by:
3       Date:
4  */
5
6  const short STR_MAX = 41;
7
8  typedef struct
9    {
10    short   year;
11    char    picture [STR_MAX];
12    char    director[STR_MAX];
13   } PICTURE;
```



Listas Aplicação
Função main()

- A aplicação para criação de uma lista de filmes pode ser descrita em três partes:
  - Inclusão de bibliotecas, dentre elas: P5-16.h que define a estrutura do nó e linkListADT.h que define o TAD Lista.
  - Declaração do protótipo das funções da aplicação.
  - A função main() que realiza o seguinte:
    - Impressão de instruções, printInstr();
    - Criação da lista, builtList();
    - Processamento das consultas do usuário process().

P5-17.c – Programa principal, Função main()

```
1  /*This program maintains and displays a list of
2    Academy Awards Motion Pictures.
3       Written by:
4       Date:
5  */
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <cType.h>
9  #include <stdbool.h>
10 #include "P5-16.h"          // Data Structure
11 #include "linkListADT.h"
12
13 //Prototype Declarations
14   void  printInstr (void);
15   LIST* buildList  (void);
16   void  process    (LIST* list);
17   char  getChoice  (void);
18   void  printList  (LIST* list);
19   void  search     (LIST* list);
20
21   int   cmpYear    (void* pYear1, void* pYear2);
22
23 int main (void)
24 {
25 // Local Definitions
26    LIST* list;
27
28 // Statements
29    printInstr ();
30    list = buildList ();
31    process (list);
32
33    printf("End Best Pictures\n"
34           "Hope you found your favorite!\n");
35    return 0;
36 } // main
```

Listas Aplicação
Função printInstr()

- Esta função imprime algumas instruções sobre o funcionamento da aplicação.
- Basicamente, deve-se inserir um ano e a aplicação mostrará o nome do filme ganhador do Oscar nesse ano e o nome de seu diretor.

P5-18.h – Função para impressão de instruções

```
1  /*================= printInstr ================
2    Print instructions to user.
3       Pre    nothing
4       Post   instructions printed
5  */
6  void printInstr (void)
7  {
8  //Statements
9    printf("This program prints the Academy Awards \n"
10     "Best Picture of the Year and its director.\n"
11     "Your job is to enter the year; we will do\n"
12     "the rest. Enjoy.\n");
13   return;
14 } // printInstr
```

Listas Aplicação
Função buildList()

- A função buildList() cria uma lista linear usando o TAD Lista, contendo informações de filmes.
- Os dados dos filmes são lidos a partir do arquivo "pictures.dat". Em cada linha do arquivo temos:
  - ano "filme" "diretor"
- Os dados de um filme são armazenados em uma estrutura tipo PICTURE.
- A variável pPic é um ponteiro a uma estrutura tipo PICTURE.
- Observe o uso da função createList() da TAD Lista, para criar a lista list.

P5-19.h – Função buildList()

```
1  /*=================== buildList ===================
2    Reads a text data file and loads the list
3    Pre    file exists in format: yy \t 'pic' \t 'dir'
4    Post   list contains data
5           -or- program aborted if problems
6  */
7  LIST* buildList (void)
8  {
9  //Local Definitions
10   FILE* fpData;
11   LIST* list;
12
13   short  yearIn;
14   int    addResult;
15
16   PICTURE* pPic;
17
18 //Statements
19   list   = createList (cmpYear);
20   if (!list)
21       printf("\aCannot create list\n"),
22              exit (100);
23   fpData = fopen("pictures.dat", "r");
24   if (!fpData)
25       printf("\aError opening input file\n"),
26              exit (110);
```

Listas Aplicação

Função buildList()

- O restante do código da função buildList() pode ser descrito da seguinte maneira:
- Enquanto for possível a leitura de um dado:
- Realiza-se a leitura do ano (yearIn);
- Aloca-se memória para uma estrutura PICTURE acessível pelo ponteiro pPic;
- Preenche-se os campos ano, filme e diretor na estrutura.
- Insere-se um novo nó na lista list armazenando o ponteiro pPic.
- Usa-se a função addNode() do TAD Lista para inserir o novo nó.

P5-19.h – Função buildList(), continuação...

