**3.1 – Strwythurau Data**

**Gofynion y Maes Lafur**

|  |  |
| --- | --- |
|  | Disgrifio, dehongli a thrin strwythurau data gan gynnwys araeau (hyd at dri dimensiwn), staciau, ciwiau, canghennau, rhestri cysylltiedig, a thablau stwnsh.  Disgrifio sut mae araeau (hyd at dri dimensiwn) yn cael eu trin.  Cynrychioli gweithrediad staciau a chiwiau gan ddefnyddio pwyntyddion ac araeau.  Cynrychioli gweithrediad rhestri cysylltiedig a changhennau gan ddefnyddio pwyntyddion ac araeau. |

**Araeau a Newidynnau**

Mae holl brosesu data ar gyfrifiadur yn golygu trin data. Gellir trefnu’r data hwn yng ngof y cyfrifiadur mewn gwahanol ffyrdd yn ôl sut y mae’n cael ei brosesu. Gelwir y wahanol ddulliau o drefnu data yn **strwythurau data**.

Mae gan ieithoedd cyfrifiadur parod **mathau o ddata elfennol** (megis cyfanrif, real, Boole a nod) a rhai parod **strwythuredig** neu fathau data **cyfansawdd** (*strwythurau data*) fel cofnod (*record*), arae (*array*) a llinyn (*string*). Mae'r mathau hyn o ddata cyfansawdd yn cynnwys nifer o elfennau o fath penodedig fel cyfanrif (*integer*) neu real.

### Newidynnau *(Variables)*

Newidyn yw ddarn o storio sylfaenol (cof cyfrifiadur / RAM) sy'n cael ei ddefnyddio i storio data dros dro. Dylid defnyddio enwau synhwyrol am newidynnau a datgan gyda mathau o ddata addas. Rydym yn diffinio newidynnau yn VBE2010 gyda’r gair Dim (sydd yn sefyll am Dimension) – sydd yn dangos bod newidyn wedi’u datgan.

### Araeuau *(Arrays)*

**Strwythur data** yw ffordd o drefnu data fel y gellir prosesu yn effeithlon gan ddefnyddio **algorithmau** syml.

Mae **arae** yn enghraifft o **strwythur data**.

Mae arau yn '**rhestr**' o eitemau data, sydd i gyd yn **un math o ddata**.

Os yr ydych yn ystyried newidyn fel blwch sengl yn y cof, yna gall arae cael ei ystyried fel nifer o flychau *o’r un math o ddata*.

Mae araeau yn cynnwys copïau o data cysylltiedig. Gallai datgan arae yn yr iaith Pascal fel a ganlyn:

Var

Exammark : array[0..9] of integer;

Neu yn VBE2010 fel:

Dim exammark(9) As Integer;

Mae hyn yn golygu bod yn lle un blwch yn y cof, bydde 10 blwch yn union yr un fath yn cael ei creu ac efallai edrych fel a ganlyn:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
| exammark |  |  |  |  |  |  |  |  |  |  |

Gellir adnabod pob blwch gan y *isysgrif* (subscript) mewn cromfachau sgwâr sy'n cyfeirio'n benodol at blwch unigol o fewn y arae.

Gallwch feddwl hefyd am yr arae fel rhestr o eitemau fel y dangosir isod:

exammark(0) =

exammark(1) =

exammark(2) =

: : :

exammark(9) =

Mae hyn yn enghraifft o **arae un-dimensiwn** a gellid ei ddefnyddio i storio copïau lluosog o'r *un data*, yn yr enghraifft hon, 10 copïau o'r marciau arholiad.

Rhesi strwythurau data pwerus iawn ac mae'n gymharol syml i ysgrifennu algorithmau a gweithdrefnau ar gyfer trin data mewn rhesi (e.e. chwilio am elfen, i roi trefn ar y casgliad, ac ati), gan eu gwneud yn strwythurau data cyffredin iawn mewn cyfrifiadura.

Prif anfantais arae yw bod rhaid i'r holl ddata fod o'r un math (er enghraifft, pob cyfanrif neu holl reals, ac ati.).

Gelwir yr eitemau data yma yn **elfennau** *(elements)*

Mae gan bob elfen **isysgrif** er mwyn ei adnabod (fel arfer nifer... ond nid o reidrwydd).

## Arae Un Dimensiwn

Esiampl o arae un dimensiwn yw **rhestr linol** **(linear list),** gallwch ddod o hyd i bob eitem yn unigol trwy ddefnyddio un is-sgript.

