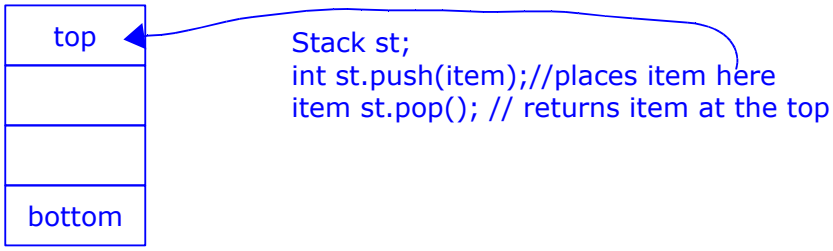
Stacks

Stacks are data structures that allow access to the last added item. "Popping" the stack removes the last added item.



The following two examples show stack **implementations using arrays**.

```
/* stack1.cpp */

#include<iostream>
using namespace std;

const int MAX_CAPACITY=128;
typedef int StackObject;

class Stack{

private:

  StackObject myArray[MAX_CAPACITY];
  int TopObjectNum;
public:
  Stack(){TopObjectNum=-1;} //Constructor
  bool empty()const;
  void push(const StackObject&); //Put an element in the stack
  StackObject top() const; //Get value of top object in the stack
  void pop(); // get rid of top element of the stack

};

bool Stack::empty() const {
  return (TopObjectNum == -1);
}

void Stack::push(const StackObject& o){
  if(TopObjectNum < MAX_CAPACITY -1){
    ++TopObjectNum;
    myArray[TopObjectNum]= o;
  }
  else{
    cerr<<"Stack is full"<<endl;
```

```
  }
}


StackObject Stack::top() const{

  if (TopObjectNum >= 0 ){
    return myArray[TopObjectNum];
  } else {
    cerr<<"Stack is empty\n";
  }
}

void Stack::pop(){
  if (TopObjectNum>=0){
    TopObjectNum--;
  } else{
    cerr<<"Stack is empty\n";
  }
}

void Stack::display(ostream& o) const{
  o<<"Stack contents:\n";
  for(int i=TopObjectNum; i>= 0 ; i--){
    o<<myArray[i]<<endl;
  }
}



int main(){
  Stack s;
  cout<<"Is stack s empty?"<<s.empty()<<endl;
  s.push(10);
  s.push(100);
  s.push(89);
  cout<<"Is stack s empty?"<<s.empty()<<endl;
  cout<<"s.top->" <<s.top()<<endl;
  s.display(cout);
  s.pop();
  cout<<"s.top->" <<s.top()<<endl;
  s.top();
  s.display(cout);
}
```

```
/* stack2.cpp */

#include<iostream>
using namespace std;

class A{
private:
  int i;
  char c;
public:
  A(int=0,char='A');
  void display();
};
A::A(int a, char b){
  i=a;
  c=b;
}

void A::display(){
  cout<<"--"<<endl;
  cout<<"i-> "<<i<<endl;
  cout<<"c-> "<<c<<endl;
}

const int MAX_CAPACITY=128;
typedef A StackObject;


class Stack{

private:

  StackObject myArray[MAX_CAPACITY];
  int TopObjectNum;
public:
  Stack(){TopObjectNum=-1;} //Constructor
  bool empty()const;
  void push(const StackObject&); //Put an element in the stack
  StackObject top(); //Get value of top object in the stack
  void pop(); // get rid of top element of the stack
  void display() ;
};

bool Stack::empty() const {
  return (TopObjectNum == -1);
}

void Stack::push(const StackObject& o){
  if(TopObjectNum < MAX_CAPACITY -1){
    ++TopObjectNum;
    myArray[TopObjectNum]= o;
  }
  else{
```

```
    cerr<<"Stack is full"<<endl;
  }
}


StackObject Stack::top() {

  if (TopObjectNum >= 0 ){
    return myArray[TopObjectNum];
  } else {
    cerr<<"Stack is empty\n";
  }
}

void Stack::pop(){
  if (TopObjectNum>=0){
    TopObjectNum--;
  } else{
    cerr<<"Stack is empty\n";
  }
}

void Stack::display() {
  cout<<"Stack contents:\n";
  for(int i=TopObjectNum; i>= 0 ; i--){
    myArray[i].display();
  }
}



int main(){
  A a1,a2(11,'B'),a3(12,'C'),a4(13,'D');
  Stack s;
  s.push(a1);
  s.push(a2);
  s.push(a3);
  s.display();
  s.pop();
  s.display();
}
```