CRICOS PROVIDER 00123M

School of Computer Science

COMP SCI 1103/2103 Algorithm Design & Data Structure

Review of Pointers

adelaide.edu.au

seekLIGHT

Overview

- Arrays and strings and pointers
- Stack
- Heap

Previously on ADDS

- Pointer: the memory address of a variable.
- How to define and use them?

```
double d;
double * ptr=&d;
cout << ptr <<endl << *ptr << endl;
```

Pointers and Arrays

- In C++, an array variable is like a pointer variable that points to the first indexed variable of the array.

```
int *ptr;
int a[10];
int i;

for(i = 0; i<10; i++){
  a[i] = i*2;
 }

ptr = a;

for(i = 0; i<10; i++){
  cout << ptr[i] << " ";
 }
cout << endl;

ptr[5] = 5;

for(i = 0; i<10; i++){
  cout << a[i] << " ";
 }
cout << endl;
```

By the way! You can go beyond the size of the array
What is a segmentation fault?

0 2 4 6 8 10 12 14 16 18

Iterating through ptr is the same as iterating through array a.

0 2 4 6 8 5 12 14 16 18

Pointers and Arrays

- We said that we can define a pointer and assign the (address of the) array to it, and work with it just like we work with an array
- What happens here:

- int b[10];
- b=ptr;
- Cout<<b[0];

You cannot change the pointer value in an array variable. Why?
Other example!

C-string

- A pointer-based string in C++ is an array of characters ending with the null terminator ('\0').
- The null terminator indicates where a string terminates in memory.
- A C-string can be accessed via a pointer

```
char city[9] = "Adelaide";
char *ptrCity = "Adelaide";
```

char city[] = {'A', 'd', 'e', 'l', 'a', 'i', 'd', 'e'}; is it equivalent to the two defined above?!

```
cout << city[1] << endl;
cout << *(city+1) << endl;
cout << ptrCity[1] << endl;
cout << *(ptrCity+1) << endl;
```

More examples on pointers and arrays

```
1 main()
2 {int a[3]={2012,2,14};
3 int* p1=&a[0];
4 int* p2=a;
5 }
```

Addr	Name	Value
0xbffff374	a[0]	2012
0xbffff375		
0xbffff376		
0xbffff377		
0xbffff378	a[1]	2
0xbffff379		
0xbffff37a		
0xbffff37b		
0xbffff37c	a[2]	14
0xbffff37d		
0xbffff37e		
0xbffff37f		
0xbffff380	p1	
0xbffff381		
0xbffff382		
0xbffff383		
0xbffff384	p2	
0xbffff385		
0xbffff386		
0xbffff387		

a=?
&a=?
a+1 = ?
*(a+2) = ?

*(a+2) = 3; What happened?

*(a+3) = ?

*(a+3) = 0xbffff37c;
What happened?

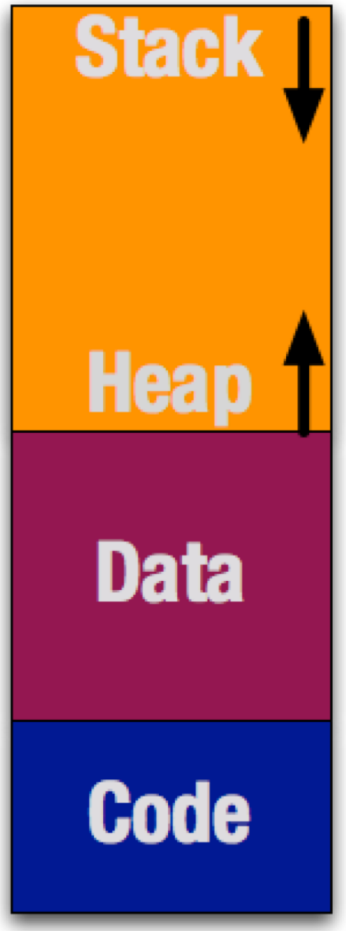
Example

```
1 main()
2 {char s1[]="abc";
3 char* s2="def";
4 char* p1=s1;
5 }
```

Addr	Name	Value
0x8048484	*s2	'd'
0x8048485	*(s2+1)	'e'
0x8048486	*(s2+2)	'f'
0x8048487	*(s2+3)	'\0'
⋮		
0xbffff38c	s1[0]	'a'
0xbffff38d	s1[1]	'b'
0xbffff38e	s1[2]	'c'
0xbffff38f	s1[3]	'\0'
0xbffff390	s2	
0xbffff391		
0xbffff392		
0xbffff393		
0xbffff394	p1	
0xbffff395		
0xbffff396		
0xbffff397		

Stack and Heap

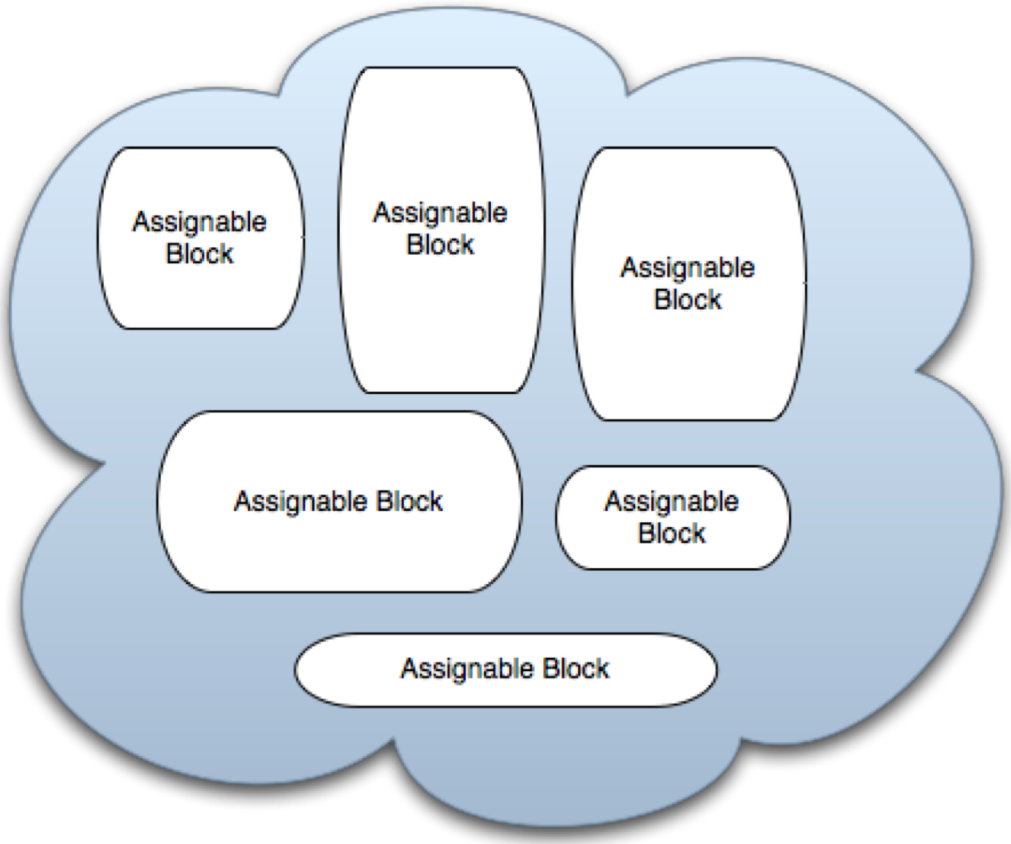
- What is stack?
- What is heap?
- What is the difference between them?



Example

- A student record management system
  - How can we store the student record?
  - How much memory do we need for a course?
  - What if some students enroll or drop out from the course?
  - The number of students is not known at the beginning
  - We need to add and remove student records dynamically

The Heap

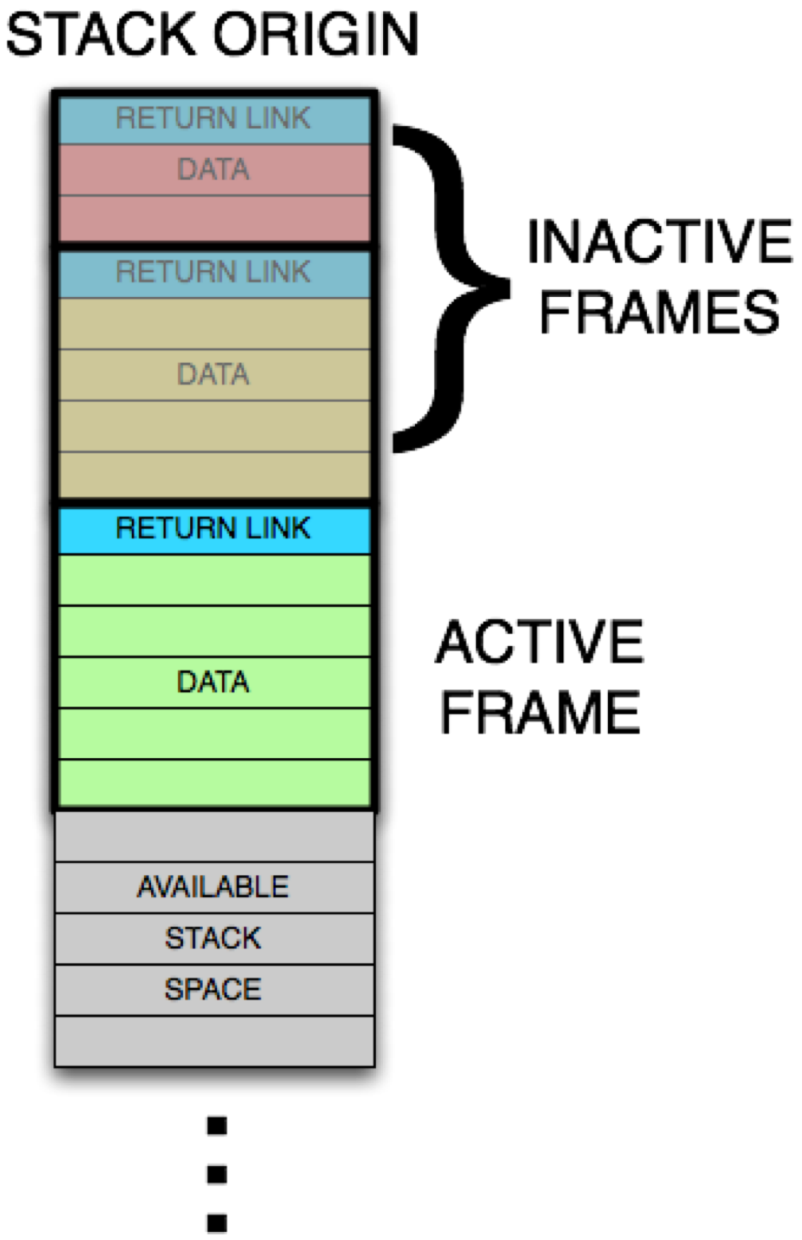


An area reserved for dynamic variables. It is also called the freestore.

Two drawbacks:

1. Searching
2. Heap fragmentation

The Stack



Last In First Out

For allocating memory to some new variable, we just need to keep track of where the free block starts.

Questions?