**Esiampl** : Dyma restr linol o enwau a elwir yn '**ENW**' .

|  |  |
| --- | --- |
| Is-Sgript | Enw |
| 1 | Geraint Davies |
| 2 | Sioned Evans |
| 3 | John Williams |
| 4 | Eleri Jones |
| 5 | Gerwyn Hughes |
| 6 | Owain Davies |

**Prosesu Arae Un Dimensiwn**

Er mwyn prosesu arae mae angen defnyddio **dolen** (loop). Dyma weithdrefn ym Pascal a fyddai'n dangos pob elfen o'r arae uchod.

|  |  |  |
| --- | --- | --- |
| **procedure DangosArae;** | | |
| **var i : integer;** | | |
| **Begin** | | |
|  | **for i := 1 to 6 do** | |
|  |  | **writeln(Enw[i]);** |
| **end;** | | |

Mae angen dwy is-sgript ar gyfer **arae 2 dimensiwn** **(two-dimensional array)**

Mae rhesi 2 ddimensiwn yn rhoi hyblygrwydd ychwanegol lle gellir cyfeirio at bob 'blwch' yn y casgliad fel cell mewn taenlen. Cofiwch bod pob eitem ddata a gedwir ym mhob blwch gyda rhesi, rhaid bod o'r un math data. Er enghraifft, gellid defnyddio arae 2 ddimensiwn i storio gwerthiant cynhyrchion ar gyfer nifer o wahanol bersonau gwerthiannau mewn cwmni (gweler yr enghraifft ar y dudalen nesaf).

**Esiampl:**

Mae'r arae 2 dimensiwn drosodd '**Gwerthiant**' yn storio cyfanswm gwerthiant misol 6 o werthwyr.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Ionawr** | **Chwefror** | **Mawrth** |
| ***John Davies*** | £450 | £320 | £520 |
| ***Linda Jones*** | £180 | £220 | £460 |
| ***Sion Griffiths*** | £620 | £650 | £710 |
| ***Lynn Hughes*** | £460 | £420 | £380 |
| ***Geraint Thomas*** | £120 | £210 | £190 |
| ***Llinos Davies*** | £420 | £380 | £550 |

Mae'r arae yn cynnwys 6 rhes a 3 colofn. (nid yw enwau'r misoedd na gwerthwyr wedi storio yn yr arae, mae'n yna i helpu ni yn unig i ddeall ystyr yr arae)

Felly mae **Gwerthiant [3,2]** yn storio '£460’ (Rhes 3, Colofn 2).

Er mwyn prosesu arae 2 dimensiwn rhaid defnyddio **dolen nythol** **(nested loop)** (dolen tu fewn i ddolen). Dyma weithdrefn ym Pascal a fyddai'n dangos pob elfen o'r arae uchod.

|  |  |  |  |
| --- | --- | --- | --- |
| **procedure DangosArray;** | | | |
| **var i, j : integer;** | | | |
| **Begin** | | | |
|  | **for i := 1 to 6 do** | | |
|  | **begin** |  | |
|  |  | **for j := 1 to 3 do** | |
|  |  |  | **write(Gwerthiant[j,i] : 15);** |
|  |  | **writeln;** | |
|  | **end** |  | |
| **end;** | | | |

**Rydych yn datgan arae 2D o linyn yn vb.net fel hyn:**



### Arae 3D (Three Dimensional Array)

Gellir ystyried araeau 3D fel casgliad o araeau 2D. Gellir diffinio maent yn yr un modd a 2D. Ystyried set o ffigurau gwerthiant, fel y dangosir yn y tablau isod.

Yr anfantais o ddefnyddio araeau 3D yw eu bod yn fwy cymhleth i'w rhaglennu.

2001

2000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Ionawr  [0] | Mis Chwefror  [1] | Mawrth  [2] | . . . |
| John  [0] | £450 | £320   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Ionawr  [0] | Mis Chwefror  [1] | Mawrth  [2] | . . . | | John  [0] | £450 | £320 | £520 | . . . | | Linda  [1] | £180 | £220 | £460 | . . . | | Sam  [2] | £620 | £650 | £710 | . . . | | Lynn  [3] | £460 | £420 | £380 | . . . | | Jim  [4] | £120 | £210 | £190 | . . . | | Lucy  [5] | £420 | £380 | £550 | . . . | | . . . | . . . | . . . | . . . | . . . | | £520 | . . . |
| Linda  [1] | £180 | £220 | £460 | . . . |
|  |  |  |  |  |
| Lynn  [3] | £460 | £420 | £380   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Ionawr  [0] | Mis Chwefror  [1] | Mawrth  [2] | . . . | | John  [0] | £450 | £320 | £520 | . . . | | Linda  [1] | £180 | £220 | £460 | . . . | | Sam  [2] | £620 | £650 | £710 | . . . | | Lynn  [3] | £460 | £420 | £380 | . . . | | Jim  [4] | £120 | £210 | £190 | . . . | | Lucy  [5] | £420 | £380 | £550 | . . . | | . . . | . . . | . . . | . . . | . . . | | . . . |
| Jim  [4] | £120 | £210 | £190 | . . . |
| Lucy  [5] | £420 | £380 | £550 | . . . |
| . . . | . . . | . . . | . . . | . . . |

Mae’r diagram yma o arae 3D yn dangos ffigurau

gwerthiant ar gyfer gwerthwyr gwahanol.

Dangoswyd 3 mis cyntaf y flwyddyn dros sawl

mlynedd. Gall storio’r data mewn arae 3D.

2002

**Gwerthiannau [0,0,1]** = £180

(lefel, x, y)

**Gwerthiannau [2,0,4]** = £120

**Rydych yn datgan arae 3D o linyn yn vb.net fel hyn:**



**Staciau**

Strwythur data olaf i mewn cyntaf allan (**LIFO** (Last In First Out)) yw'r STAC - meddyliwch am stac o blatiau wedi golchi, ar ôl golchi plât rhoddir ar ben y stac, pob tro y cymerwyd plât o'r stac fe gymerwyd o ben y stac.

