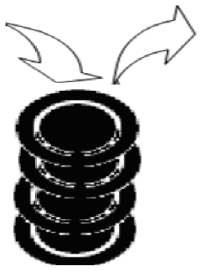
الأكوام
Stacks

1- تعريف

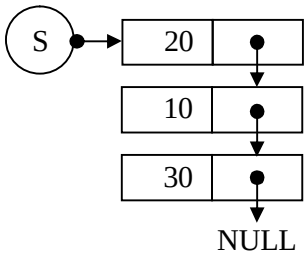
الكومة مجموعة من العناصر المرصوفة فوق بعضها بحيث تتم الإضافة والحذف دائما من القمة (The top of the stack).



آخر عنصر يتم إضافته إلى الكومة هو أول عنصر يتم إخراجه مما يعني أن معالجة عناصر الكومة تتم حسب مبدأ (Last In First Out - LIFO).

2- تمثيل الكومة

تعتبر الكومة نوع خاص من القوائم المتصلة التي تطبق فيها عمليات الإضافة والحذف دائما على العنصر الأول (رأس القائمة).



في لغة C++ يمكن الإعلان عن الكومة بستخدام البناء التالي:

```
struct Node
{
        int Data;
        Node  *Next;
};

Node  *S;
```

تتم معالجة الكومة عن طريق الدوال الخمسة التالية:

الدالة	دورها
Node* InitStack(Node *S)	استهلال الكومة S لتكون فارغة في البداية

bool IsEmpty(Node *S)	التحقق إذا ما كانت الكومة S فارغة أم لا
Node* Push(int x, Node *S)	إضافة العنصر x إلى قمة الكومة S
int Top(Node *S)	الحصول على نسخة من قمة الكومة S
Node* Pop(Node *S)	حذف العنصر الأول (قمة الكومة S)

يقوم البرنامج التالي بإنشاء كومة فارغة ثم يضيف إليها 3 عناصر. بعد ذلك يقوم البرنامج بحذف رأس الكومة وعرض بقية العناصر على الشاشة.

```
#include <iostream>
using namespace std;

struct Node
{
        int Data;
        Node  *Next;
};

Node* InitStack(Node *S)
{
        S = NULL;
        return S;
}

bool IsEmpty(Node *S)
{
        if (S == NULL)
              return true;
        else
              return false;
}

Node* Push(int x, Node *S)
{
        Node *P = new(Node);
        P -> Data = x;
        P -> Next = S;
        S = P;
        return S;
}

int Top(Node *S)
{
        if (!IsEmpty(S))

              return S->Data;
        else
```

```
                cout<<"Stack empty ...";    //underflow

}

Node* Pop(Node *S)
{
        if (!IsEmpty(S))
        {
                Node *P = S;
                S = S ->Next;
                delete P;
                return S;
        }
        else
                cout<<"Stack empty ...";
}

void main()
{
        Node *S;
        S = InitStack(S);
        S = Push(10, S);
        S = Push(5,S);
        S = Push(2,S);
        S = Pop(S);
        while (!IsEmpty(S))
        {
                cout<<Top(S)<<"\t";
                S=Pop(S);
        }
        cout<<endl;
}
```

مخرجات البرنامج

```
5       10
Press any key to continue
```

تمرين: قم بتعريف تصنيف (class) يحتوي على البيانات والوظائف الضرورية لمعالجة الأكوام.

```
#include <iostream>
using namespace std;

class Stack
{
        private:
                int Item;
                Stack *Next;
        ……..
                Stack* InitStack();
                bool IsEmpty();
                Stack* Push(int)
                int Top()
                ……..
                void Print();
};

Stack* Stack::InitStack()
{
        ……..;
}

bool Stack::IsEmpty()
{
        if (this==NULL)
                return true;
        else
                ……..;
}

Stack* Stack::Push(int x)
{
        ….
}

int Stack::Top()
{
        ….
}

Stack* Stack::Pop()
{
        ….
}
```

ملاحظة: يمكن إنشاء كومة بدون استخدام المؤشرات وإنما باستخدام مصفوفة كما في المثال التالي:

```
// Mystack.h

class Stack
{
      private:
            int q[100];
            int pointer;
      public:
            Stack()
            {
                  pointer = 0;
            }
            void put(int i);
            int get();
};
```

```
//MyStack.cpp

#include <iostream>
#include "MyStack.h"

using namespace std;

void Stack::put(int i)
{
      if (pointer == 100)
            cout<<"Stack is full ..."<<endl;
      else
      {
            q[pointer] = i;
            pointer++;
      }
}

int Stack::get()
{
      if (pointer == -1)
      {
            cout<<"Stack is underflow ..."<<endl;
            return 0;
      }
      else
      {
            pointer--;
            return q[pointer];
      }
}

int main()
{
      Stack A;
```

```
      A.put(10);
      A.put(20);
      A.put(30);
      cout<<A.get()<<endl;
      cout<<A.get()<<endl;
      cout<<A.get()<<endl;
      return 0;
}
```

مخرجات البرنامج (Output)