UDP server in C++ con le classi

// UDP.H // libreria

```
extern "C"
 {
#include <WinSock2.h>
}
```

diagramma della classe (UML)

UDP
– **socket_id** : SOCKET
+ **UDP** (port_number = 0 : ushort)   + **~UDP**( )   + **send** (ip_address: ulong, port_number: ushort , buffer[]: uchar, byte: int): int   + **receive** (ip_address: ulong*, port_number: ushort*, buffer: uchar*, size: int): int

```
class UDP
{
 private:
    SOCKET   socket_id;   // identificatore del socket

 public:

  // costruttore
  // parametro: numero di porta locale (default = 0   indica che il sistema operativo sceglierà un  numero
                  di porta  arbitrario)
  UDP(unsigned short   port_number = 0 )
  {
   WSADATA wsaData;
   struct sockaddr_in add; // struttura per indirizzo
   unsigned long arg = 1;

   // inizializzazione WinSock (versione 2.2)
   if (WSAStartup(0x0202, &wsaData) != 0)
      {
         socket_id = INVALID_SOCKET;
         return;
       }

   // apertura socket UDP
   if ((socket_id = socket( AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == INVALID_SOCKET)
      {
         WSACleanup();
         return;
      }

   // costruzione struttura indirizzo
   memset(&add, 0, sizeof(add)); // azzeramento struttura
   add.sin_family = AF_INET;                  // dominio indirizzi IP
   add.sin_addr.s_addr = 0;                   // indirizzo IP locale
   add.sin_port = htons(port_number);     // numero di porta assegnato
   if (bind( socket_id, (struct sockaddr *)&add, sizeof(add))== SOCKET_ERROR)
      {
        closesocket(socket_id);
        WSACleanup();
        socket_id = INVALID_SOCKET;
        return;
      }
```

```
  // impostazione del socket come non bloccante
  if (ioctlsocket(socket_id, FIONBIO, &arg) == SOCKET_ERROR)
      {
           closesocket(socket_id);
           WSACleanup();
           socket_id = INVALID_SOCKET;
           return;
      }
  return;
 }
```

// distruttore

```
~UDP(void)
 {
   if (socket_id != INVALID_SOCKET)
       closesocket(socket_id); // chiusura socket
  // terminazione WinSock
  WSACleanup();
  return;
 }
```

// trasmissione di dati al computer remoto

```
// parametri:
// ip_address: indirizzo IP destinatario
// port_number: numero di porta UDP destinatario
// buffer: dati da trasmettere
// byte: numero di byte da trasmettere
// valore restituito: -1 in caso di errore, altrimenti 0
int send( unsigned long  ip_address, unsigned short  port_number,  unsigned char buffer[],  int byte)
{
  struct sockaddr_in add; // struttura per indirizzo di destinazione
  int n;

  if (socket_id == INVALID_SOCKET)
      return -1;

 // costruzione struttura indirizzo
 memset(&add, 0, sizeof(add)); // azzeramento struttura
 add.sin_family = AF_INET; // dominio indirizzi IP
 add.sin_port = htons(port_number); // numero di porta UDP
 add.sin_addr.s_addr = htonl(ip_address); // indirizzo IP
 // trasmissione datagram
 if ((n = sendto( socket_id, (char*)buffer, byte, 0, (struct sockaddr*)&add, sizeof(add))) < 0)
        return -1;
 return n;
}
```

```
// ricezione di dati dal computer remoto
// parametri:
// ip_address: restituisce l'indirizzo IP del mittente
// port_number: restituisce il numero di porta UDP del mittente
// buffer: dati ricevuti
// size: dimensione del buffer
// valore restituito: numero di byte ricevuti, -1 in caso di errore
int receive( unsigned long* ip_address, unsigned short* port_number, unsigned char* buffer, int size)
{
  struct sockaddr_in add; // struttura per indirizzo mittente
  int dim = sizeof(add);
  int n;

  if (socket_id == INVALID_SOCKET)
        return -1;

 // ricezione dati (non bloccante)
 if ((n = recvfrom( socket_id, (char*)buffer, size, 0, (struct sockaddr*)&add, &dim)) <= 0)
        return -1;

// estrazione indirizzo IP e numero di porta UDP
 *ip_address = (unsigned long)(ntohl(add.sin_addr.s_addr));
 *port_number = (unsigned short)(ntohs(add.sin_port));
 return n;
}

};  // fine classe
```

// Server.cpp

```
#include <stdio.h>
#include <string.h>

#include "UDP.H"        // Inclusione della libreria
#include "sequence.h"


int main(void)
{
 unsigned char buffer[1024]; // buffer per ricezione
 unsigned long ip_address;
 unsigned short port_number;
 unsigned int num; // variabile per numero da generare
 int n;

 // creazione oggetto di classe UDP (socket con porta numero 23365)
 UDP  socket (23365);

printf("Servizio attivo...\r\n");

 while (1)
 {
   if ((n = socket.receive( &ip_address, &port_number, buffer, sizeof(buffer))) > 0)
       {
            // ricezione di un datagram e verifica del messaggio
           buffer[n] = '\0'; // terminatore di stringa
           if (strcmp((char*)buffer, "REQ") == 0)
                   {
                     // richiesta di generazione di un nuovo numero
                     num = htonl(sequence());
                     socket.send (ip_address, port_number,(unsigned char*)&num, sizeof(unsigned int));
                     printf("…inviato numero %u.\r\n", ntohl(num));
                    }
       }
 }
 return 0;
}
```