**Linked list**

تعد القوائم المترابطة إحدى بنى المعطيات الخطية التي تقوم بتخزين العناصر الموجودة في أماكن مختلفة في الذاكرة وترتبط هذه العناصر مع بعضها بواسطة مؤشرات

تتآلف من عدد من العقد المرتبطة ببعضها وتحوي كل عقدة على قيمة للعنصر ومؤشر يشير إلى العنصر الذي يليه

يدعى عنوان العقدة الأولى Head بينما عنوان العقدة الأخيرة غير معرف لأن العقدة الأخيرة لا تشير إلى شيء أي إلى NULL



**أهم ما يميز القوائم المترابطة:**

تكون العناصر مرتبة بشكل عشوائي في الذاكرة

عند استخدام القوائم المترابطة فإن الذاكرة تحجز في وقت التنفيذ runtime

ذات حجم متغير وتمتاز بالمرونة والقابلية لتكبير أو تصغير حجمها حسب الحاجة

سهولة إضافة عناصر وحذفها من أي مكان

**مساوئ القوائم المترابطة:**

لا يمكن الوصول إلى عناصر القوائم المترابطة بطريقة عشوائية

تحتاج إلى مساحة تخزينية لان كل عنصر يرتبط بمؤشر

**تمثيل القوائم المترابطة :**

كل عقدة تتألف من:

* قيمة العنصر
* مؤشر يحمل عنوان عقدة أخرى
* struct node
* {
* int data;
* struct node \*next;
* };

لنقوم بإنشاء قائمة مترابطة بسيطة تتألف من ثلاث عناصر لفهم كيفية عملها:

/\* تهيئة العقد\*/

struct node \*head;

struct node \*one = NULL;

struct node \*two = NULL;

struct node \*three = NULL;

/\*تخصيص الذاكرة \*/

one = malloc(sizeof(struct node));

two = malloc(sizeof(struct node));

three = malloc(sizeof(struct node));

/\* تعيين قيم البيانات\*/

one->data = 1;

two->data = 2;

three->data=3;

/\* وصل العقد \*/

one->next = two;

two->next = three;

three->next = NULL;

/\* حفظ عنوان أول عنصر في الرأس\*/

head = one;

بهذه الخطوات البسيطة قمنا بإنشاء قائمة مترابطة مؤلفة من ثلاث عناصر :



تطبيق عملي :

// Linked list implementation in C++

#include <bits/stdc++.h>

#include <iostream>

using namespace std;

// إنشاء عقدة

class Node {

public:

int value;

Node\* next;

};

int main() {

Node\* head;

Node\* one = NULL;

Node\* two = NULL;

Node\* three = NULL;

// heap تخصيص ثلاث عقد في الذاكرة

one = new Node();

two = new Node();

three = new Node();

// valueتعيين قيم ل

one->value = 1;

two->value = 2;

three->value = 3;

// وصل العقد

one->next = two;

two->next = three;

three->next = NULL;

// طباعة قيم القائمة المترابطة

head = one;

while (head != NULL) {

cout << head->value;

head = head->next;

}

**العمليات على السلاسل المترابطة :**

**الانتقال عبر القائمة المترابطة :**

يمكن عرض محتويات القائمة المترابطة بشكل بسيط وذلك عن طريق الاستمرار في نقل العقدة المؤقتة ونقوم بعرض محتواها عندما يكون فارغ أي NULL نعلم حينها أننا وصلنا إلى نهاية القائمة ونخرج من حلقة التكرار

struct node \*temp = head;

cout<<”List elements are \_ \n”;

while(temp != NULL) {

cout<<”--->”<<temp->data;

temp = temp->next;

}

عندها يكون الخرج :

List elements are -

1 --->2 --->3 --->

**إضافة عنصر إلى القائمة المترابطة:**

يمكن أضافة عنصر في البداية أو في المنتصف أو في النهاية

**أضافة عنصر في البداية:**

* 1. تهيئة الذاكرة للعقدة الجديدة
  2. تخرين بيانات العنصر
  3. جعل مؤشر next الخاص بالعقدة الجديدة يشير إلى رأس القائمةhead
  4. تغير الرأس head وجعله العقدة التي تم إنشاءها مؤخرا

1. struct node \*newNode;
2. newNode = malloc(sizeof(struct node));
3. newNode->data = 4;
4. newNode->next = head;
5. head = newNode;