**Gwthio** (pushing) - dull o ychwanegu eitem i'r stac.

**Popio** (popping) - dull o ddileu eitem o'r stac.

Gyda phob stac cei **pwyntydd stac** (stack pointer) sy'n dangos ble mae pen y stac. (Pe bawn yn gweithredu stac mewn rhaglennu defnyddir **ARAE** i dal elfennau'r stac gyda newidyn o fath gyfanrif '***TopOfStack***')

**Esiampl** : Dyma ddiagram o'r stac, yn cynnwys tri enw yn barod ar y stac.

Frig y pentwr = 3

(pwyntydd)

|  |
| --- |
|  |
|  |
|  |
| Mark |
| Daniel |
| Roxanne |

Ychwanegir enw newydd 'EGBERT' i'r stac, dyma'r stac newydd..

Frig y pentwr = 4

(pwyntydd)

|  |
| --- |
|  |
|  |
| Egbert |
| Mark |
| Daniel |
| Roxanne |

***(Nodwch fod y Pwyntydd Stac wedi cynyddu).***

Mae staciau yn strwythurau data defnyddiol ym myd cyfrifiaduron: e.e.

**Storio cyfeiriadau dychwelyd** (storing return addresses)**.**

 Wrth redeg rhaglen weithiau mae is-reolwaith yn cael ei galw. Rhoddir cyfeiriad

ble mae'r prosesu wedi stopio ar y stac (gwthio). Rhedir yr isreolwaith, ac wedyn mae prosesu yn dychwelyd i'r cyfeiriad sydd ar ben y stac (popio). Mae hyn yn golygu y gallwn alw is-rheolwaith o isreolwaith arall ayb….

 Defnyddir staciau i werthuso mynegiadau rhifyddol sydd wedi eu cynrychioli mewn **Nodiant Cil-Bwyl** (*Reverse Polish Notation*).

**Algorithmau ar gyfer gweithredu stac**

Dychmygwch arae **Key[1..stackpointer]** o gofnodion. **Dyma'r algorithm ar gyfer gwthio cofnod newydd a elwir NewRecord ar y stac.**

Procedure Push

If Top = MaxStackSize

Then Write ‘Stack is full’

Else

Add 1 to Top

Stack [Top] := NewItem

Endif

**Dyma'r algorithm ar gyfer popio cofnod o'r stac.**

Procedure Pop

If Top = 0

Then Write ‘Stack is empty’

Else

PoppedItem := Stack[Top]

Subtract 1 from Top

EndIf

EndProc

**Defnydd o staciau**

Defnyddir staciau mewn llawer o sefyllfaoedd gwahanol mewn cyfrifiadureg, er enghraifft:

1. I storio cyfeiriadau dychwelyd, paramedr a chynnwys y gofrestr wrth alw is-rheolweithiau (subroutines). Wrth i’r is-rheolwaith dod i ben, mae’r cyfeiriad sydd ar frig y stac yn cael ei phopio ac mae’r cyfrifiadur yn parhau i weithredu oddi wrth y cyfeiriad hwnnw.

2. Hefyd ei ddefnyddio ar gyfer pentyrru ymyriadau *(interrupt)* sydd wedi digwydd a poping hwy oddi ar un ar y tro i ddelio gyda nhw.

**Ciwiau (Queues)**

Dyma strwythur data Cyntaf i mewn - Cyntaf allan (**FIFO** (First In First Out)). Yr eitem gyntaf sy'n cael ei adio yw'r un cyntaf i gael gwared. Storiwyd elfennau data mewn lleoliadau cof cyfagos (contiguous) Meddyliwch am giw mewn archfarchnad. Mae angen 2 pwyntydd i ddefnyddio ciw:

 **Blaen (Front)** - sydd yn pwyntio at blaen y ciw (yr elfen sydd wedi bod yno hyraf)

 C**efn (Back)** - sydd yn pwyntio at gefn y ciw (yr elfen olaf i ymuno ar giw) Dyma ddiagram yn dangos ciw gyda 2 enw wedi adio :

|  |  |
| --- | --- |
| 8 |  |
| 7 |  |
| 6 |  |
| 5 |  |
| 4 |  |
| 3 |  |
| 2 | Gareth |
| 1 | Elen |

* Pwyntydd blaen = 1
* Pwyntydd cefn = 2

Pe bai **'Geraint'** a **'Siân'** yn ymuno ar ciw:

|  |  |
| --- | --- |
| 8 |  |
| 7 |  |
| 6 |  |
| 5 |  |
| 4 | Sian |
| 3 | Geraint |
| 2 | Gareth |
| 1 | Elen |

* Pwyntydd blaen = 1
* Pwyntydd cefn = 4

Wedyn fe ddileu elfen a ychwanegir **'Owain'**..

|  |  |
| --- | --- |
| 8 |  |
| 7 |  |
| 6 |  |
| 5 | Owain |
| 4 | Sian |
| 3 | Geraint |
| 2 | Gareth |
| 1 |  |

* Pwyntydd blaen = 2
* Pwyntydd cefn = 5

**Algorithmau**

Dyma’r algorithm sydd yn creu **Ciwiau Cylchol** (*circular queue*)

