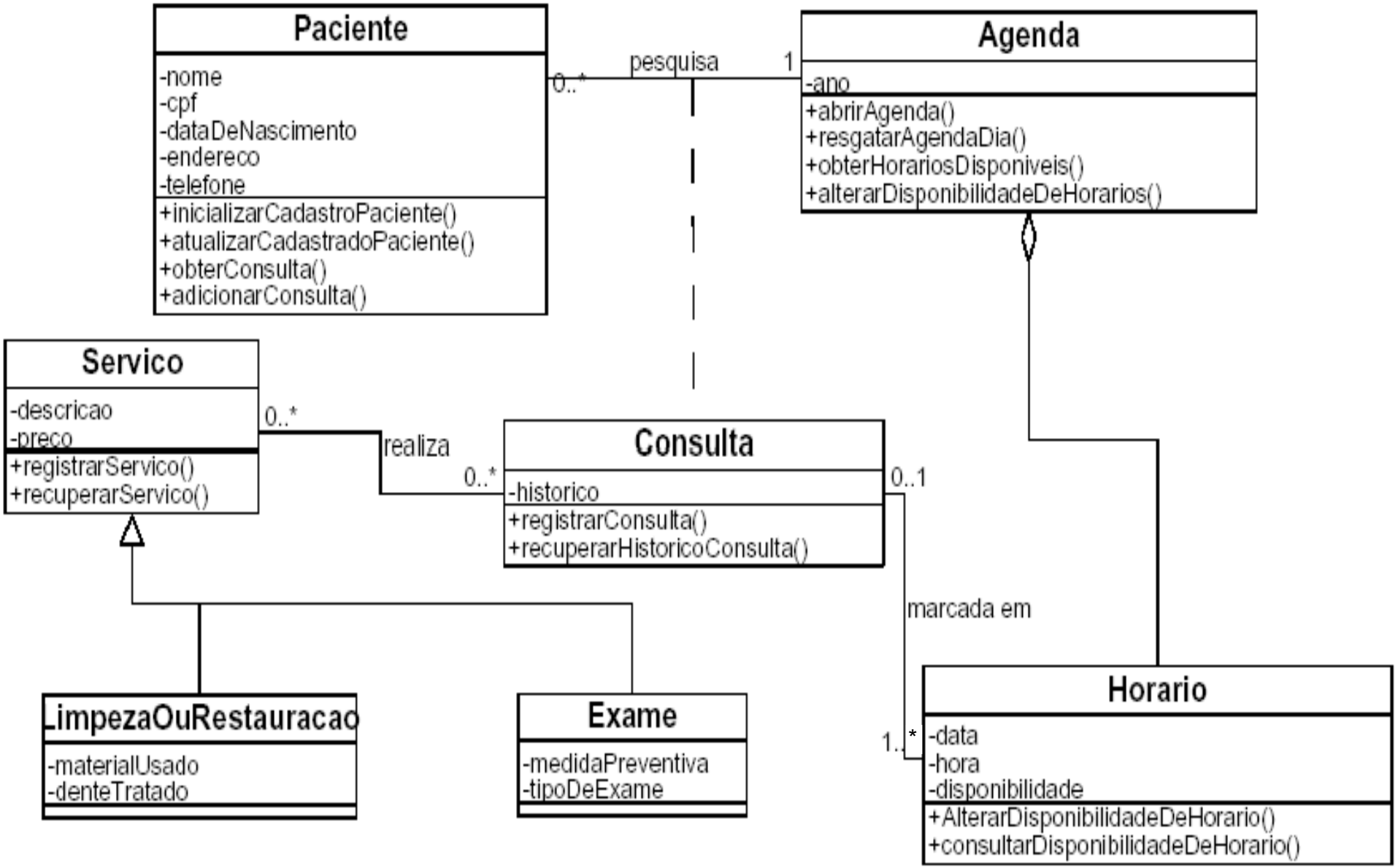

Paciente
-nome
-cpf
-dataDeNascimento
-endereco
-telefone
+inicializarCadastroPaciente()
+atualizarCadastradoPaciente()
+obterConsulta()
+adicionarConsulta()
pesquisa
0..*
1
Agenda
-ano
+abrirAgenda()
+resgatarAgendaDia()
+obterHorariosDisponiveis()
+alterarDisponibilidadeDeHorarios()
Servico
-descricao
-preco
+registrarServico()
+recuperarServico()
0..*
realiza
0..*
Consulta
-historico
+registrarConsulta()
+recuperarHistoricoConsulta()
0..1
marcada em
1..*
Horario
-data
-hora
-disponibilidade
+AlterarDisponibilidadeDeHorario()
+consultarDisponibilidadeDeHorario()
LimpezaOuRestauracao
-materialUsado
-denteTratado
Exame
-medidaPreventiva
-tipoDeExame

Introdução a UML

Exercício:

Com base nos conceitos estudados, modele, utilizando UML, um sistema OO para gerenciar as locações de uma empresa de locação de automóveis. O sistema deve se utilizar de todos os conceitos estudados.

Aplicação dos conceitos de programação orientada a objeto em linguagens de alto nível

Aplicação dos conceitos de programação orientada a objeto em linguagens de alto nível

Nesta etapa estudaremos linguagens de programação de alto nível que suportam a programação orientada a objeto (OO).

Dedicaremos nossa atenção a duas das mais populares linguagens de programação OO utilizadas atualmente, a linguagem C++ e a linguagem Java.

Iniciaremos nossos estudos pela linguagem C++, devido ao nosso conhecimento prévio da linguagem C, o que tornará mais amigável nosso contato com C++.

UNIVASF

Linguagem de Programação C++

Linguagem de Programação C++

Breve histórico:

- Desenvolvida por Bjarne Stroustrup;
- Nos laboratórios Bell;
