UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

TECNOLOGIAS DE PROGRAMAÇÃO DE SISTEMAS DE INFORMAÇÃO

Programação orientada a objectos

PROGRAMAÇÃO ORIENTADA A OBJECTOS| Prof. Doutora Frederica Gonçalves

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA

Fundo Social Europeu

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

Classe:

– É uma estrutura de **dados** e **código** (funções) que permite a criação e utilização de **objectos** com essa estrutura.

Objecto:

– É uma instância ou concretização da estrutura definida numa **classe**. É definido como **dados** ou **atributos** e **métodos**.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Tipos de objectos:**

Dados ou atributos:

– Designa-se para as **variáveis** existentes na classe de origem do objecto.

Métodos:

– Designa-se para as **funções** incluídas na classe de origem do objecto. (são conjuntos de **instruções** como as **funções** estudadas)

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Características principais da programação orientada para objecto:**

Encapsulamento:

Polimorfismo:

Herança:

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Comparação entre estrutura e classe (struct e class):**

Estrutura de dados:

```
struct nome_da_estrutura {
        <lista de campos ou membros da estrutura>;
} [lista de variáveis do tipo da estrutura];
```

Classe:

```
class nome_da_classe {
        <lista de campos ou membros da classe>;
} [lista de variáveis do tipo da classe];
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Comparação entre estrutura e classe (struct e class):**

Estrutura de dados:

```
struct pessoa {
        char nome[40];
        int idade;
} p1;
```

Classe:

```
class pessoa {
        char nome [40];
        int idade;
} p1;
```

– No caso das classes, uma **variável** declarada como sendo do tipo de determinada classe deixa de se chamar variável para se chamar **objecto**.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Exemplo de uma classe:**

```
#include <iostream>
using namespace std ;

main () {
class pessoa {
public:
      char nome[40] ;
      int idade ;
} p1 ;
strcpy (p1.nome, "Ana Cruz"); p1.idade = 16 ;
cout << "Nome: " << p1.nome << '\t';
cout << "Idade: " << p1.idade << '\n'; system("pause"); }
```

Exercício:
Realizar o mesmo exemplo, com as seguintes alterações:
- Crie a classe “pessoa”, fora do main;
- Peça ao utilizador que introduza os dados.

- **Resolução e output:**

```
…
class pessoa {
public:
        char nome[40] ;
        int idade ;
} p1 ;
main () {
cout << “Introduza o seu nome: ”; gets(p1.nome);
cout << “Introduza a sua idade: ”; cin >> p1.idade;
system (“cls”);
…
```

```
Nome: Ana Cruz   Idade: 16
Prima qualquer tecla para continuar
```

- **Utilização da palavra-chave "*public*:"*:***

– Ao contrário dos programas com estruturas (***struct***), as variáveis (**objectos**) das classes são definidas por defeito com **acesso restrito**;

– A palavra – chave ***public*** permite o acesso por parte de todo o programa dos objectos da **classe**;

–Todo o objecto que está para além do "***public*:**" tem livre acesso em todo o programa

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Exemplo de classes com funções:**

```
…
main () {
class pessoa {
public:
        char nome[40] ;
        int idade ;
        int ano (int ano_actual) { return ano_actual – idade;}
} p1 ;
strcpy (p1.nome, "Ana Cruz"); p1.idade = 16 ;
cout << "Nome: " << p1.nome << '\t';
cout << "Idade: " << p1.idade << '\n';
cout << "Ano de Nacimento:" << p1.ano(2008) << '\n';
…
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Explicação da função e do funcionamento da mesma:**

```
int ano (int ano_actual)
{ return ano_actual – idade;}
```