```
27    while (fscanf(fpData, " %hd", &yearIn) == 1)
28       {
29        pPic = (PICTURE*) malloc(sizeof(PICTURE));
30        if (!(pPic))
31            printf("\aOut of Memory in build list\n"),
32                   exit (100);
33        pPic->year = yearIn;
34
35        // Skip tabs and quote
36
37
38        while ((fgetc(fpData)) != '"')
39           ;
40        fscanf(fpData, " %40[^\"]  %*c", pPic->picture);
41
42
43        while ((fgetc(fpData)) != '"')
44           ;
45        fscanf(fpData, " %40[^\"]  %*c", pPic->director);
46
47        // Insert into list
48        addResult = addNode (list, pPic);
49        if (addResult != 0)
50            if (addResult == -1)
51              printf("Memory overflow adding movie\a\n"),
52                       exit (120);
53            else
54               printf("Duplicate year %hd not added\n\a",
55                       pPic->year);
56        while (fgetc(fpData == '\n')
57              ;
58       } // while
59    return list;
60  } // buildList
```

Listas Aplicação
Notas sobre a Linguagem C

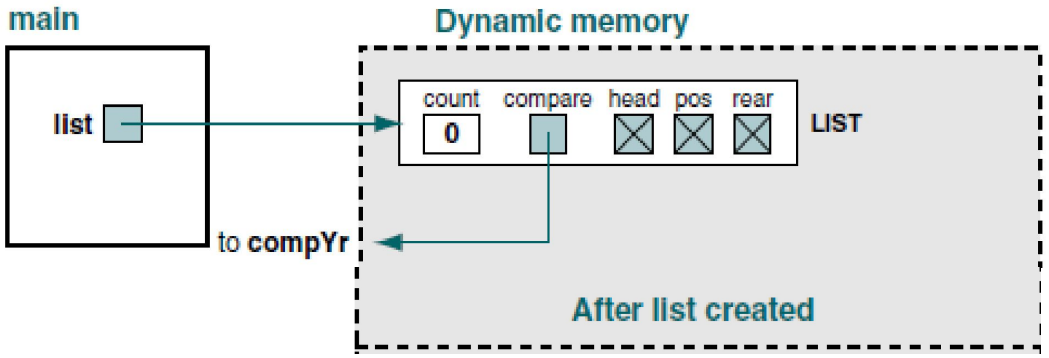
- %hd: formatação para ler um dado short integer;
- %-40s: alinhamento da string à esquerda, ocupando 40 caracteres, preenchendo com branco caso necessário;

- " %40[^\"]": formatação da entrada de dados usada no fscanf(), limita o tamanho da string a 40 caracteres, onde o [ ] define um scanset válido, neste caso o operador ^ exclui o caráter "; assim, lê 40 caracteres até a ocorrência do caracter ".
- " %*c": formatação da entrada de dados usada no fscanf(), o operador * ignora o último caracter lido;

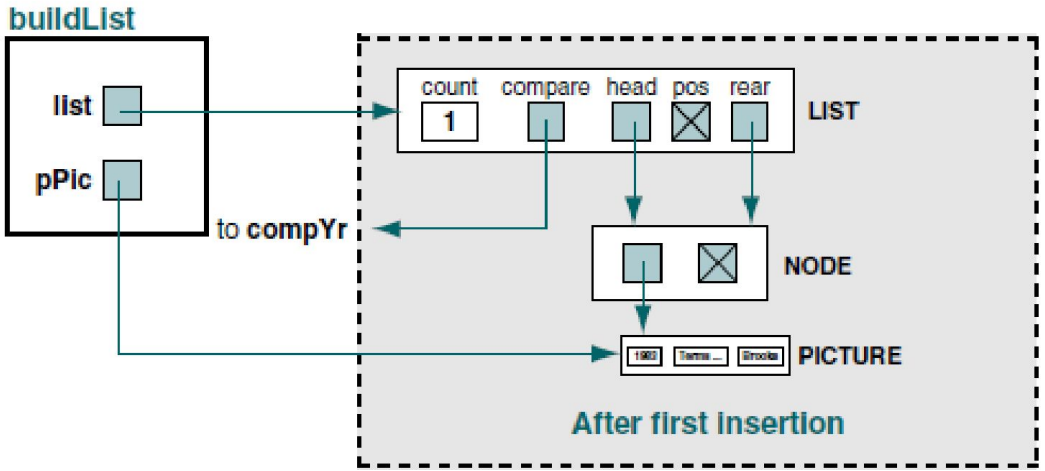
- fscanf()==1: verifica que o número de argumentos lidos seja =1.

Listas Aplicação
Ilustração da Lista

- Lista de filmes após a sua criação.

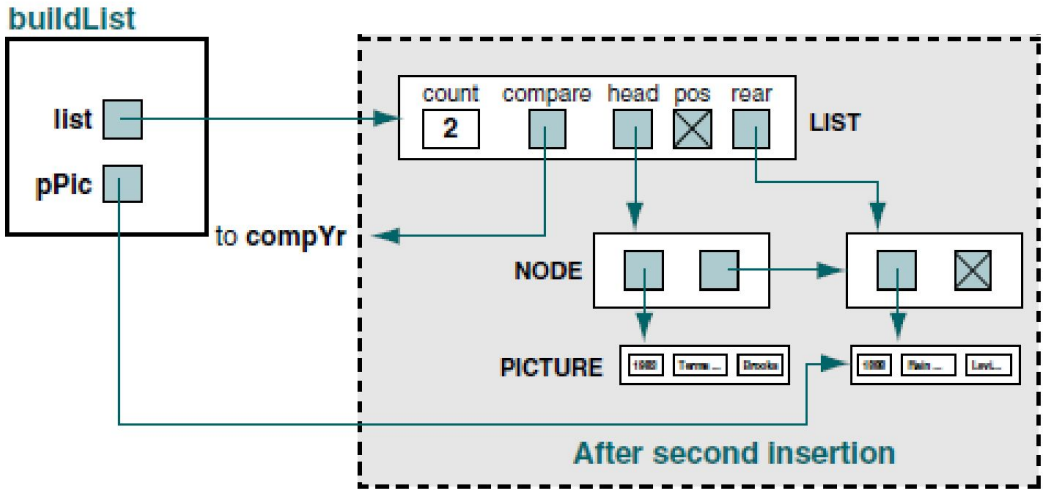


- Lista de filmes após a primeira inserção.



Listas Aplicação
Ilustração da Lista

- Lista de filmes após a segunda inserção.



Listas Aplicação
Função process()

- A função process() consiste de um loop de processamento que se repete indefinidamente até o usuário escolher a opção Q: Sair.
- Dentro do loop o usuário interage mediante a função getChoice() para escolher entre as opções: P, S, Q.
- Após a escolha, executa-se a função correspondente:
  - P: Imprimir dados de todos os anos, função printList();
  - S: Buscar por um ano específico, função search();

P5-20.h – Função process()