- No início dos anos 80;
- Criada para suportar a escrita de algumas simulações complexas, para as quais as considerações de eficiência tornaram impossível o uso da Simula67.

UNIVASF

Linguagem de Programação C++

Características básicas:

- Extensão da linguagem C;
- Compatível com a linguagem C, preservando desta a integridade de numerosas bibliotecas e ferramentas;
- Acrescenta vários recursos às características de sua linguagem-mãe, sendo os mais importantes aqueles que suportam a abstração de dados e a programação orientada a objeto;
- Adotou do C os tipos básicos de dados, operações, sintaxe de instruções, e estrutura de programa;
- Linguagem híbrida.

UNIVASF

Linguagem de Programação C++

Foi mencionado que a linguagem C++ é compatível com a linguagem C e que este foi o fator que influenciou na opção de começarmos nosso estudo das linguagens OO pela linguagem C++.

Até que ponto são compatíveis?

A melhor forma de respondermos a esta pergunta é analisando alguns exemplos.

Nosso primeiro programa exemplo em C++ enviará a mensagem “Olah mundo!” para o monitor.

UNIVASF

Linguagem de Programação C++

```
#include <stdio.h>
int main()
{
   printf("Olah mundo!\n");
   return 0;
}
```

Este é o código fonte de um programa em C++?

Sim.

Linguagem de Programação C++

Para enfatizarmos a compatibilidade analisaremos um segundo programa exemplo.

Desta vez o programa receberá através da entrada padrão um valor inteiro pertencente ao intervalo [0,255] e a posição do bit que se deseja saber o valor, considerando a representação binária do valor recebido. O programa escreverá na saída padrão o valor do bit solicitado.