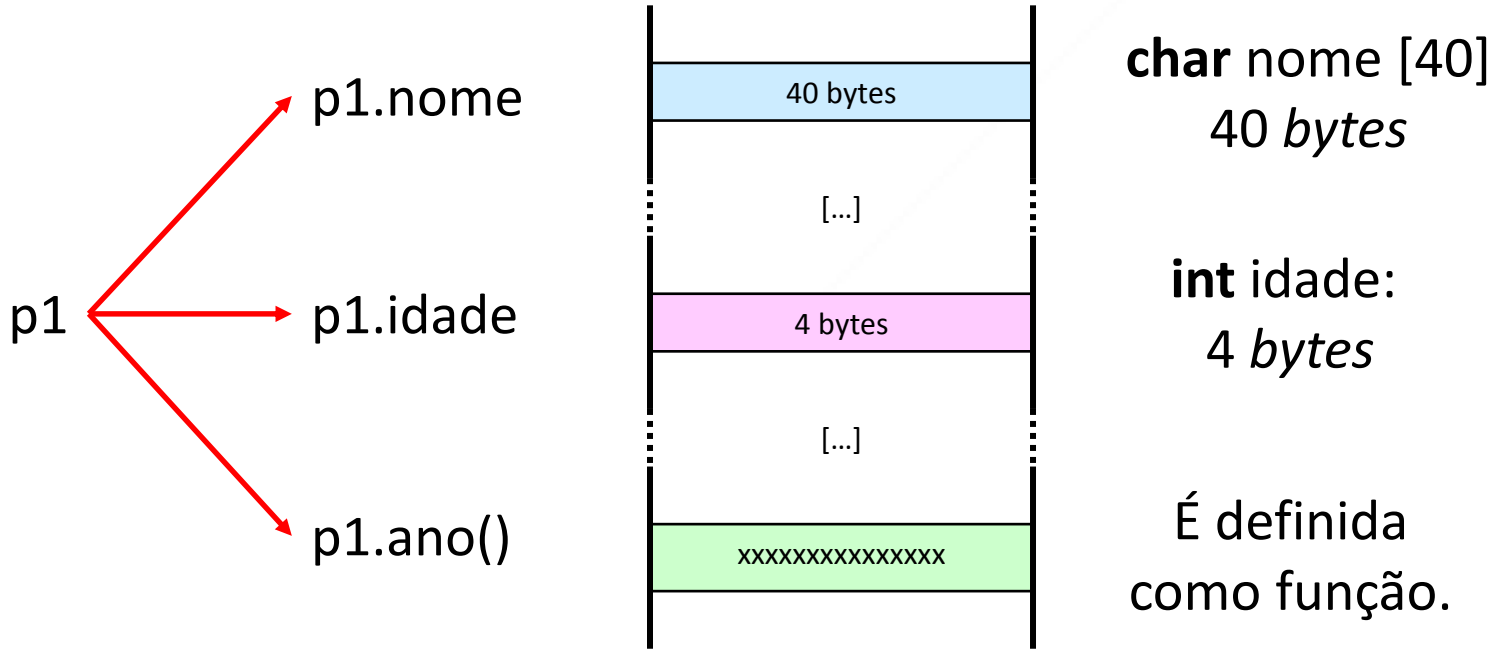
– A função **ano()** funciona com um parâmetro inteiro (**int ano_actual**).

– Devolve um inteiro: **return ano_actual – idade**;

– A chamada da função funciona como outro **objecto** da **classe**: **p1.ano(*ano_actual*)**.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Representação esquemática de uma classe na memória:**



UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Definir a utilização das estruturas e das classes:**

  – Todas as aplicações feitas nas **classes**, são válidas para as **estruturas**.

Estrutura de dados:	**Classe:**
– Quando se opera apenas com **dados (variáveis).**	– Quando se opera com **dados** e **funções** na mesma unidade de código.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Limitar o acesso aos objectos das classes:**

```
…
main () {
class pessoa {
        int idade;
public:
        char nome[40] ;
        int ano (int ano_actual) { return ano_actual – idade;}
} p1 ;
strcpy (p1.nome, "Ana Cruz"); p1.idade = 16;
cout << "Nome: " << p1.nome << '\t';
cout << "Idade: " << p1.idade << '\n';
cout << "Ano de Nacimento:" << p1.ano(2008) << '\n';
…
```

ERRO!!! PORQUÊ?

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Limitar o acesso aos objectos das classes:**

```
…
main () {
class pessoa {
        int idade;
public:
        char nome[40] ;
        int ano (int ano_actual) { return ano_actual – idade;}
} p1 ;
strcpy (p1.nome, "Ana Cruz"); // p1.idade = 16;
cout << "Nome: " << p1.nome << '\t';
// cout << "Idade: " << p1.idade << '\n';
cout << "Ano de Nacimento:" << p1.ano(2008) << '\n';
…
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Output:**