Q : array [1..6] of string;

Front, Rear, NumberInQueue : integer

To initialise the queue

Procedure Initialise

Front :=1

Rear := 6

NumberInQueue := 0

End Procedure

**Dyma’r algorithm i ychwanegu elfen i’r ciw:**

Procedure EnQueue

If NumberInQueue = 6

Then Write (‘Queue overflow’)

Else

If Rear = 6

Then Rear := 1

Else Add 1 to Rear

Else If

Q[Rear] : = NewItem

Add 1 to NumberInQueue

EndIf

End Procedure

**Dyma’r algorithm i gael gwared elfen o’r ciw**

Procedure DeQueue

If NumberInQueue = 0

Then Write (‘Queue empty’)

Else

NewItem := Q[Front]

Subtract 1 from NumberInQueue

If Front = 6

Then Front := 1

Else

EndIf

EndIf

End Procedure

**Problem gyda chiwiau:**

Wrth i eitemau data cael ei ddileu a'u hychwanegu fe fydd y lleoliadau mewn cof ar gyfer y ciw yn llenwi. Defnyddir y dull nesaf i osgoi hyn.

**Ciwiau Cylchol (Circular Queues)**

Edrychwch ar y dilyniant (sequence) canlynol:

|  |  |
| --- | --- |
| 8 |  |
| 7 | Geraint |
| 6 | Elen |
| 5 | Owain |
| 4 | Sian |
| 3 | Rhys |
| 2 | Dafydd |
| 1 |  |

* Pwyntydd blaen = 2
* Pwyntydd cefn = 7

Ychwanegir ‘**Sioned’** ac mae un enw yn cael ei ddileu

|  |  |
| --- | --- |
| 8 | Sioned |
| 7 | Geraint |
| 6 | Elen |
| 5 | Owain |
| 4 | Sian |
| 3 | Rhys |
| 2 |  |
| 1 |  |

* Pwyntydd blaen = 3
* Pwyntydd cefn = 8

**Mae’r arae nawr yn llawn i’r top..... OND**

Pe bai ‘**Llinos’** yn cael ei ychwanegu fe fydd y ciw yn edrych fel yr un isod...

|  |  |
| --- | --- |
| 8 | Sioned |
| 7 | Geraint |
| 6 | Elen |
| 5 | Owain |
| 4 | Sian |
| 3 | Rhys |
| 2 |  |
| 1 | Llinos |

* Pwyntydd blaen = 3
* Pwyntydd cefn = 1

Defnyddir strwythur cylchol (***circular structure***)

Pe bai enw arall yn cael ei ychwanegu fe fydd y ciw yn llawn, ac ni ellir ychwanegu eitem arall.

**Termau Pwysig....**

**Gorlifo (*Overflow*)** - Os ydych yn trio ychwanegu eitem i’r ciw ac mae pwyntydd cefn yn hafal (=) i'r pwyntydd blaen.

**Is-lifo (*Underflow*)** - Trio dileu o giw sydd yn wag.

**Defnyddio ciwiau**

Defnyddir ciwiau mewn amrywiaeth o gymwysiadau fel:

1. Trefnu swyddi sydd yn aros i gael ei redeg gan y cyfrifiadur

2. Bwffer i’r bysellfwrdd, sy’n caniatáu llinell gyfan i gael eu teipio a'u golygu tra bod y prosesydd yn brysur yn gwneud rhywbeth arall;

3. Wrth sbwlio ffeil ymlaen i ddisg wrth aros am argraffydd

**Coed Deuaidd . Canghennau (Binary Trees)**

Coeden ddeuaidd yn strwythur data a ddangosir isod. Nodi y gall pob cwgn ddisgynyddion ddwy ar y mwyaf.

Gwraidd

Cwgn

Rhiant

Dail

Descendant chwith

Ddisgynnydd iawn

**Adeiladu coeden ddeuaidd drefnus:**

Gall rhestr o eitemau cael ei osod mewn strwythur coeden ddeuaidd gan ddefnyddio’r rheolau canlynol:

 Rhowch yr eitem gyntaf ar y gwraidd

 Yn ei dro cymerwch bob eitem ddilynol

 Ddechrau gyda’r gwraidd. Os yw'r eitem yn LLAI na’r gwraidd, ychwanegwch i’r gangen ar y chwith. Os yw’’n FWY na’r gwraidd, ychwanegwch i’r gangen ar y dde.

 Defnyddiwch y rheol yma wrth i chi cyrraedd pob cwgn (node) - Llai = Chwith. Mwy = De

Er enghraifft rhowch yr enwau canlynol mewn coeden ddeuaidd: Gareth, Robert, Hope, Ashley, Neil a Stuart.

Gareth (1)

Robert (2)

Ashley (4)

Stuart (6)

Hope (3)

Neil (5)

(Mae'r rhifau mewn cromfachau i ddangos dilyniant y gosodwyd yr eitemau yn y goeden.)

**Gweithredu coeden ddeuaidd**

Er mwyn cynnal y strwythur hwn yng nghof y cyfrifiadur, rhaid defnyddio tair arae, un i ddal y rhestr o eitemau ac eraill i gynnal pwyntyddion y chwith a'r dde. Yn y goeden uchod, mae’r eitem gyntaf, **Gareth**, yn cyfeirio at eitem 4 (Ashley) ar y chwith ac eitem 2 (Robert) ar y dde.

