Linguagem de Programação C++

Universidade Federal de Santa Catarina

Departamento de Engenharia Elétrica, CTC

Prof. Eduardo Augusto Bezerra

Class x Struct

Estruturas de dados

Uma estrutura de dados é um conjunto de diversos tipos de dados, os quais podem ter tamanhos diferentes, agrupados em uma única declaração:

```
struct nome_modelo {
  tipo1 elemento1;
  tipo2 elemento2;
  tipo3 elemento3;
  .
  .
} nome_objeto;
```

onde *nome_modelo* é o nome para o novo tipo criado (definido pela struct), e o parâmetro opcional *nome_objeto* é um identificador válido (ou identificadores) para instâncias do objeto. Por exemplo:

```
struct produtos {
  char nome [30];
  float preco;
} ;

produtos apple;
produtos orange, melon;
```

apple, **orange** e **melon** são todos objetos (variáveis) do tipo "produtos". Este novo tipo é utilizado de forma semelhante aos tipos primitivos (ex. int, float, ...). Alternativamente:

```
struct produtos {
  char nome [30];
  float preco;
} apple, orange, melon;
```

Para acessar os campos da struct de forma a realizar operações, é utilizado um ponto (.) inserido entre o nome do objeto e o nome do campo:

```
apple.nome
apple.preco
orange.nome
orange.preco
melon.nome
melon. preco
```

Exemplo com filmes:

```
// uso de struct                                  Entre com o titulo: Alien
```

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;

struct filmes_t {
  char titulo [50];
  int ano;
} meu, seu;

void mostra_filme (filmes_t filme);

int main ()
{
  char buffer [50];

  strcpy (meu.titulo , "2001 A Space Odyssey");
  meu.ano = 1968;

  cout << "Entre com o titulo: ";
  cin.getline (seu.titulo ,50);
  cout << "Entre com o ano: ";
  cin.getline (buffer,50);
  seu.ano = atoi (buffer);

  cout << "Meu filme favorito eh:\n ";
  mostra_filme (meu);
  cout << "E o seu eh:\n ";
  mostra_filme (seu);
  return 0;
}

void mostra_filme (filmes_t filme)
{
  cout << filme.titulo;
  cout << " (" << filme.ano << ")\n";
}
```

```
Entre com o titulo: Alien
Entre com o ano: 1979

Meu filme favorito eh:
 2001 A Space Odyssey (1968)
E o seu eh:
 Alien (1979)
```

Exemplo de array (vetor) de structs:

```
// array de structs
#include <iostream>
#include <stdlib.h>
using namespace std;

#define N_MOVIES 5

struct movies_t {
  char title [50];
  int year;
} films [N_MOVIES];

void printmovie (movies_t movie);

int main ()
{
  char buffer [50];
  int n;
  for (n=0; n<N_MOVIES; n++)
  {
    cout << "Nome do filme: ";
    cin.getline (films[n].title,50);
    cout << "Ano: ";
    cin.getline (buffer,50);
```

```
Nome do filme: Pulp Fiction
Ano: 1994
Nome do filme: Blade Runner
Ano: 1982
Nome do filme: Matrix
Ano: 1999
Nome do filme: Tropa de Elite
Ano: 2007
Nome do filme: Taxi Driver
Ano: 1975

Filmes favoritos:
Pulp Fuction (1994)
Blade Runner (1982)
Matrix (1999)
Tropa de Elite (2007)
Taxi Driver (1975)
```

```
      films[n].year = atoi (buffer);
    }
    cout << "\nFilmes favoritos:\n";
    for (n=0; n<N_MOVIES; n++)
      printmovie (films[n]);
    return 0;
  }

  void printmovie (movies_t movie)
  {
    cout << movie.title;
    cout << " (" << movie.year << ")\n";
  }
```

Classes em C++

Uma classe, de forma semelhante ao que ocorre com structs, é um tipo definido pelo usuário.

A classe é a unidade de ocultação e encapsulamento de dados do C++.

Declaração de classe especifica a representação de objetos da classe e o conjunto de operações que se pode aplicar a tais objetos.

Instância de uma classe: QUANDO UMA VARIÁVEL É DECLARADA DO TIPO DA CLASSE, UMA INSTÂNCIA DA CLASSE É CRIADA. ESSA INSTÂNCIA É DENOMINADA UM OBJETO.

Palavras utilizadas para a definição de classes:

- class
- struct
- union

Em uma classe declarada com *struct* as funções membros e variáveis são públicos por default. Isso significa que todos podem ser acessados com o operador . .

Se a palavra *class* é utilizada para declarar a classe, então os membros são privados por default.

O conceito de classe em C++ pode ser visto como uma generalização da noção de estrutura em C (*struct* em C não suporta funções).

Uma declaração de classe introduz um novo tipo, da mesma forma como apresentado para struct. Exemplo:

```
struct x {
   int a;
}

struct y {
   int a;
}

x a1;
y a2;
int a3;

a1 = a2; // erro: tipo y atribuído a tipo x
a1 = a3; // erro: tipo int atribuído a tipo x
```

Projeto de uma classe:

- Ao pensar sobre a classe a ser criada, imaginar qual será o objetivo da classe. Quais as propriedades e capacidades o objeto instanciado a partir dessa classe deverá possuir? Como será a interação entre esse objeto e outros?
- Listar qualquer dado relevante que precisa ser encapsulado no objeto como membro.
- Listar tudo o que o objeto sabe, e encapsular esses dados no objeto como funções membro.
- Solidificar a interface da classe. Escrever a declaração da classe e explorar as relações entre os membros (funções e dados). Decidir quais dados devem ser expostos e quais devem ser ocultados.

Estudo de caso: classe SystemUser

```
/* File SystemUser.h

   Eduardo Augusto Bezerra <Eduardo.Bezerra @ eel.ufsc.br>

   Departamento de Engenharia Elétrica, CTC, UFSC

   Agosto de 2012.

   Descricao: Definicao dos "headers" (interface) para a classe SystemUser

*/

#include <string>
using namespace std;

class SystemUser {
```

```
      string userName;
      string password;
      char accessLevel;
   public:
      SystemUser();
      ~SystemUser();
      void setUserName(string newUserName);
      void setPassword(string newPassword);
      void setAccessLevel(char newAccessLevel);
      string getUserName();
      string getPassword();
      char getAccessLevel();
};

/* File SystemUser.cpp

   Eduardo Augusto Bezerra <Eduardo.Bezerra @ eel.ufsc.br>

   Departamento de Engenharia Elétrica, CTC, UFSC

   Agosto de 2012.

   Descricao: Definicao das funcoes membro para os "headers" declarados na classe SystemUser

*/

#include "SystemUser.h"
using namespace std;

SystemUser::SystemUser() {
   userName = "";
   password = "";
   accessLevel = 'X';
}
SystemUser::~SystemUser() {
   userName = "";
   password = "";
   accessLevel = 'X';
}
void SystemUser::setUserName(string newUserName) {
   userName = newUserName;
}
void SystemUser::setPassword(string newPassword) {
   password = newPassword;
}
void SystemUser::setAccessLevel(char newAccessLevel) {
   accessLevel = newAccessLevel;
}
string SystemUser::getUserName(){
   return userName;
}
string SystemUser::getPassword(){
   return password;
}
char SystemUser::getAccessLevel(){
   return accessLevel;
}

/* File test_SystemUser.cpp

   Eduardo Augusto Bezerra <Eduardo.Bezerra @ eel.ufsc.br>

   Departamento de Engenharia Elétrica, CTC, UFSC

   Agosto de 2012.

   Descricao: Programa de teste para a classe SystemUser.

*/
```

```
#include <iostream>
#include "SystemUser.cpp"

using namespace std;

int main() {
   SystemUser usuario1, usuario2;
   string aux_user;
   char aux_level;
   string aux_passwd;
   cout << "User name? ";
   cin >> aux_user;
   cout << endl << "Password? ";
   cin >> aux_passwd;
   cout << endl << "Access level (A to E)? ";
   cin >> aux_level;

   usuario1.setUserName(aux_user);
   usuario1.setPassword(aux_passwd);
   usuario1.setAccessLevel(aux_level);

   usuario2.setUserName("Eduardo Bezerra");
   usuario2.setPassword("1234567890");
   usuario2.setAccessLevel('D');

   // usuario1 = usuario2;

   cout << endl << "----------------------" << endl;
   cout << "User name: " << usuario1.getUserName() << endl;
   cout << "Password: " << usuario1.getPassword() << endl;
   cout << "Access Level: " << usuario1.getAccessLevel() << endl;
   cout << endl << "----------------------" << endl;
   cout << "User name: " << usuario2.getUserName() << endl;
   cout << "Password: " << usuario2.getPassword() << endl;
   cout << "Access Level: " << usuario2.getAccessLevel() << endl;
}
```