```
Nome: Ana Cruz
Ano de nascimento: -2009093308
Prima qualquer tecla para continuar . . .
```

- **Encapsulamento:**

– Define-se pela possibilidade de se **ocultar** ou **controlar o acesso** à composição interna das **classes**, aos seus **dados** e ao **código** das suas **funções – membro**.

– Podemos esconder ou proteger o código de uma classe através de palavras – chave.

- **Especificadores de acesso, *private* e *public*:**

private:	**public:**
– determina que o acesso aos **membros/objectos** da classe seja **privada**, ou seja, apenas dentro da **classe**.	– determina que o acesso aos **membros/objectos** da classe seja **pública**, ou seja, dentro ou fora da **classe**.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Estrutura de uma classe mais completa:**

```
class nome_da_classe {
        [ private: ]
        <lista de membros privados da classe>;
         [ public: ]
        <lista de membros públicos da classe>;
} [lista de variáveis do tipo da classe];
```

- **Terceiro especificador de acesso:**

protected:

– será especificado nas classes **derivadas** e de **herança**.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Solução do exemplo anterior:**

```
…
main () {
class pessoa {
  private:
  int idade ;
  public:
  char nome[40] ;
  void setidade (int i) {idade=i;}
  int getidade() {return idade;}
  int ano (int ano_actual)
  {return ano_actual - idade;}
} p1 ;
```

```
cout << "Indique o seu nome: ";
gets(p1.nome);
cout << "\nIndique a sua idade: ";
int j; cin >> j; p1.setidade(j);
system("pause");
system("cls");
cout << "Nome: " << p1.nome << '\t';
cout << "Idade: " << p1.getidade() <<
'\n';
cout << "Ano de nascimento: ";
cout << p1.ano(2008);
…
```

- **Funções de acesso:**

set():

– significa “**colocar**”;

– Permite a **introdução** de valores em campos/objectos **privados** das classes.

get():

– significa “**obter**”;

– permite **obter** os valores contidos nos campos/objectos **privados** das classes.

- **Funções de acesso:**

```
void setidade (int i) { idade = i; }
```

– ***void***, pois não retorna nenhum valor;

– utiliza o parâmetro (**int i**) para receber o argumento e atribuir o valor ao **objecto privado** “idade” – (idade = i).

- **Funções de acesso:**

```
int getidade () { return idade; }
```

– ***int***, pois retorna um valor inteiro;

– não utiliza parâmetros, porque apenas retorna o valor do **objecto privado** "idade" – (return idade).

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Acesso aos objectos privados:**

```
int j;
cin >> j;
p1.setidade(j);
```

```
cout << “Idade: ” << p1.getidade() << ‘\n’;
```

- **Funções – membro definidas fora da classe:**

– **funções** declaradas como **protótipos** na **classe**, e definidas fora da classe;

– são utilizadas para funções com maior número de instruções.

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Declaração de uma função – membro como protótipo:**

```
…
class empregado {
  private:
  char *codigo; float salario;
  public:
  char nome[40] ; int idade;
  void setcodigo (char *c) {codigo=c;}
  void setsalario (float s) {salario=s;}
  int *getcodigo() {return codigo;}
  int getsalario() {return salario;}
  void mostradados() //prototipo
} e1 ;
…
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Declaração de uma função – membro fora da classe, com o operador de resolução de escopo (::):**

```
...
void empregado::mostradados () {
cout << "Nome: " << nome << '\t';
cout << "Idade: " << idade << '\n';
cout << "Codigo: " << getcodigo() << '\t';
cout << "Salario: " << getsalario() << '\n';
system("pause");
}
```

- **Exercício:**

  – Com a classe definida “empregado”, e com a função – membro definida fora da classe “mostradados”, crie um programa/main(), que realize as operações de introdução de dados no “empregado e1” e chame a função mostredados().

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Exercício:** – Com a classe definida “empregado”, e com a função – membro definida fora da classe “mostradados”, crie um programa/main(), que realize as operações de introdução de dados no “empregado e1” e chame a função mostredados().

```
...
main () {
        strcpy(e1.nome, "Maria Matias");
        e1.idade = 28;
        e1.setcodigo("A-101");
        e1.setsalario(1000);
        e1.mostradados ();
}
...
```

- **Exercício:**

Elabore um programa em C++ que crie uma **classe** do tipo **aluno**, com os seguintes **objectos**: **char[ ]** nome_do_aluno, **int** número_do_aluno, **float** nota_esperada.
A introdução de informação deverá ser feita manualmente, através do **cout**, **cin/gets()**. Deve limpar o ecrã depois da introdução dos dados.

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS