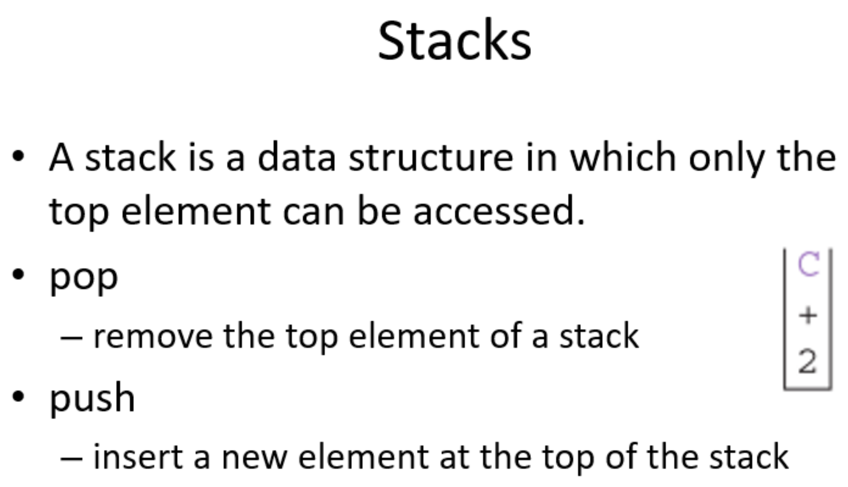
CS 210 **Oct. 6, 2016**
Day 12

More of Ch. 7 – Array Pointers



Stacks

- A stack is a data structure in which only the top element can be accessed.
- pop
  - remove the top element of a stack
- push
  - insert a new element at the top of the stack

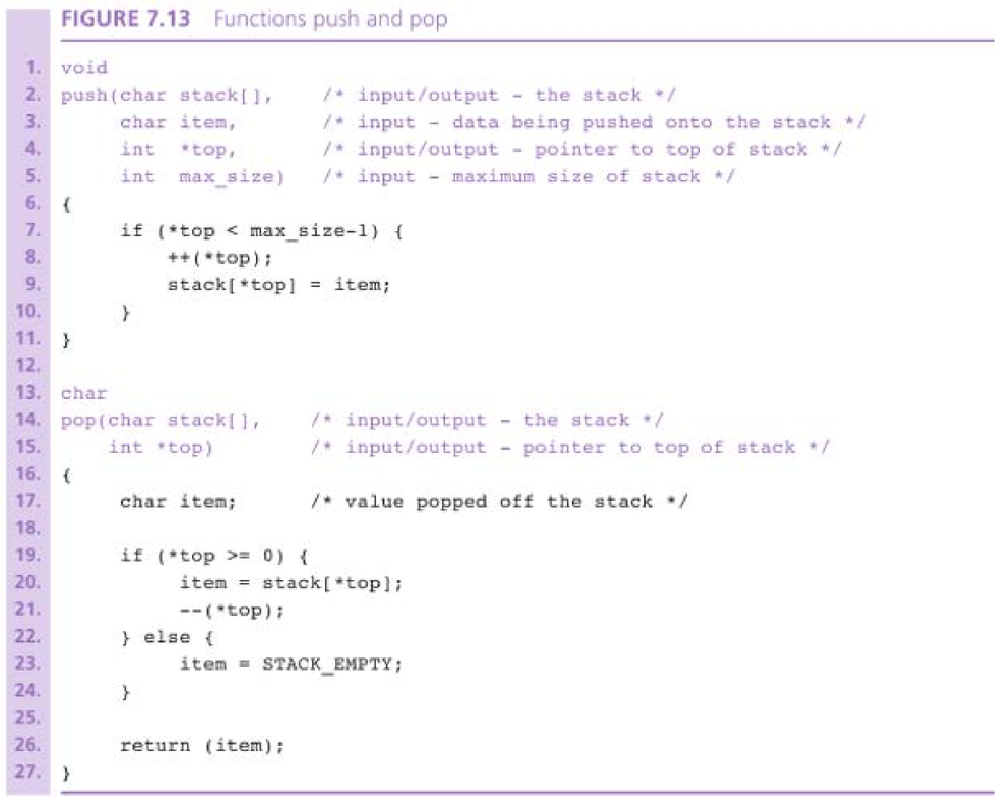


FIGURE 7.13 Functions push and pop

```
void
push(char stack[],    /* input/output - the stack */
     char item,       /* input - data being pushed onto the stack */
     int  *top,       /* input/output - pointer to top of stack */
     int  max_size)   /* input - maximum size of stack */
{
     if (*top < max_size-1) {
         ++(*top);
         stack[*top] = item;
     }
}

char
pop(char stack[],    /* input/output - the stack */
    int *top)        /* input/output - pointer to top of stack */
{
     char item;      /* value popped off the stack */

     if (*top >= 0) {
          item = stack[*top];
          --(*top);
     } else {
          item = STACK_EMPTY;
     }

     return (item);
}
```

Stacks are used almost universally to save the return address when a function is called. If you ever used an HP calculator with Reverse Polish Notation, you have a better idea of what a stack can do in terms of arithmetic.

Selection Sort

```
1.  for each value of fill from 0 to n-2
    2.  Find index_of_min, the index of the smallest
        element in the unsorted subarray list[fill]
        through list[n-1]
    3.  if fill is not the position of the smallest element
        (index_of_min)
        4.  Exchange the smallest element with the one at
            position fill.
```

Selection Sort

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void SelectionSort(int d[], int n);
void Swap(int *x, int *y);
void FindMin(int d[], int n, int start, int *minIndx);
int main()
{
   int i;
   int data[100];
   srand(23);
   for(i=0;i<100;i++)
      data[i] = rand() % 101;
   for(i=0;i<100;i++)
      printf("%d, ", data[i]);
   printf("\n\n");
   SelectionSort(data, 100);
   for(i=0;i<100;i++)
      printf("%d, ", data[i]);
   printf("\n");
}
void SelectionSort(int d[], int n)
    {int i, j, swpCnt, minIndx;
     swpCnt = 0;
     for(i=0;i<n-1;i++)
        {for(j=i;j<n;j++)
           {FindMin(d, n, i, &minIndx);
            Swap(&d[i], &d[minIndx]);
            swpCnt++;
            }
        }
     printf("Selection sort done with %d swaps.\n", swpCnt);
    }
void FindMin(int d[], int n, int start, int *minIndx)
   {int i;
    *minIndx = start;
    for(i=start;i<n;i++)
       {if(d[i] < d[*minIndx])
           *minIndx = i;
       }
   }
void Swap(int *a, int *b)
   {int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
   }
```

Bubble Sort

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void BubbleSort(int d[], int n);
void Swap(int *x, int *y);
int main()
{
   int i;
   int data[100];
   srand(23);
   for(i=0;i < 100;i++)
      data[i] = rand() % 101;
   for(i=0;i < 100;i++)
      printf("%d, ", data[i]);
   printf("\n\n");
   BubbleSort(data, 100);
   for(i=0;i < 100;i++)
      printf("%d, ", data[i]);
   printf("\n");
}
void BubbleSort(int d[], int n)
   {int i, swpCnt, fDone;
    fDone = 0;
    swpCnt = 0;
    while(!fDone)
        {fDone = 1;
         for(i=0;i<n-1;i++)
             {if(d[i] > d[i+1])
                 {Swap(&d[i], &d[i+1]);
                  fDone = 0;
                  swpCnt++;
                 }
             }
         }
     printf("Bubble sort done with %d swaps.\n", swpCnt);
     }
void Swap(int *a, int *b)
   {int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
   }
```

Enumerated Types

- enumerated type
  - a data type whose list of values is specified by the programmer in a type declaration
- enumeration constant
  - an identifier that is one of the values of an enumerated type

```
typedef enum
        {Monday, Tuesday, Wednesday, Thursday,
         Friday, Saturday, Sunday}
day_t;
```

```
#include <stdio.h>
enum day { sunday, monday, tuesday, wednesday, thursday, friday, saturday };
int main()
{
    enum day today = wednesday;
    printf("Day %d\n",today+1);
    return 0;
}
//Day 4
//Press any key to continue . . .
```

Alternatively, you can write it like this:

```
#include <stdio.h>

typedef enum
{sunday, monday, tuesday, wednesday, thursday, friday, saturday
}day;

int main()
{
    day today = wednesday;
    printf("Day %d\n",today+1);
    return 0;
}
//Day 4
//Press any key to continue . . .
```

This is a way to create a Boolean variable type:

```
#include<stdio.h>
enum Boolean {false, true};  //false is 0 and true is 1
int main()
{
    enum Boolean flag;
    flag = false;
    printf("%d\n",  flag);
}
```

CS 210 **October 6, 2016**

Sorting characters

The code segment below creates an array named `data` which has 100 random lower case characters. Use this segment in your own main program to create an array of 100 random lower case characters. Print your characters to the screen with one space between each character.

```
int i;
char data[100];
srand(23);
for(i=0;i < 100;i++)
   data[i] = (char)((rand() % 26) + 'a');
for(i=0;i < 100;i++)
   printf("%c, ", data[i]);
printf("\n\n");
```

Write a sorting function (either a bubble sort or a section sort) to sort your data. Call the sorting program from your main program and again print the characters with one space between each to show that they were successfully sorted.

Turn in a printed copy of your source file.