**أضافة عنصر في النهاية:**

* 1. تهيئة الذاكرة للعقدة الجديدة
  2. تخرين بيانات العنصر
  3. الانتقال إلى العقدة الأخيرة
  4. جعل المؤشر next الخاص بأخر عقدة هو العقدة التي تم إنشاءها مؤخرا

1. struct node \*newNode;
2. newNode = malloc(sizeof(struct node));
3. newNode->data = 4;
4. newNode->next = NULL;
5. struct node \*temp = head;
6. while(temp->next != NULL){
7. temp = temp->next;
8. }
9. temp->next = newNode;
10. **أضافة عنصر في المنتصف :**
11. تهيئة الذاكرة وتخزين قيمة العنصر للعقدة الجديدة
12. الانتقال إلى العقدة التي تسبق الموقع المطلوب للعقدة الجديدة
13. تغير المؤشرات next التالية لتضمين العقدة الجديدة
14. struct node \*newNode;
15. newNode = malloc(sizeof(struct node));
16. newNode->data = 4;
17. struct node \*temp = head;
18. for(int i=2; i < position; i++) {
19. if(temp->next != NULL) {
20. temp = temp->next;
21. }
22. }
23. newNode->next = temp->next;
24. temp->next = newNode;

**حذف عنصر من القائمة المترابطة :**

يمكن الحذف من البداية أو المنتصف أو النهاية

**الحذف عنصر من البداية :**

جعل الرأس head هو العقدة الثانية

**الحذف من النهاية :**

1. الانتقال إلى العنصر قبل الأخيرة
2. جعل المؤشر الخاص به يشير إلى Null

**الحذف من المنتصف :**

1. الانتقال إلى العنصر الذي يسبق العنصر الذي نريد حذفه
2. تغير المؤشرات التالية لاستبعاد هذا العنصر من القائمة
3. for(int i=2; i< position; i++) {
4. if(temp->next!=NULL) {
5. temp = temp->next;
6. }
7. }
8. temp->next = temp->next->next;

**البحث عن عنصر في القائمة المترابطة :**

يمكننا البحث عن عنصر معين في القائمة باستخدام حلقة التكرار عن طريق الخطوات التالية:

1. جعل الرأس head هو العقدة الحالية
2. البدء في تنفيذ الحلقة حتى تصبح العقدة الحالية NULL لأن أخر عنصر يشير إلى NULL
3. في كل تكرار تحقق إذا كان العنصر الذي نبحث عنه key مساويا لقيمة العنصر حينها يرجع true وإلا يرجع false
4. // البحث عن عنصر
5. bool searchNode(struct Node\*\* head\_ref, int key) {
6. struct Node\* current = \*head\_ref;
7. while (current != NULL) {
8. if (current->data == key) return true;
9. current = current->next;
10. }
11. return false;
12. }

**ترتيب عناصر القائمة المترابطة :**

**نحن نستخدم خوارزمية ترتيب بسيطة وهي bubble sort لترتيب عناصر القائمة بشكل تصاعدي**

1. جعل الرأس head هو العقدة الحالية وإنشاء عداد index للاستخدام لاحقا
2. إذا كان head فارغ أي NULLينتهي البرنامج
3. وإلا تنفيذ الحلقة حتى نصل إلى أخر عقدة
4. في كل تكرار اتبع الخطوات e و f
5. نقوم بتخزين المؤشر الخاص بالعقدة الحالية في index
6. تحقق إذا كانت قيمة العنصر للعقدة الحالية أكبر من العقدة التالية حينها نقوم بالمبادلة بين index والعقدة الحالية
7. // ترتيب القائمة المترابطة
8. void sortLinkedList(struct Node\*\* head\_ref) {
9. struct Node \*current = \*head\_ref, \*index = NULL;
10. int temp;
11. if (head\_ref == NULL) {
12. return;
13. } else {
14. while (current != NULL) {
15. // يشير إلى المؤشر الخاص بالعقدة الحاليةindex
16. index = current->next;
17. while (index != NULL) {
18. if (current->data > index->data) {
19. temp = current->data;
20. current->data = index->data;
21. index->data = temp;
22. }
23. index = index->next;
24. }
25. current = current->next;
26. }
27. }
28. }

**التعقيد الزمني للسلاسل المترابطة:**

|  |  |  |
| --- | --- | --- |
|  | الحالة الأسوء | الحالة المتوسطة |
| **البحث** | O(n) | O(n) |
| **الإضافة** | O(n) | O(n) |
| **الحذف** | O(1) | O(1) |

**أنواع القوائم المترابطة :**

* **القوائم المترابطة المفردة :**

وهي الأكثر انتشارا وكل عقدة تتألف من قيمة العنصر ومؤشر يشير إلى العنصر التالي

****

**العقدة تكون على الشكل التالي :**

struct node {

int data;

struct node \*next;

}

**يمكن إنشاء قائمة مترابطة مفردة تتألف من ثلاث عناصر على الشكل التالي :**

/\* إنشاء العقد\*/

struct node \*head;

struct node \*one = NULL;

struct node \*two = NULL;

struct node \*three = NULL;

/\* تهيئة الذاكرة \*/

one = malloc(sizeof(struct node));

two = malloc(sizeof(struct node));

three = malloc(sizeof(struct node));

/\* ÷عطاء قيم للعناصر\*/

one->data = 1;

two->data = 2;

three->data = 3;

/\* توصيل العقد \*/

one->next = two;

two->next = three;

three->next = NULL;

/\* head حفظ العنوان الخاص بأخر عقدة في ا\*/

head = one;

* **القوائم المترابطة المزدوجة :**

نقوم بإضافة مؤشر إلى للعقدة السابقة في القائمة المترابطة المزدوجة يمكننا حينها الذهاب في كلا الاتجاهين إلى الامام أو إلى الخلف

****

**العقدة تكون على الشكل التالي :**

struct node {

int data;

struct node \*next;

struct node \*prev;

}

**يمكن إنشاء قائمة مترابطة مزدوجة تتألف من ثلاث عناصر على الشكل التالي :**

/\* إنشاء العقد\*/

struct node \*head;

struct node \*one = NULL;

struct node \*two = NULL;

struct node \*three = NULL;

/\* تخصيص الذاكرة \*/

one = malloc(sizeof(struct node));

two = malloc(sizeof(struct node));

three = malloc(sizeof(struct node));

/\* إعطاء قيم \*/

one->data = 1;

two->data = 2;

three->data = 3;

/\* توصيل العقد \*/

one->next = two;

one->prev = NULL;

two->next = three;

two->prev = one;

three->next = NULL;

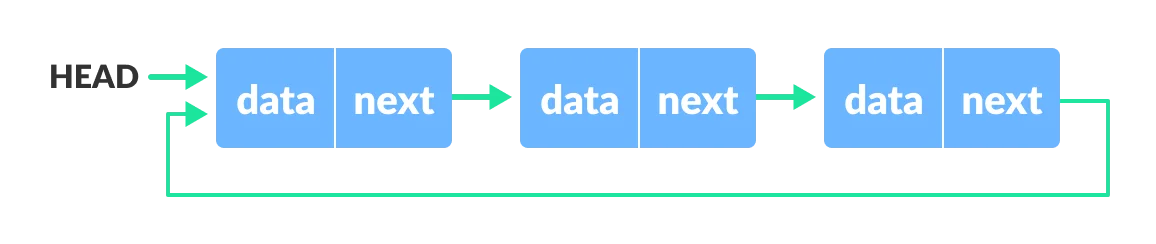
three->prev = two;

/\* head حفظ العنوان الخاص بأخر عقدة في ا\*/

head = one;

* **القائمة المترابطة الدائرية :**

في هذا النوع يرتبط العنصر الأخير من القائمة المترابطة بالعنصر الأول



القائمة المترابطة الدائرية يمكن أن تكون مفردة أو مزدوجة

في القائمة المفردة المؤشر الخاص بالعنصر الأخير يشير إلى العنصر الأول

في القائمة المزدوجة المؤشر prv الخاص بالعنصر الأول يشير إلى العنصر الأخير أيضا

يمكن إنشاء قائمة مترابطة دائرية مفردة تتألف من ثلاث عناصر على الشكل التالي :

/\* إنشاء العقد \*/

struct node \*head;

struct node \*one = NULL;

struct node \*two = NULL;

struct node \*three = NULL;

/\* تخصيص الذاكرة \*/

one = malloc(sizeof(struct node));

two = malloc(sizeof(struct node));

three = malloc(sizeof(struct node));

/\* تعيين قيم للعناصر\*/

one->data = 1;

two->data = 2;

three->data = 3;

/\* توصيل العقد \*/

one->next = two;

two->next = three;

three->next = one;

/\* head حفظ العنوان الخاص بأخر عقدة في ال\*/

head = one;

**تطبيقات القوائم المترابطة :**

تخصيص الذاكرة الديناميكي

يستخدم في إنشاء المكدس وقائمة الانتظار

جداول التجزئة والرسوم البيانية

يستخدم في التراجع عن وظائف البرامج