UNIVASF

Linguagem de Programação C++

```
#include <stdio.h>
main ()
{
   unsigned char desloca, valor_byte, aux=1;
   printf("\nDigite um numero pertencente ao intervalo ");
   printf("[0,255]: ");
   scanf("%d", &valor_byte);
   printf("\nDigite o bit a testar(0 a 7): ");
   scanf("%d", &desloca);
   aux = aux << desloca;
   valor_byte = valor_byte & aux;
   valor_byte = valor_byte >> desloca;
   printf("\nO valor do bit  eh %d",valor_byte);
   return 0;
}
```

Linguagem de Programação C++

Isso significa que não há diferença entre um código em C++ e em C?

Existe sim diferença entre os códigos, se o programador se utilizar das características particulares da linguagem C++.

Uma grande diferença entre as linguagens C e C++ é o fato de C++ possibilitar ao programador a criação de novos tipos de dados representados por classes, tipos estes que possibilitam ao programador trabalhar com todos os princípios da OO.

Antes de tratarmos dos aspectos OO da linguagem, vamos analisar outras diferenças entre as linguagens.

Linguagem de Programação C++

Existem diferenças sutis como:

- em C++ podemos usar // para definirmos um comentário de fim de linha;
- em C++ variáveis podem ser declaradas em "praticamente" qualquer ponto do programa.

Porém, vamos analisar agora a forma particular com que C++ trata a entrada e saída de dados utilizando periféricos padrões.

Esta forma é denominada entrada e saída por fluxo e trata-se de um recurso orientado a objeto mais elaborado. Porém, mais prático que o utilizado na linguagem C.

Linguagem de Programação C++

Para tal trabalharemos com os operadores de inserção de fluxo (<<) e extração de fluxo (>>). Tais operadores estão disponíveis na biblioteca iostream.h.

O objeto de fluxo de saída padrão é denominado *cout* (da classe ostream, para saída) e normalmente é "conectado" à tela (monitor); O objeto de fluxo de entrada padrão é denominado *cin* (da classe istream, para leitura) e normalmente é "conectado" ao teclado; O objeto de fluxo de erro padrão é denominado *cerr* (canal alternativo de saída para mensagens de erro).

Veremos agora um exemplo da utilização da entrada e saída por fluxo.

UNIVASF

Linguagem de Programação C++

```
#include <iostream.h>
int main()
{
   float preco;
   int n;
   std::cout << "Entre com o preco:\n";
   std::cin >> preco;
   std::cout << "Numero de itens:\n";
   std::cin >> n;
   std::cout << "\nTotal = ";
   std::cout << n*preco;
   return 0;
}
```

Linguagem de Programação C++

O std:: é colocado antes de cout e cin, pois é necessário quando utilizam-se nomes trazidos ao programa pela diretiva de pré-processador #include<iostream.h>. As notações std::cout e std::cin especificam que estamos utilizando nomes, nesse caso, cout e cin, que pertence ao 'namespace' std. O nome cerr também pertence ao namespace std. Os namespaces são um recurso avançado do C++.[3]

A declaração **using** permite omitir std:: antes de cada uso de um nome no namespace std. Sua utilização se dá da seguinte forma:

```
#include <iostream.h>
using std::cout;
using std::cin;
int main()
{
   float preco;
   int n;
   cout << "Entre com o preco:\n";
   cin >> preco;
   cout << "Numero de itens:\n";
   cin >> n;
   cout << "\nTotal = ";
   cout << n*preco;
   return 0;
}
```

Linguagem de Programação C++

Existe também a opção de utilizar a declaração **using** para incluir todos os nome do namespace std.

Sua utilização se dá da seguinte forma:

```
#include <iostream.h>
using namespace std;
int main() {
   float preco;
   int n;
   cout << "Entre com o preco:\n";
   cin >> preco;
   cout << "Numero de itens:\n";
   cin >> n;
   cout << "\nTotal = ";
   cout << n*preco;
   return 0;
}
```

Os operadores de inserção e extração de fluxo podem ser colocados em cascata, o que também denomina-se como inserção de fluxo de concatenação e encadeamento.