Cynrychiolir y goeden gan y tair arae isod:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Enw** | **Pwyntydd chwith** | **Pwyntydd de** |
| 1 | Gareth | 4 | 2 |
| 2 | Robert | 3 | 6 |
| 3 | Hope | 0 | 5 |
| 4 | Ashley | 0 | 0 |
| 5 | Neil | 0 | 0 |
| 6 | Stuart | 0 | 0 |

**Trawsdeithio’r coeden ddeuaidd (Traversing a Binary Tree)**

Mae **‘Trawsdeithio coeden ddeuaidd’** yn golygu ymweld â phob cwgn y goeden yn ei dro ac yn perfformio rhyw broses ar y cynnwys e.e. dim ond ei darllen neu argraffu. Gallwch trawsdeithio coeden mewn unrhyw un o'r tair ffordd yma:

 Cyn-trefn (*Pre-order)*,

 Mewn trefn (*In-order)*

 Ol-trefn (*Post Order*).

Nodwch fod y ‘cyn’, ‘mewn’ ac ‘ôl’ yn cyfeirio at y pwynt lle rydych yn ymweld â’r gwraidd. Rydych yn ymweld â’r cwgn chwith bob amser cyn y cwgn sydd ar y dde.

Er enghraifft, ystyriwch y goeden ganlynol sy'n gynnwys dim ond tri chwgn:

D

E

A

**Trawsdeithio Cyn-trefn (*Pre-order)*,**

1. Ymweliad i’r gwraidd

2. Trawseithio’r is-goeden i’r chwith

3. Trawsdeithio’r is-goeden i’r dde

Ymwelir y cygnau yn y drefn **D A E**

Ymwelir â chygnau yn y goeden enw yn y drefn : Gareth, Ashley, Robert, Hope, Neil, Stuart.

**Trawsdeithio Mewn-Trefn (*In-Order)***

1. Trawseithio’r is-goeden i’r chwith

2. Ymweliad i’r gwraidd

3. Trawsdeithio’r is-goeden i’r dde

Ymwelir y cygnau yn y drefn **A D E**

Ymwelir â cygnau yn y goeden enw yn y drefn : Ashley, Gareth, Hope, Neil, Robert, Stuart.

**Trawsdeithio Ol-Drefn (*Post-Order)***

1. Trawseithio’r is-goeden i’r chwith

2. Trawsdeithio’r is-goeden i’r dde #

3. Ymweliad i’r gwraidd

Ymwelir y cygnau yn y drefn **A E D**

Ymwelir â chygnau yn y goeden enw yn y drefn : Ashley, Neil, Hope, Stuart, Robert, Gareth.

**Defnydd o goed deuaidd**

Mae coeden ddeuaidd yn strwythur data defnyddiol i storio eitemau sy'n cyrraedd mewn trefn ar hap, ond sydd angen cael ei chwilio yn gyflym i ddod o hyd i eitem benodol - er enghraifft, geiriadur mewn prosesydd geiriau. Gall yr eitemau cael ei llwytho’n hawdd mewn trefn yr wyddor neu rifiadol gan ddefnyddio **Trawsdeithio Mewn-Trefn (*In-Order).***

**Rhestrau Cysylltiedig (Linked Lists)**

 Cwgn (*node*) yw enw cofnod mewn rhestr.

 Mae gan bob cwgn maes sydd yn bwyntydd i’r cwgn nesaf yn y rhestr gysylltiedig. Y cwgn olaf yw'r nylbwyntydd (null pointer).

 Y **pwyntydd blaen** (*Start Pointer*) sydd yn cysylltu’r cwgn cyntaf i’r rhestr.

 Mae hefyd pwyntydd i’r **lleoliad rhydd nesaf** (*Next Free Pointer*) sydd yn rhoi cyfeiriad rhydd nesaf.

**Esiampl**:

Dyma restr gysylltiedig sydd wedi trefnu a ganddo bwyntyddion :

|  |  |  |
| --- | --- | --- |
| **Cyfeiriad** | **Tref** | **Pwyntydd** |
| 1 | Bridgend | 4 |
| 2 | Swansea | 5 |
| 3 | Reading | 2 |
| 4 | Cardiff | 3 |
| 5 | Swindon | 0 |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| Blaen = 1 | | |
| Pwyntydd rhydd nesaf = 6 | | |

Gallwn ddilyn y cysylltiadau a chyrchu'r cofnodion mewn trefn yr wyddor.

System well yw defnyddio 2 rhestr gysylltiedig - 1 i gysylltu'r llefydd rhydd i gyd -

|  |  |  |
| --- | --- | --- |
| **Cyfeiriad** | **Tref** | **Pwyntydd** |
| 1 | Bridgend | 4 |
| 2 | Swansea | 5 |
| 3 | Reading | 2 |
| 4 | Cardiff | 3 |
| 5 | Swindon | 0 |
| 6 |  | 7 |
| 7 |  | 8 |
| 8 |  | 0 |
| Blaen = 1 | | |
| Pwyntydd rhydd nesaf = 6 | | |

**Ychwanegu cwgn**

Beth am gysylltu cwgn newydd **‘Pembroke’** i restr uchod Dyma’r camau….

