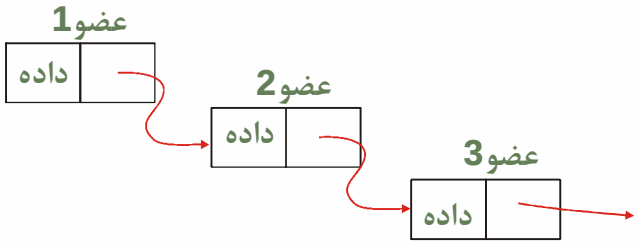
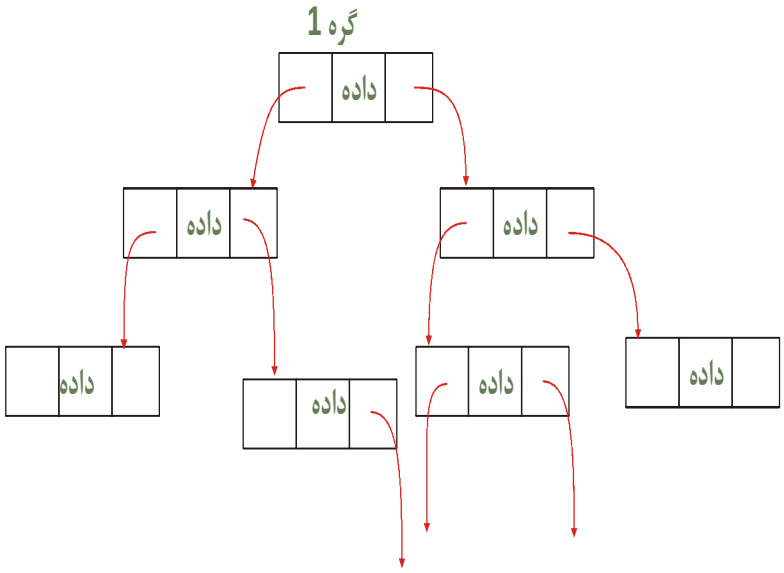
در پیاده سازی بسیاری از الگوریتم های آماده و موجود به انواع لیست های پیوندی بر می خوریم که خود نمونه‌ای از استفاده از انتزاع داده در طراحی راه حل مسایل است.

لیست پیوندی لیستی از داده هاست که بصورت ناپیوسته در حافظه ذخیره شده است هر عضویت حداقل آدرس ذخیره شدن یکی از اعضای بعدی یا قبلی لیست را در خود دارد.



در مواردی که کمبود حافظه یکپارچه وجود دارد (چیزی که امروزه در برنامه های معمولی مانند برنامه هایی که ما با آن ها سرو کار داریم اصلاً به چشم نمی خورد!) این نحو پیاده سازی لیست ها بجای آرایه ها الزامی است و البته این نوع داده بسیاری از ساختارها که به حالت عادی قابل پیاده سازی نیست پیاده سازی می کند. مثل یک درخت:



برای شروع به بررسی این پیاده سازی داشتن یکسری پیش زمینه الزامی است

مثل آنکه اشاره گر به یک نوع را می توان قبل تعریف نوع به عنوان یک نوع جدید ثبت کرد. برای فهم

موضوع مثال زیر را در نظر بگیرید.

```
type
  plinkedList =^ TlinkedList;
 TlinkedList = record
    Data : Extended;
    Next : PlinkedList;
end;
```

قبل تعریف نوع داده *TlinkedList* **اشاره گر به آن تعریف شده.**

این کار کمک می کند کـه درون نـوع داده *TlinkedList* **یـک عنـصر از نـوع اشـاره گـر بـه خـود** *TlinkedList* **داشته باشیم. فلسفه این کار اینست که چون اشاره گر فقط آدرسی از حافظه است و نـوع آن در واقع نوع برخورد با آن مکان حافظه است می توان درون یک ساختار آدرس جایی از حافظه را ذخیره کرد**

و در هنگام اجرا با آن آدرس به عنوان یک متغیر ازنوع ساختار مذکور برخورد کرد که تعریف لیست پیونـدی است.

مثالی از استفاده به شکل زیر است.

به ترتیب آزاد سازی حافظه هم دقت کنید.

```
var
  X : PlinkedList;
  M
  new (X );
  X .Data := 12.5;
  new (X .Next );
  X .Next ^.Data = 135.42;
  X .Next ^.Next := nil;
        M
  Dispose(X .Next );
  Dispose(X );
  M
  Dispose(X .Next );
  Dispose(X );
  M
```

برنامه و نمونه لیست پیوندی در ادامه آمده است.

درختهای یک نوع لیست پیوندی هستند. گره‌های هر درخت به تعدادی گره مشابه پیوند دارد. نمونـه ای از پیاده سازی بصورت زیر است. برای اجرا کردن آن یک *Console Application* ایجاد کنید و سـپس متن زیر را وارد کنید.