```
1  /*================== process ==================
2    Process user choices
3    Pre    list has been created
4    Post   all of user's choice executed
5  */
6  void process (LIST* list)
7  {
8  //Local Definitions
9    char choice;
10
11 //Statements
12   do
13      {
14       choice = getChoice ();
15
16       switch (choice)
17          {
18           case 'P': printList (list);
19                     break;
20           case 'S': search (list);
21           case 'Q': break;
22          } // switch
23      } while (choice != 'Q');
24   return;
25 }  // process
```

Listas Aplicação
Função getChoice()

- A função getChoice() funciona como um menu do opções para o usuário.
- O usuário pode escolher entre três opções válidas:
  - S: para buscar por um ano específico;
  - P: para imprimir dados de todos os anos;
  - Q: para sair.
- O menu se repete até que o usuário escolha uma o~pção válida.

P5-21.h – Função getChoice()

```
1  /*================== getChoice ==================
2    Prints the menu of choices.
3       Pre    nothing
4       Post   menu printed and choice returned
5  */
6  char getChoice (void)
7  {
8  //Local Definitions
9    char choice;
10   bool valid;
11
12 //Statements
13   printf("======== MENU ======= \n"
14          "Here are your choices:\n"
15          "  S: Search for a year\n"
16          "  P: Print all years  \n"
17          "  Q: Quit             \n\n"
18          "Enter your choice: ");
19   do
20      {
21       scanf(" %c", &choice);
22       choice = toupper(choice);
23       switch (choice)
24          {
25           case 'S':
26           case 'P':
27           case 'Q': valid = true;
28                     break;
29           default:  valid = false;
30                     printf("\aInvalid choice\n"
31                            "Please try again: ");
32                     break;
33          } // switch
34      } while (!valid);
35   return choice;
36 } // getChoice
```

Listas Aplicação
Função printList()

- A função printList () imprime o conteúdo da lista. Para cada nó da lista imprime-se: year(ano), nome do filme(picture) e diretor(director).
- Observe o uso da função traverse() do TAD Lista para:
  - Inicialmente, posicionar e recuperar no ponteiro pPic a primeira posição;
  - Recuperar no ponteiro pPic a próxima posição.

P5-22.h – Função printList()