Ychwanegwch **Cwgn Newydd** i gyfeiriad **Pwyntydd rhydd nesaf** a diweddarwch (update) **Pwyntydd rhydd nesaf** i **Bwyntydd rhydd nesaf [Pwyntydd]**.

Ffeindiwch **Cwgn Blaenorol** trwy ddilyn y cysylltiadau o **Blaen** nes dod o hyd i’r cwgn olaf gyda Thref sy’n llai na **Cwgn Newydd**.

Newidiwch **Cwgn Newydd[Pwyntydd]** i **Cwgn Blaenorol[pwyntydd]** a newidiwch **Cwgn Blaenorol[pwyntydd]** i gyfeiriad y **Cwgn Newydd.**

Dilynwch y canlynol er mwyn esbonio'r uchod:

|  |  |  |
| --- | --- | --- |
| **Cyfeiriad** | **Tref** | **Pwyntydd** |
| 1 | Bridgend | 4 |
| 2 | Swansea | 5 |
| 3 | Reading | 2 |
| 4 | Cardiff | 3 |
| 5 | Swindon | 0 |
| 6 |  | 7 |
| 7 |  | 8 |
| 8 |  | 0 |
| Blaen = 1 | | |
| Pwyntydd rhydd nesaf = 6 | | |

**Cam 1.**

Ychwanegwch **Cwgn Newydd** i gyfeiriad **Pwyntydd rhydd nesaf** a diweddarwch (update) **Pwyntydd rhydd nesaf** i **Bwyntydd rhydd nesaf [Pwyntydd]**.

**Pwyntydd rhydd nesaf** = **6** a phwyntydd y cwgn hynny yw **7**, felly rhoddwn '**Pembroke**' yn gwgn **6**, a newid y **Pwyntydd rhydd nesaf i 7**.

(Mae rhestr gysylltiedig lle gwag yn gywir)

|  |  |  |
| --- | --- | --- |
| **Cyfeiriad** | **Tref** | **Pwyntydd** |
| 1 | Bridgend | 4 |
| 2 | Swansea | 5 |
| 3 | Reading | 2 |
| 4 | Cardiff | 3 |
| 5 | Swindon | 0 |
| 6 | Pembroke | 7 |
| 7 |  | 8 |
| 8 |  | 0 |
| Blaen = 1 | | |
| Pwyntydd rhydd nesaf = 7 | | |

**Cam 2.**

Ffeindiwch **Cwgn Blaenorol** trwy ddilyn y cysylltiadau o **Blaen** nes dod o hyd i’r cwgn olaf gyda Thref sy’n llai na **Cwgn Newydd**.

Cysylltiadau yw **Bridgend - Cardiff** - arhoswn oherwydd bod y cwgn nesaf **Reading**) y dod ar ôl '**Pembroke**'.

Felly mae **Cwgn** blaenorol = 4 (Cardiff)

|  |  |  |
| --- | --- | --- |
| **Cyfeiriad** | **Tref** | **Pwyntydd** |
| 1 | Bridgend | 4 |
| 2 | Swansea | 5 |
| 3 | Reading | 2 |
| 4 | Cardiff | 3 |
| 5 | Swindon | 0 |
| 6 | Pembroke | 7 |
| 7 |  | 8 |
| 8 |  | 0 |
| Blaen = 1 | | |
| Pwyntydd rhydd nesaf = 7 | | |

**Cam 3.**

Newidiwch **Cwgn Newydd[Pwyntydd]** i **Cwgn Blaenorol[pwyntydd]** Pwyntydd Pembroke yw **3** a newidiwch **Cwgn Blaenorol[pwyntydd]** i gyfeiriad y **Cwgn Newydd.**

Pwyntydd Cardiff nawr yw **6**.

|  |  |  |
| --- | --- | --- |
| **Cyfeiriad** | **Tref** | **Pwyntydd** |
| 1 | Bridgend | 4 |
| 2 | Swansea | 5 |
| 3 | Reading | 2 |
| 4 | Cardiff | 6 |
| 5 | Swindon | 0 |
| 6 | Pembroke | 3 |
| 7 |  | 8 |
| 8 |  | 0 |
| Blaen = 1 | | |
| Pwyntydd rhydd nesaf = 7 | | |

**Dileu Cwgn**

Rydym am ddileu **‘Swansea**'. Dyma’r camau (*ffug god syml)*:

1. Dilynwch y pwyntyddion nes dod o hyd i’r cwgn sydd yw dileu (**Swansea**), nodwch beth yw'r pwyntydd blaenorol (**Reading**).

2. Newidiwch bwyntydd y cwgn blaenorol (**Reading**) i bwyntydd y cwgn sydd yw dileu (**Swansea**) sef pwyntydd (**5**).

3. Newidiwch bwyntydd y cwgn sydd wedi dileu (**Swansea**) (**5**) i **Bwyntydd rhydd nesaf** (**7**) a’r **Pwyntydd rhydd nesaf[pwyntydd]** (**7**) i gyfeiriad y cwgn sydd wedi dileu (**2**).

