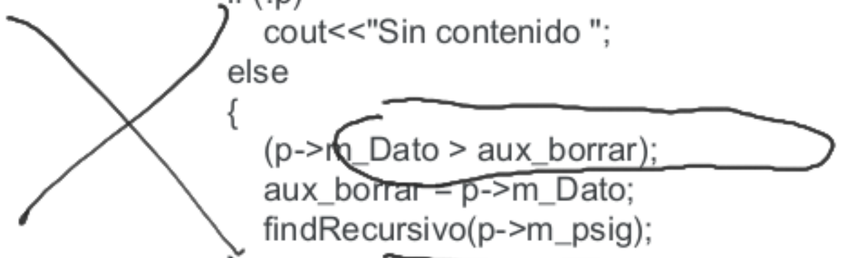
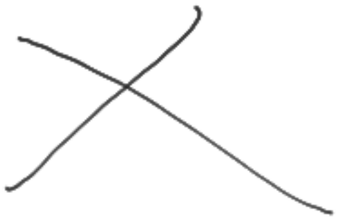
```
T max()
  {
     Nodo<T> *tmp = m_pHead;
     T max =tmp->m_Dato;
     while (tmp)
     {
        if (tmp->m_Dato >= max)
           max = tmp->m_Dato;
        tmp = tmp->m_pSig;
     }
     return max;
  }
```

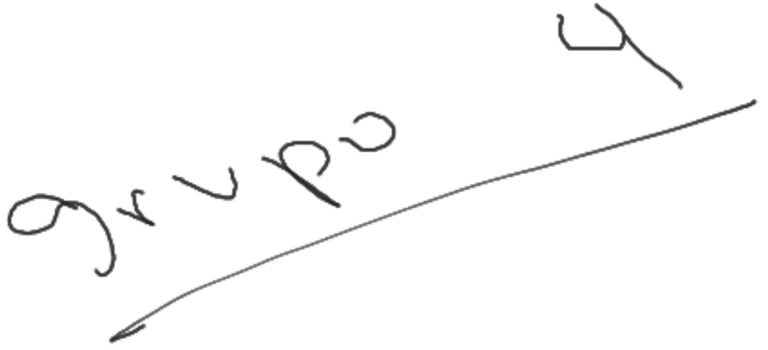
```
int maxRecursivo(Nodo <T> *p)
      {
         int aux_borrar = 0;
         if (!p)
            cout<<"Sin contenido ";
         else
         {
            (p->m_Dato > aux_borrar);
            aux_borrar = p->m_Dato;
```

```
int maxRecursivo(Nodo <T> *p)
    {
        int aux_borrar = 0;
        if (!p)
            cout<<"Sin contenido ";
        else
        {
            (p->m_Dato > aux_borrar);
            aux_borrar = p->m_Dato;
            findRecursivo(p->m_psig);