Program *Project*1*;*

```
{$APPTYPE CONSOLE}
type
  PNode=^TNode;
  TNode=record
    Value:Integer;
    Children:array of  PNode;
end;

var
  indent:Indent;

procedure PrintNode(Node:PNode);
begin
  if Node=nil then
    Exit;
  else
  begin
    for i:=1 to indent*2 do
      write(' ');
    writeln(Node.Value);
    for i:=1 to  indent*2 do
      write(' ');
    writeln('Children');
    inc(indent);
    for i:=0 to Length(Node.Children)-1 do
```

```
    PrintNode(Node.Children[i]);
  dec(indent);
  for i:=1 to indent*2 do
    write(' ');
  writeln('End of Children');
 end;
end;

var
 i:Integer;
 a:array[1..10] of TNode;
begin
 for i:=1to 10 do
  a[i].Value:=i;

 SetLength(a[1].Children,3);
 a[1].Children[0]:=@a[3];
 a[1].Children[1]:=@a[5];
a[1].Children[2]:=@a[7];

SetLength(a[2].Children,1);
a[2].Children[0]:=@a[4];

SetLength(a[5].Children,2);
a[2].Children[0]:=@a[6];
```

```
a[2].Children[1]:=@a[10];

indent=0;

for i:=1 to 10 do
begin
  writeln('Tree under ',i);
  PrintNode(@a[i]);
end;

readln;
end.
```

درخت دودویی جستجو یک نوع درخت است که هر گره آن حداکثر دو فرزند (گره مشابه در لایه زیرین) دارد. این ساختار چنان که از نامش پیداست برای جستجو مفید است. در این نوع درخت که تفاوتهایی با درختهایی که در ریاضیات گسسته خوانده اید دارد فرزندها با هم تفاوت دارند و یکی از فرزند سمت چپ و دیگری را فرزند سمت راست می نامیم.

درخت دودویی جستجو در حین دریافت اطلاعات ساخته می شود.روال کار بدین صورت است که اولین داده به عنوان گره ریشه (بدون هیچ والد) قرار می گیرد. پس از آن داده هایی که دریافت می شوند به این ترتیب در درخت قرار می گیرند که با شروع از گره ریشه اعمال زیر را انجام می دهیم:

1. **مقدار جدید را با مقدار گره مقایسه می کنیم.**

2. اگر بیشتر بود فرزند سمت چپ را انتخاب می کنیم و در غیر اینصورت فرزند سمت راست را.

3. اگر فرزند انتخاب شده در گره وجود نداشت آنرا ایجاد کرده و مقدارش را داده ورودی قرار می دهیم و اگر وجود داشت کل عملیات را برای گره فرزند انتخاب شده تکرار می کنیم.

با این ترتیب برای یافتن یک داده در درخت دیگر لازم نیست تمام درخت را بگردیم. کافیست از گره ریشه شروع کرده با مقایسه داده مورد نظر با مقدار اگر برابر نبودند شاخه سمت چپ یا راست ذیل گره را بگردیم.

با این ترفند حداکثر تعداد جستجوها برای یافتن یک داده خاص در میان n داده ذخیره شده برابر اولین عدد صحیح بزرگتر مساوی $\log_2^n$ خواهد بود.

$$Count\ \ of\ \ comparisons = \left|\log_2^n\right|$$

یک پیاده سازی برنامه زیر است که آنرا باید در حالت *Console* وارد کنید:

```
Program Project1;
{$APPTYPE CONSOLE}
type
  PBSTNode=^TBSTNode;
  TBSTNode=record
    Value:Integer;
    Left:PBSTNode;
    Right:PBSTNode;
end;

var
```

```
  Root:PBSTNode;


procedure Insert(Data:Insert;var Node:PBSTNode);
begin
  if Node=nill then
  begin
    New(Node);
    Node.Value:=Data;
    Node.Left:=nil;
    Node.Right:=nil;
    Exit;
  end;
  if  Node.Value>Data then
    Insert(Data,Node.Right);
  else
    Insert(Data,Node.Left);
end;


procedure Destroy(var Node:PBSTNode);
begin
  if  Node=nil then
    Exit;
  Destroy(Node.Left);
  Destroy(Node.Right);
  Destroy(Node);
```

```
  Node:=nil;
end;

procedure Find(Data:Integer;var Node:PBSTNode);
begin
  if Node=nil then
    Exit;
  If Node.Value=Data then
    writeln('Found!');
  else
    if  Node.Value>Data then
    begin
      writeln('Going to right…');
      Find(Data,Node.Right);
    end;
    else
      begin
        writeln('Going to left…');
        Find(Data,Node.Left);
      end;
end;

const
  RandomValues:array [1..80] of Integer=
  (0,200,50,25,40,66,77,90,112,500,442,1024,203,101,333,230,512,302,105,201,
```

```
  1,201,51,26,41,67,78,91, 113,501,443,1025,204,102,334,231,513,303,106,202,
  2,202,52,27,42,69,79,92,114,502,444,1026,206,103,335,233,515,306,109,203,
  9,212,12,37,52,89,75,82,214,402,244,3026,906,303,135,225,516,399,199,299);
var
  i:Integer;
begin
  for i:=1 to 80 do
    Insert(RandomValues[i],Root);
  Find(299,Root);
  Destroy(Root);
  readln;
end.
```