Dyma restr gysylltiedig ar ôl i bopeth digwydd:

|  |  |  |
| --- | --- | --- |
| **Cyfeiriad** | **Tref** | **Pwyntydd** |
| 1 | Bridgend | 4 |
| 2 | Swansea | 5 |
| 3 | Reading | 2 |
| 4 | Cardiff | 3 |
| 5 | Swindon | 0 |
| 6 |  | 7 |
| 7 |  | 8 |
| 8 |  | 0 |
| Blaen = 1 | | |
| Pwyntydd rhydd nesaf = 2 | | |

Mae cwgn '**Swansea**' dal wedi storio yn ffisegol ond mae’n rhan o restr cygnau rhydd ac fe fydd yn cael ei ysgrifennu drosodd yn y dyfodol.

Dyma restr arall yn cynnwys mwy nag un pwyntydd (un yn nhrefn yr wyddor ar llall yn ôl y gorsafoedd yn ei threfn i Lundain):

|  |  |  |  |
| --- | --- | --- | --- |
| **Cyfeiriad** | **Tref** | **Pwyntydd(1)** | **Pwyntydd (2)** |
| 1 | Bridgend | 4 | 4 |
| 2 | Swansea | 7 | 1 |
| 3 | Reading | 5 | 0 |
| 4 | Cardiff | 6 | 5 |
| 5 | Swindon | 0 | 3 |
| 6 | Pembroke | 3 | 2 |
| 7 |  | 8 | 8 |
| 8 |  | 0 | 0 |
| Blaen (trefn yr wyddor)= 1 | | | |
| Blaen (gorsafoedd tren) = 6 | | | |
| Pwyntydd rhydd nesaf = 7 | | | |

**Nodwch :**

Nid oes angen symud cofnodion yn ffisegol i’w trefnu dim ond newid y pwyntyddion

Mae hefyd…

 **Rhestri cysylltiedig cylchol** (circular linked lists) - mae'r eitem olaf yn cysylltu nôl i'r eitem gyntaf.

 **Rhestri cysylltiedig dwyffordd** (two-way linked list) - mae pob cwgn yn cynnwys dau bwyntydd. Un i'r cwgn **nesaf** a'r llall i'r cwgn **blaenorol**.

 Ac wrth gwrs **Rhestri cysylltiedig cylchol dwyffordd** (two-way circular linked lists).

**Defnyddio rhestrau cyswllt**

Mae **rhestr gysylltiedig** yn strwythur data defnyddiol am gadw eitemau mewn trefn arbennig wrth iddynt gael eu hychwanegu, heb orfod trefnu.

Er enghraifft wrth redeg rhaglen sydd yn creu cyfrifon cwsmeriaid, gallai popeth a brynir gan gwsmer penodol yn ystod y mis cael ei storio fel rhestr gysylltiedig mewn trefn dyddiad, yn barod i'w argraffu.

**Tablau Stwnsh**

Tabl Stwnsh yw’r term rhoddir i strwythur data sy’n cynnwys dwy ran: tabl (neu arae) o ddata, ac allwedd sy'n nodi lleoliad y data yn y tabl. Rhedir **algorithm stwnsio** ar yr allwedd sydd wedyn yn gweithredu fel indecs ar gyfer y lleoliad a bennwyd eitem ddata honno yn yr arae.

Pan fydd angen y data eu hadalw, er enghraifft, os yw'r chwiliad ar y data, defnyddir yr un algorithm stwnsh ar yr allwedd sy'n cael ei chwilio i gyfrifo'r indecs ac felly canfod y data mewn un cam. Dyma dechneg chwilio effeithiol iawn, a dyma pam defnyddir tablau yn lle mae angen mynediad cyflym i angen setiau data.

**Enghraifft:** Efallai bydd cronfa ddata cwsmeriaid yn cael ei storio fel arae sy’n cynnwys miliynau o gofnodion gwsmeriaid gan gynnwys CustomerID, enw, cyfeiriad ac ati. Gellir defnyddio algorithm stwnsh ar y maes CustomerID a gellir ei ddefnyddio fel allwedd yn yr achos hwn. Byddai hyn yn creu indecs ar gyfer pob cwsmer, a fyddai'n pwyntio at leoliad y cofnod yn yr arae

Gellir gweld hwn isod:

**Allwedd Indecs Allwedd/Gwerth pâr**

014563 01 014564, Mary Jones, 14 Any Road

02

014564 03 014563, Joe Bloggs, 23 Rhodfa fawr

04

014565 05

06 014565, Gordon Pike, 4 Spender Road

Gellir rhagweld yr arae fel cyfres o leoliadau a bydd gan bob un indecs unigryw.

**Defnydd o algorithmau stwnsio**

 Cronfeydd data - i greu indecs ar gyfer cronfeydd ar gyfer galluogi storio ac adalw data cyflym

 Cyfeirio’r Cof - defnyddio i gynhyrchu cyfeiriadau cof lle caiff data ei storio.

 Systemau gweithredu - Rhai systemau gweithredu yn defnyddio algorithmau stwnsio i storio a dod o hyd i ffeiliau

 Amgryptio - defnyddir i amgryptio data. Yn yr achos hwn rhaid yr algorithm fod yn gymhleth iawn fel y ni all dadgryptio'r os yw data eu dwyn.