        }
        return aux_borrar;

    }
```







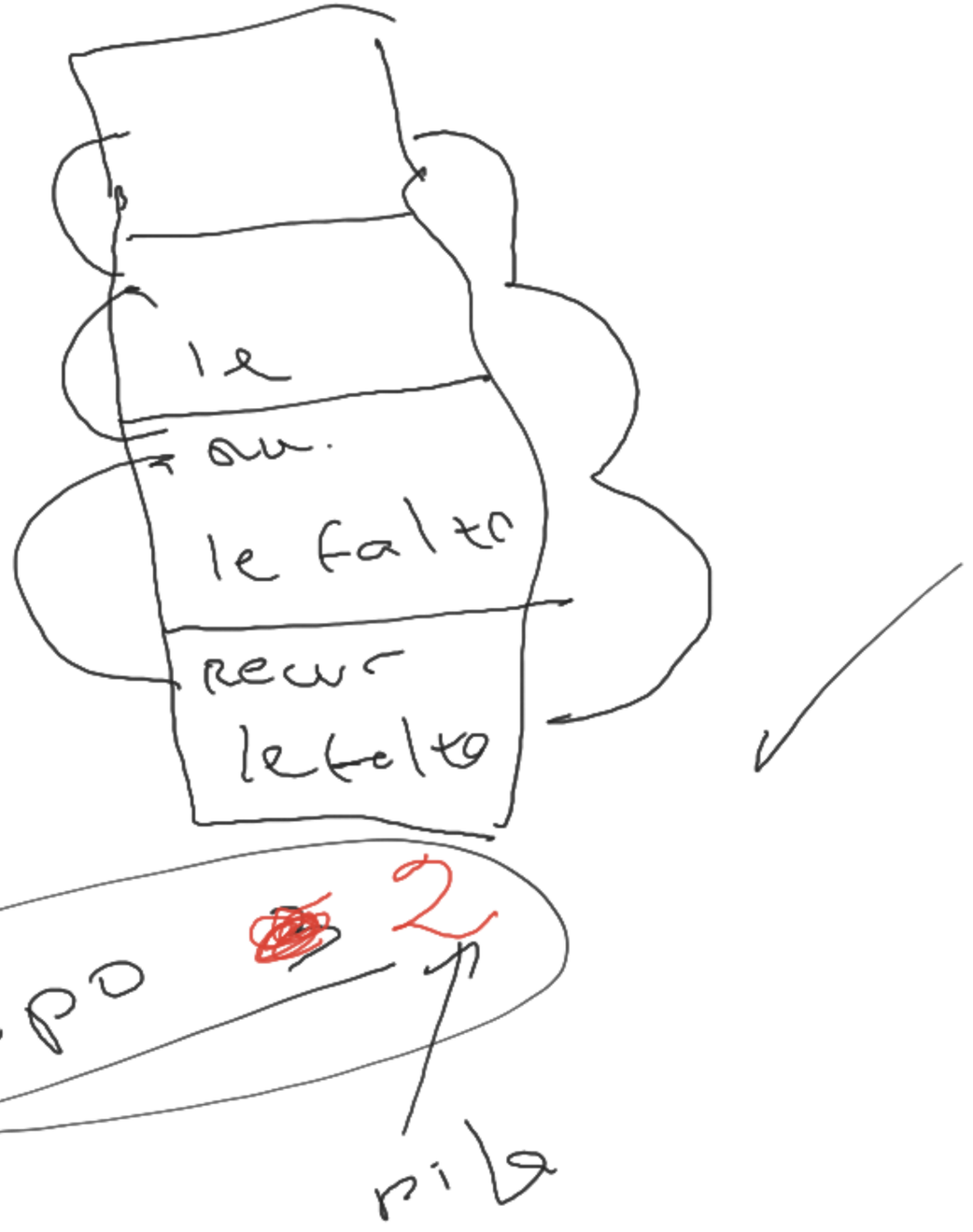
```
T max(Nodo<T> * m_pHead){

        if(m_pHead!=nullptr){
            if(m_pHead->m_Dato > maximo){
                maximo = m_pHead->m_Dato;

            }
            return max(m_pHead->m_pSig);
        }
        return maximo;

    }
```

```
void recursive_print_reverse(Nodo<T> * nodo)
    {
      if(nodo)
        {
            recursive_print_reverse(nodo->m_pSig);
             cout << nodo->m_Dato << "->";
        }
      else
        {
          cout << '\n';
        }
    }
     void recursive_print()
     {
        recursive_print_reverse(this->m_pHead);
     }
```

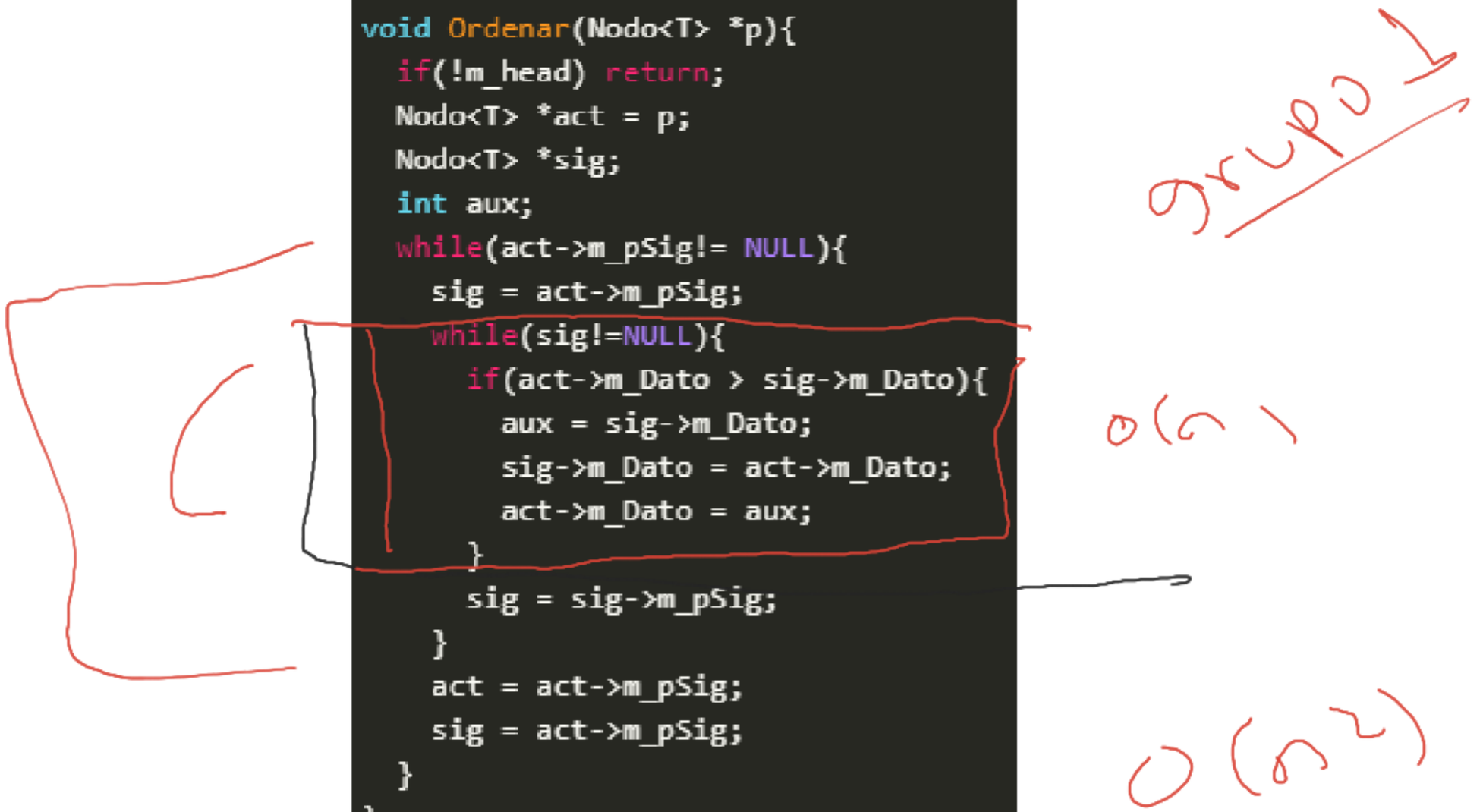


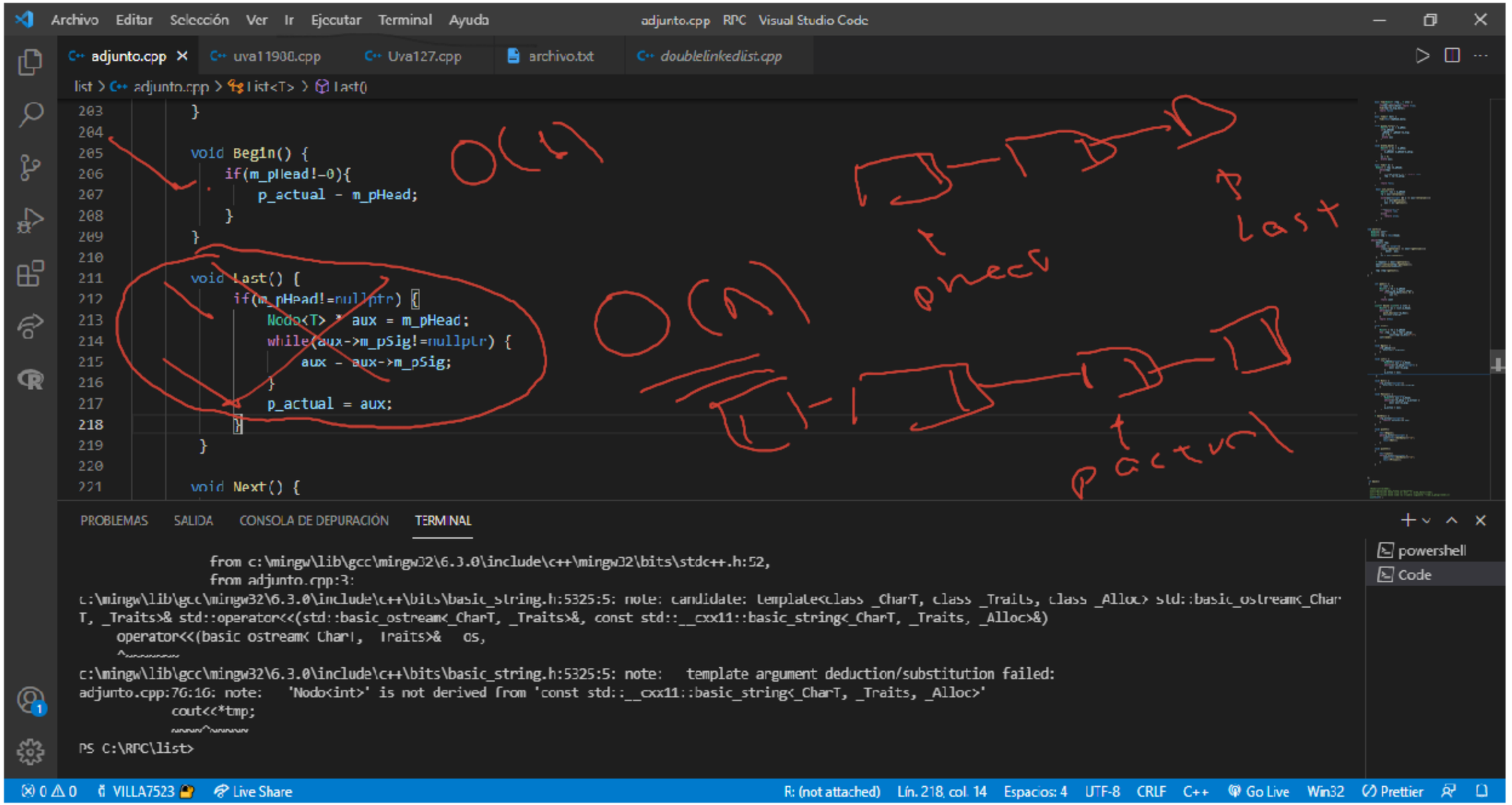
GRUPO 6

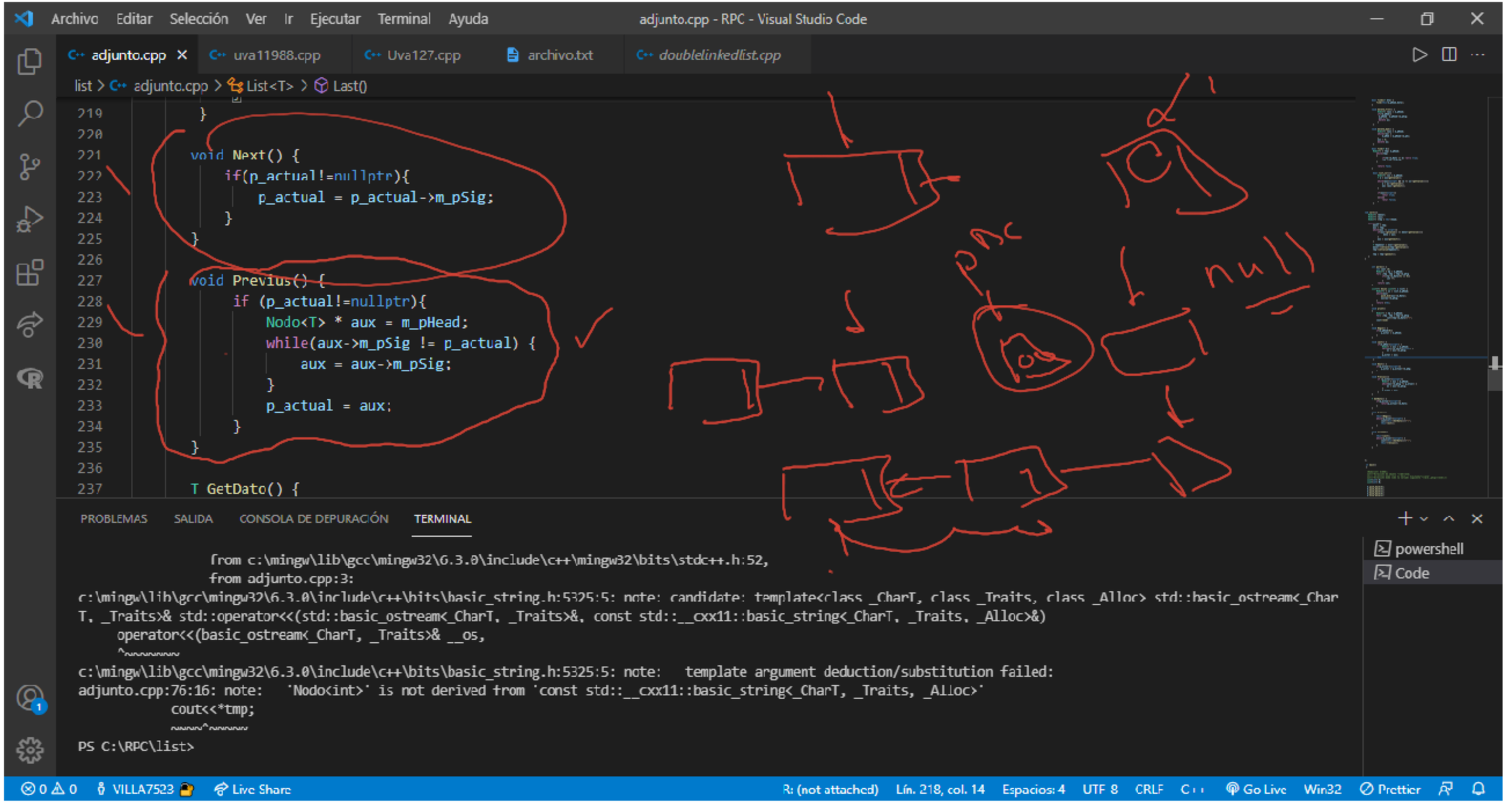
```
void printReverse(Nodo<T> * m_p){
        if(m_p==0) return;
        printReverse(m_p->m_pSig);
        cout<<m_p->m_Dato<<"<-";
    }
    void printReverseRecursivo(){
        cout<<"recursivo";
        printReverse(m_pHead);
    }
```

```
int count_evens() {
    int evens = 0;
    if (m_pHead) {
        Nodo<T>* tmp = m_pHead;
        for(; tmp; tmp = tmp->m_pSig) {
            if (tmp->m_dato%2 == 0)
                evens++;
        }
    return evens
}
```

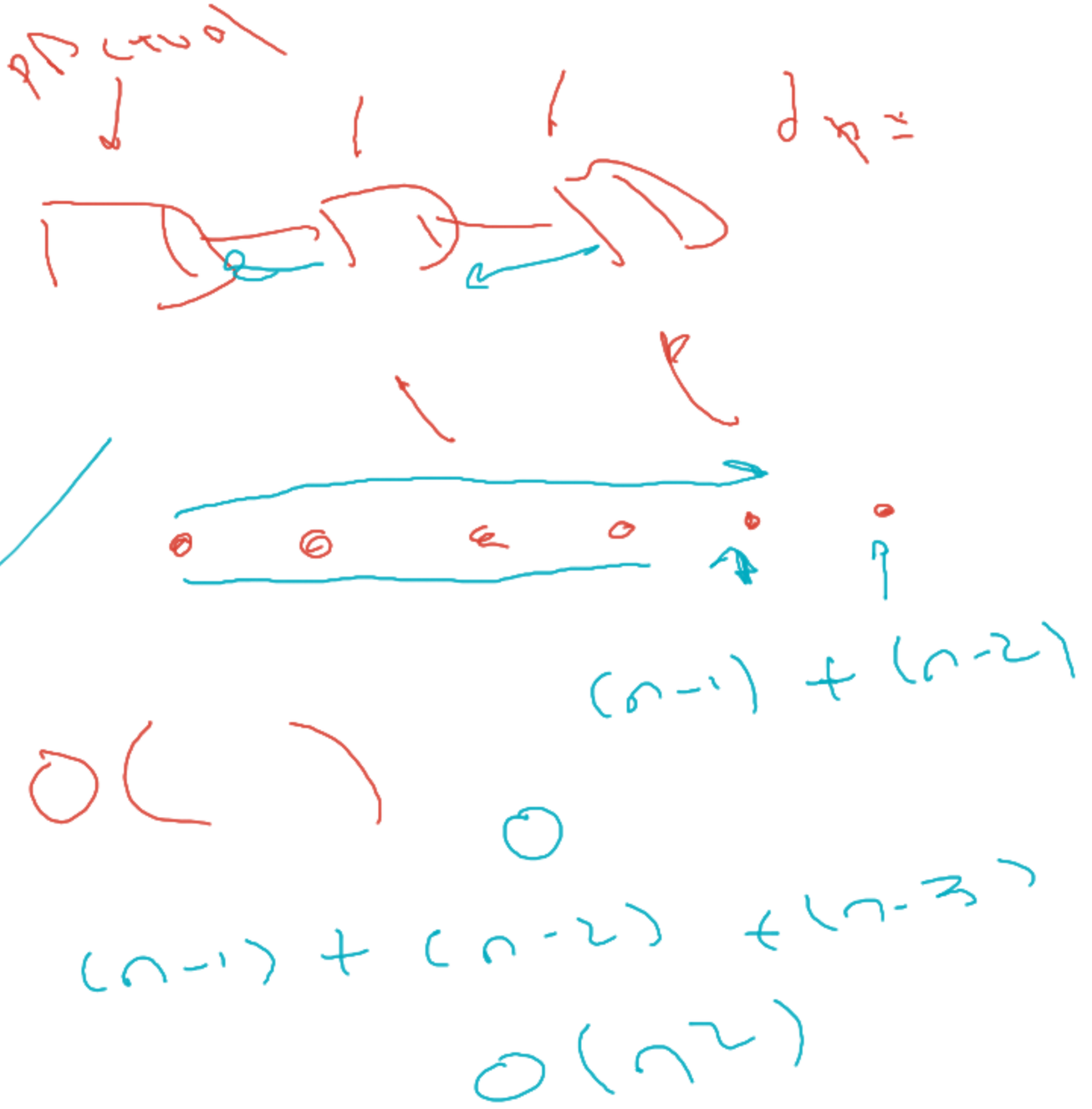
```
void Ordenar(Nodo<T> *p){
  if(!m_head) return;
  Nodo<T> *act = p;
  Nodo<T> *sig;
  int aux;
  while(act->m_pSig!= NULL){
    sig = act->m_pSig;
    while(sig!=NULL){
      if(act->m_Dato > sig->m_Dato){
        aux = sig->m_Dato;
        sig->m_Dato = act->m_Dato;
        act->m_Dato = aux;
      }
      sig = sig->m_pSig;