Por exemplo, no programa anterior podemos substituir as linhas:

```
cout << "\nTotal = ";
cout << n*preco;
```

Pela linha `cout << "\nTotal = "<< n*preco;`

Note que a instrução acima gera saída de múltiplos valores de diferentes tipos.

A sequência de escape \n pode ser substituída por std::endl. Neste caso, std:: também pode ser suprimido através da utilização de using.

endl é denominado manipulador de fluxo, gerando a saída de um caractere de nova linha e depois, esvaziando o buffer de saída.

Linguagem de Programação C++

Exercício:

Com base no que foi estudado informe qual será a saída no monitor gerada pelo seguinte programa:

```
#include <iostream.h>
using namespace std;
int main()
{
   cout << "Entre com o o multiplicando e o multiplicador: ";
   float multiplicando, multiplicador;
   cin >> multiplicando >> multiplicador;
   cout << endl << "Resultado: " << multiplicando * multiplicador;
   return 0;
} //defina a(s) entrada(s) necessária(s)
```


UNIVASF

Linguagem de Programação C++

Alunos atentos devem ter se perguntado:

Se a entrada e saída por fluxo é mais elaborada que os antigos printf() e scanf(), deve existir uma maneira de determinar o número de colunas que serão reservadas para a exibição de um determinado valor, de determinar a precisão de valores em ponto flutuante e a base dos valores?

Existe.

Estas e outras formatações são possíveis devido a outras funções e objetos predefinidos associados com **streams** (entrada e saída por fluxos) são os chamados **manipuladores de E/S**, da classe **ios** (biblioteca iomanip.h).

UNIVASF

Linguagem de Programação C++

Este são:

setw (int n) – tamanho do campo

setprecision (int n) – precisão do campo de ponto flutuante

setiosflags (long int f) – define ajuste de campo

resetiosflags (long int f) – cancela ajustes de campo

std::dec, std::hex, std::oct – definem a base dos valores

Linguagem de Programação C++

Alguns ajustes de campo disponíveis para setiosflags:

ios::left – campos ajustados à esquerda dentro da largura de campo setw()

ios::right – campos ajustados à direita dentro da largura de campo setw()

ios::scientific – formatação em notação científica

ios::showpoint – mostra os zeros à direita, quando necessário por questões de precisão

Linguagem de Programação C++

Para melhor fixarmos o que foi visto, vamos imaginar a situação hipotética em que deseja-se valores decimais impressos em hexadecimal, ajustados à esquerda, em campos de largura 10:

```
#include <iostream.h>
#include <iomanip.h>
using namespace std;
...
int num = 44;
cout << setw(10) << setiosflags(ios::left) << std::hex << num
<< num+1 << endl;
```

A saída em tela gerada será:

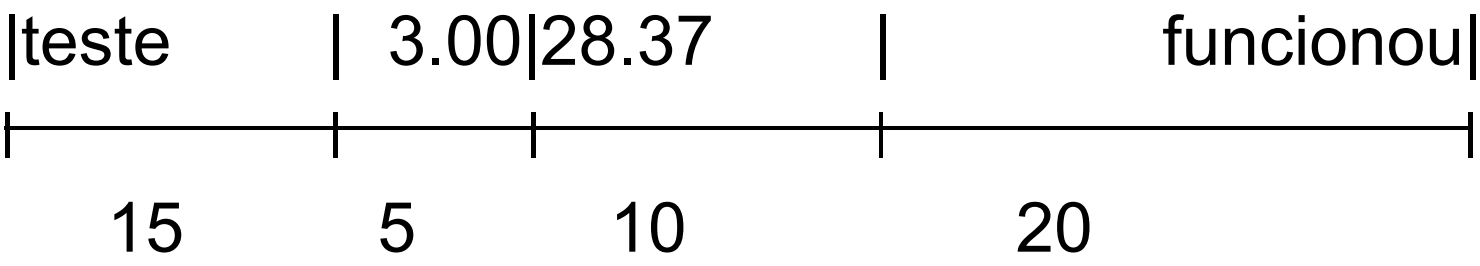
Exercício:

Com base no que foi visto, determine o que será exibido na saída padrão devido à execução do programa a seguir:

```
#include <iostream.h>
#include <iomanip.h>
using namespace std;
main()
{
   cout<<'|'<<setw(15)<<setiosflags(ios::left)<<"teste";
   cout<<'|'<<resetiosflags(ios::left)<<setw(5);
   cout<<setprecision(3)<<setiosflags(ios::showpoint)<<3.0;
   cout<<'|'<<resetiosflags(ios::showpoint)<<setw(10);
   cout<<setiosflags(ios::left)<<setprecision(4)<<28.37;
   cout<<'|'<<setw(20)<<resetiosflags(ios::left);
   cout<<"funcionou"<<'|';
   return (0);
}
```

Linguagem de Programação C++

Sequência apresentada no monitor:



Exercício:

Creio que algumas perguntas devem ter surgido nas mentes de vocês, como:

- ao definir a largura de um campo com o manipulador de fluxo setw() esta configuração perdurará até que eu efetue uma nova manipulação com setw()?

- ao setar a precisão de um número em ponto flutuante esta configuração perdura até que eu efetue uma nova manipulação do fluxo?

- ao definir uma nova base de numeração esta definição se manterá até que haja uma nova definição?

-...

Efetue testes e responda estas e outras perguntas.

Obs.: Em nosso estudo não foram esgotadas as possibilidades de manipulação de fluxo!

UNIVASF

Linguagem de Programação C++

Vamos nos recordar agora do que acontece se o programa a seguir receber como entrada a string "Software Orientado a Objeto":

```
#include <stdio.h>
main()
{
   char frase[30];
   scanf("%s", frase);
   printf("%s", frase);
   return (0);
}
```

Linguagem de Programação C++

O mesmo acontece com o programa a seguir escrito em C++:

```
#include <iostream.h>
using namespace std;
int main()
{
   char frase[30];
   cin >> frase;
   cout << frase;
   return 0;
}
```


UNIVASF

Linguagem de Programação C++

Para não pararmos a leitura da string ao ser encontrado um espaço utilizávamos, na linguagem C, por exemplo, o fgets().

No C++ utilizaremos os métodos do objeto cin:

get (char* cp, int tam[, char final]) – lê um vetor de até *tam-1* elementos, terminando com o caractere *final*, ou se este não for especificado, com <enter>.

getline – variante do get, que elimina o caractere de terminação.

Linguagem de Programação C++

Exemplo anterior com o método getline():

```
#include <iostream.h>
using namespace std;
int main()
{
   char frase[30];
   cin.getline (frase, 30);
   cout << frase;
   return 0;
}
```

Linguagem de Programação C++

Entretanto devemos ter em mente que a entrada e saída de dados por fluxo é fortemente tipada, ou seja, para se saber de que forma será feita a operação (E/S) é feita uma análise do tipo do operando.

Sendo assim, é possível imprimir o caractere correspondente a um inteiro?

Sim, estas e outras operações são possíveis.

Para tal analisaremos o programa a seguir que se utiliza das funções-membros (dos métodos) ***get*** e ***put***.

UNIVASF

Linguagem de Programação C++

```
#include <iostream.h>
using std::cin;
using std::cout;
using std::endl;
int main()
{
   int caractere;
   cout << "Forneca uma sequencia de simbolos qualquer: " << endl;
   while ((caractere = cin.get()) != '\n')
      cout.put(caractere);
   cout << endl << "Inteiro correspondente a '\n': " << caractere << endl;
   return 0;
}
```

//a função get sem argumentos extrai um caractere a partir do fluxo designado (incluindo caracteres não gráficos) e o retorna como o valor da chamada da função.