Data Abstraction

Process of
- Creating our own data types
- User Defined Data Types (UDT)
- Abstract Data Types (ADT)
  (is a data type we've created)
- List ADT
nok + head)

pratic:}
}(char *)
}(char *,
}(char *,
}{ Student
}{ Student
}{ Student
}

Defining the List Class
All Arrays dynamically allocated

```c

struct student
{
    char *name;
    char *psw;
    char *addr;
    char *email;
};

struct node
{
    struct student *student;
    struct node *next;
};

struct node
{
    struct student *student;
    struct node *next;
};

```
private:

}]}

public:

class List:

List &AT

Application => pp

11 Test bed
ask

Head = temp
delete Head

delete [ ] Head = a Student

next = Head

while (next)
    temp = next
    next = temp->next
    list = list + next

if (list == Null)
    list = list + temp

}
scribble (head = q - student. node).

shift (s. node) + 1.

if (head)

head = a - student. node.

head = new node.

if (head)

int count; insert (count, student s).

s.


```
while (current != null) {
  node = current.next;
  current = node;
  current.next = current.next.next;
  next = current.next;
  current.next = null;
}
```
list :: ~ list()
{
    node * temp;
    while (head)
    {
        temp = head->next;
        delete head;
        head = temp;
    }
}

student :: ~ student()
{
    delete [] name;
    delete [] psnid;
    delete [] email;
}
if (strcmp (name, s.name) ≠ 0) return 1;

int student::compare (student &s) { return 1;

switch (name) {
    case name: return 1;
    default: return 0;
}

while (current ≠ node) {
    current = current->next;
}

while (current ≠ node)
    current = current->next;

int main