    }
    act = act->m_pSig;
    sig = act->m_pSig;
  }
}
```




Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
adjunto.cpp RPC Visual Studio Code
adjunto.cpp
uva11900.cpp
Uva127.cpp
archivo.txt
doublelinkedlist.cpp
list > adjunto.cpp > List<T> > Last()
203 }
204
205 void Begin() {
206 if(m_pHead!=0){
207 p_actual = m_pHead;
208 }
209 }
210
211 void Last() {
212 if(m_pHead!=nullptr) {
213 Nodo<T> * aux = m_pHead;
214 while(aux->m_pSig!=nullptr) {
215 aux = aux->m_pSig;
216 }
217 p_actual = aux;
218 }
219 }
220
221 void Next() {
O(1)
O(n)
p_actual
p_Last
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL
from c:\mingw\lib\gcc\mingw32\6.3.0\include\c++\mingw32\bits\stdc++.h:52,
from adjunto.cpp:3:
c:\mingw\lib\gcc\mingw32\6.3.0\include\c++\bits\basic_string.h:5325:5: note: candidate: template<class _CharT, class _Traits, class _Alloc> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, const std::__cxx11::basic_string<_CharT, _Traits, _Alloc>&)
operator<<(basic_ostream<_CharT, _Traits>& __os,
^~~~~~~~
c:\mingw\lib\gcc\mingw32\6.3.0\include\c++\bits\basic_string.h:5325:5: note: template argument deduction/substitution failed:
adjunto.cpp:76:16: note: 'Nodo<int>' is not derived from 'const std::__cxx11::basic_string<_CharT, _Traits, _Alloc>'
cout<<*tmp;
^~~
PS C:\RPC\list>
powershell
Code
0 0 VILLA7523 Live Share
R: (not attached) Lín. 218, col. 14 Espacios: 4 UTF-8 CRLF C++ Go Live Win32 Prettier


Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
adjunto.cpp - RPC - Visual Studio Code
adjunto.cpp
uva11988.cpp
Uva127.cpp
archivo.txt
doublelinkedlist.cpp
list > adjunto.cpp > List<T> > Last()
219 }
220