```
1  /*================= printList =================
2    Prints the entire list
3       Pre    list has been created
4       Post   list printed
5  */
6  void printList (LIST* list)
7  {
8  //Local Definitions
9    PICTURE* pPic;
10
11 //Statements
12
13   // Get first node
14   if (listCount (list) == 0)
15      printf("Sorry, nothing in list\n\a");
16   else
17      {
18       printf("\nBest Pictures List\n");
19      traverse (list, 0, (void**)&pPic);
20       do
21         {
22          printf("%hd %-40s %s\n",
23                 pPic->year,      pPic->picture,
24                 pPic->director);
25         } while (traverse (list, 1, (void**)&pPic));
26      } // else
27   printf("End of Best Pictures List\n\n");
28 } // printList
```

Listas Aplicação
Notas sobre a Linguagem C

- traverse(list, 0, (void**)&pPic): o terceiro parâmetro da função traverse foi declarado como um ponteiro para um ponteiro void. Usar uma referencia dupla nos permite modificar o valor do ponteiro. Usar o tipo void nos permite que todos os tipos de ponteiros sejam passados.
- Mas porque precisamos fazer o casting (void**) para o ponteiro? Se não o fizermos recebemos um alerta do compilador.

- Para evitar essa notação sobrecarregada podemos definir uma macro:
  - #define traverse(l, n, p) traverse(l, n, (void**)&p)
  - De maneira a usar simplesmente:
  - traverse(list, 0, pPic)

Listas Aplicação
Notas sobre a Linguagem C

- Ponteiros void: um ponteiro void é um ponteiro de propósito geral que usado para armazenar referencias a qualquer tipo de dado.
- Um ponteiro void tem a mesma representação e alinhamento de memória que um ponteiro a caracter. Um ponteiro void nunca é igual a outro ponteiro.
- Qualquer ponteiro pode ser alocado a um ponteiro a void. Mas depois deve ser modificado a seu tipo original usando cast(). Somente assim, os valores serão iguais.

```
int num;
int *pi = &num;
printf("Value of pi: %p\n", pi);
void* pv = pi;
pi = (int*) pv;
printf("Value of pi: %p\n", pi);
```

Listas Aplicação
Função search()

- A função search() realiza a busca na lista de filmes procurando por um nó que corresponda a um ano específico e caso encontrado, imprime seus dados.
- O ano(year) é escolhido pelo o usuário.
- Observe o uso da função searchList() do TAD Lista para realizar a busca:
  - Passa-se um ponteiro a um nó com campo ano(year) desejado para comparação;
  - Caso seja encontrado, recuperar no ponteiro pPic o nó desejado.

P5-23.h – Função search()

```
1  /*==================== search ====================
2    Searches for year and prints year, picture, and
3    director.
4       Pre    list has been created
5              user has selected search option
6       Post   year printed or error message
7  */
8  void search (LIST* list)
9  {
10 //Local Definitions
11   short    year;
12   bool     found;
13
14   PICTURE  pSrchArgu;
15   PICTURE* pPic;
16
17 //Statements
18   printf("Enter a four digit year: ");
19   scanf ("%hd", &year);
20   pSrchArgu.year = year;
21
22   found = searchList (list, &pSrchArgu,
23                       (void**)&pPic);
24
25   if (found)
26       printf("%hd %-40s %s\n",
27              pPic->year, pPic->picture, pPic->director);
28   else
29       printf("Sorry, but %d is not available.\n", year);
30   return;
31 } // search
```

Listas Aplicação
Função cmpYear()

- A função cmpYear() (compare year ou comparar ano) compara os campos ano (year) de dois nós do tipo PICTURE.
- A função retorna:
  - -1: Se o ano (year) do primeiro nó for menor que o do segundo;
  - 0: Se ambos campos ano (year) forem iguais;
  - 1: Se o ano (year) do primeiro nó for menor que o do segundo.

P5-24.h – Função cmpYear()

```
1  /*==================== cmpYear ====================
2    Compares two years in PICTURE structures
3       Pre  year1 is a pointer to the first structure
4            year2 is a pointer to the second structure
5       Post two years compared and result returned
6    Return -1 if year1 less; 0 if equal; +1 greater
7  */
8  int cmpYear (void* pYear1, void* pYear2)
9  {
10 //Local Definitions
11   int   result;
12   short year1;
13   short year2;
14
15 //Statements
16   year1 = ((PICTURE*)pYear1)->year;
17   year2 = ((PICTURE*)pYear2)->year;
18
19   if (year1 < year2)
20       result = -1;
21   else if (year1 > year2)
22       result = +1;
23   else
24       result = 0;
25   return result;
26 } // cmpYear
```

Referências

- Gilberg, R.F. e Forouzan, B.A. Data Structures_A Pseudocode Approach with C. Capítulo 6. General Linear Lists. Segunda Edição. Editora Cengage, Thomson Learning, 2005.