**Algorithmau stwnsio**

I gynhyrchu indecs, bydd angen ichi gael algorithm addas. Byddwn yn gyntaf yn edrych ar enghraifft syml iawn er mwyn dangos y cysyniad. Efallai bod gennych arae o 6 elfen. Gallwn gyfrifo indecs gan ddefnyddio algorithm sy'n adio rhifau (digid) yr allwedd at ei gilydd ac yna berfformio swm ‘modulo 6’ ar y canlyniad, gan fod chwech blwch yn ein tabl stwnsh:

|  |  |
| --- | --- |
| 0 |  |
| 1 |  |
| 2 | 25463 |
| 3 |  |
| 4 |  |
| 5 |  |

* Gwerth yr eitem ddata gyntaf efallai fyddai allwedd 25463
* Adio’r rhifau (digid) gyda'i gilydd 2 + 5 + 4 + 6 + 3 = 20
* Berfformio cyfrifiad modulo 6 felly rhannwch gan 6 = 3 r 2
* Felly yr indecs yw y gweddill – 2
* Rhoddir y data yn y blwch 2
* Gwerth yr ail eitem o ddata allweddol yw efallai 34255
* Adio’r rhifau nifer y gyda'i gilydd eto 3 + 4 + 2 + 5 + 5 = 19
* Berfformio ‘modulo 6’ ar y swm = 3 r 1

|  |  |
| --- | --- |
| 0 |  |
| 1 | 34255 |
| 2 | 25463 |
| 3 |  |
| 4 |  |
| 5 |  |

* Felly'r indecs yw 1
* Rhoddir y data ym mlwch 1

Mae'r broses yn parhau ar gyfer pob allwedd. Gallwch weld yn hawdd o'r darlun uchod sut y crëir yr indecs. Mantais go iawn ddefnyddio algorithm yw eich bod yn defnyddio i storio data yn y lle cyntaf ac wedyn ei ddefnyddio i ganfod y data pan fo angen. Felly grëwyd ac ail-grewyd y gwahanol indecs pan fydd eu hangen.

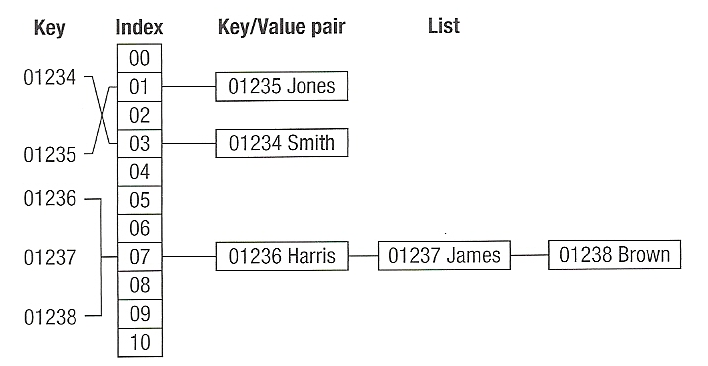
### Gwrthdrawiadau (*Collisions*)

Un o brif nodweddion algorithm stwnsh yw bod rhaid cynhyrchu **indecs unigryw**. Pan fo gwrthdrawiad yn digwydd, mae rhaid bod rhyw ffordd o ymdrin â hynny fel y gellir neilltuo indecs unigryw i’r allwedd.

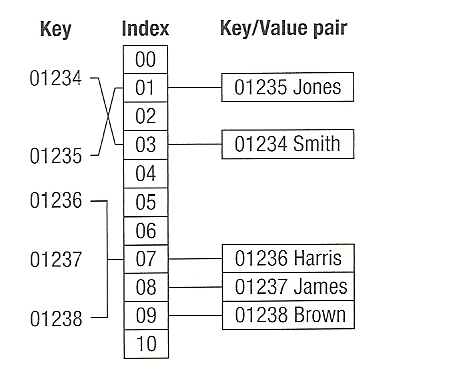
Mae yna ddau brif ffordd o wneud hwn:

 **‘Chaining’** - Yma, os mae yna wrthdrawiad yn digwydd, crëir rhestr yn y blwch hwnnw ac mae’r allwedd/gwerth yn troi yn elfennau o'r rhestr. Os digwyddir damwain arall mae’r allwedd/gwerth hwnnw yw’r elfen nesaf ar y rhestr ac ati.

Mae'r diagram isod yn dangos y cysyniad hwn:



* **Ail-stwnsio** – Yn yr achos hwn os oes gwrthdrawiad yn digwydd, rhedir yr algorithm eto, tan y crëir allweddol unigryw. Defnyddir techneg o'r enw ‘probing’, sy'n golygu bod yr algorithm yn chwilio am bwlch gwag. Gallai wneud hyn drwy edrych yn syml am y blwch rhydd nesaf sydd ar gael i'r indecs lle nad oes gwrthdar. Mae'r diagram isod yn dangos chwiliad llinol syml lle y defnyddir y slot nesaf sydd ar gael.



Nid yw’r broses yma yn soffistigedig iawn oherwydd os yw'r algorithm stwnsh sy'n arwain at glystyru fel yn yr enghraifft, bydd y canlyniadau'n dal i fod mewn grŵp o gwmpas yr un blwch.