221 void Next() {
222 if(p_actual!=nullptr){
223 p_actual = p_actual->m_pSig;
224 }
225 }
226
227 void Previus() {
228 if (p_actual!=nullptr){
229 Nodo<T> * aux = m_pHead;
230 while(aux->m_pSig != p_actual) {
231 aux = aux->m_pSig;
232 }
233 p_actual = aux;
234 }
235 }
236
237 T GetDato() {
PROBLEMAS
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
from c:\mingw\lib\gcc\mingw32\6.3.0\include\c++\mingw32\bits\stdc++.h:52,
from adjunto.cpp:3:
c:\mingw\lib\gcc\mingw32\6.3.0\include\c++\bits\basic_string.h:5325:5: note: candidate: template<class _CharT, class _Traits, class _Alloc> std::basic_ostream<_CharT, _Traits>& std::operator<<(std::basic_ostream<_CharT, _Traits>&, const std::__cxx11::basic_string<_CharT, _Traits, _Alloc>&)
operator<<(basic_ostream<_CharT, _Traits>& __os,
^~~~~~~~
c:\mingw\lib\gcc\mingw32\6.3.0\include\c++\bits\basic_string.h:5325:5: note: template argument deduction/substitution failed:
adjunto.cpp:76:16: note: 'Nodo<int>' is not derived from 'const std::__cxx11::basic_string<_CharT, _Traits, _Alloc>'
cout<<*tmp;
~~~~^~~~~
PS C:\RPC\list>
powershell
Code
0 0 VILLA7523 Live Share
R: (not attached) Lín. 218, col. 14 Espacios: 4 UTF-8 CRLF C++ Go Live Win32 Prettier
~~~~

```
void PrintIT()
    {
        if(!m_pHead) return;
        this->Begin();
        while(pActual)
        {
            cout<<pActual->m_Dato<<" ";
            this->Next();
        }
    }
    void PrintIT2()
    {
        if(!m_pHead) return;
        this->Last();
        while(pActual)
        {
            cout<<pActual->m_Dato<<" ";
            this->Previus();
        }
    }
```



Lunes:

Implementar una lista doble enlazada