BBM 201 Data Structures Hacettepe University

Lecture 3: Pointers and Arrays

Lecturer: Lale Özkahya

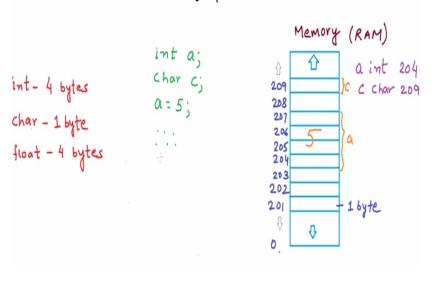
Resources:

Lecture Notes by Jason Zych, "Data Structures"
BBM201 Notes by Mustafa Ege and Sevil Şen
Lecture Videos
www.mycodeschool.com/videos/pointers-and-arrays

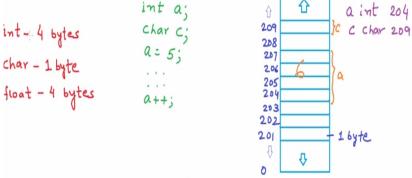
Outline

- 1 Introduction to Pointers
- Pointers and Arrays
- 3 Arrrays as Function Arguments
- 4 Character Arrays and Pointers
- 5 Character Arrays and Pointers Part II
- 6 Pointers and Multi-Dimensional Arrays

Introduction to pointers in C

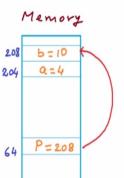


Introduction to pointers in C Memory (RAM) int a; 209 Char C; 208 a= 5; 207 206 205



Introduction to pointers in C

Pointers - variables that store
address of another
variable
int a;
int *P;
P = Aa;



Memory Pointers - Variables that Store address of another variable a=5 204 int a; int *P; P= 40; P=204 a = 5; Print P // 204 Print 4a 1/ 204 Print 2P 1/ 64 Print *P 115 => devergencing

Memory Pointers - Variables that store address of another variable a=8 204 int a; -P -> address int *P; *P-) value at P= 40; address P=204 a = 5; Print P // 204 Print 4a 1/ 204 Print 2P 1/ 64 Print *P 115 => deveroncing Print a 1/8

```
#include<stdio.h>
                          C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\Debug\SampleApp2.exe
int main()
                          Address of P is 3144576
                         Value at p is 10
                         Address of P is 3144576
     int a; int *p;
                         Value at p is 20
     a = 10;
     p = &a; // &a =
     printf("Address
     printf("Value at
     int b = 20;
     *p = b; // Will the address in p change to point b
     printf("Address of P is %d\n",p);
     printf("Value at p is %d\n",*p);
```

```
#include<stdio.h>
int main()
    int a = 10;
    int *p = &a;
    // p = &a; // &a = address of a
```

```
#include<stdio.h>
int main()
    int a = 10;
    int *p;
    p = &a;
    // Pointer arithmetic
    printf("%d\n",p); // p is 2002
    printf("%d\n",p+1); //_Ip+1 is ??
```

```
#include<stdio.h>
int main()
    int a = 10;
    int *p;
    p = &a;
    // Pointer arithmetic
    printf("Address p is %d\n",p); // p is 2002
    printf("size of integer is %d bytes\n", sizeof(int));
    printf("Address p+1 is %d\n",p+1); // p+1 is 2006
```

```
#include<stdio.h>
                          C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\Debug\SampleApp2.exe
int main()
                          Address p is 4586052
                          size of integer is 4 bytes
                          Address p+1 is 4586056
     int a = 10;
     int *p;
     p = &a;
     // Pointer arithmetic
     printf("Address p is %d\n",p); // p is 2002
     printf("size of integer is %d bytes\n", sizeof(int));
     printf("Address p+1 is %d\n",p+1); // p+1 is 2006
```

```
#include<stdio.h>
int main()
    int a = 10;
    int *p;
    p = &a;
   // Pointer arithmetic
    printf("Address p is %d\n",p);
    printf("value at address p is %d\n",*p);
    printf("size of integer is %d bytes\n", sizeof(int));
    printf("Address p+1 is %d\n",p+1);
    printf("value at address p+1 is %d\n",*(p+1));
}
```

```
#include<stdio.h>
                           animesh\documents\visual studio 2010\Proiects\SampleApp2\Debug\SampleApp2 exe
                      Address p is 2750832
int main()
                      value at address p is 10
                      size of integer is 4 bytes
                      Address p+1 is 2750836
    int a = 10;
                      value at address p+1 is -858993460
    int *p;
    p = &a;
    // Pointer arithmetic
    printf("Address p is %d\n",p);
    printf("value at address p is %d\n",*p);
    printf("size of integer is %d bytes\n", sizeof(int));
    printf("Address p+1 is %d\n",p+1);
    printf("value at address p+1 is %d\n",*(p+1));
```

```
#include<stdio.h>
                                                                         C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\Debug\SampleApp2\exe
                      Address p is 27<u>50832</u>
int main()
                       value at address p is 10
                       size of integer is 4 bytes
                       Address p+1 is 2750836
    int a = 10;
                      value at address p+1 is -858993460
    int *p;
    p = &a:
    // Pointer arithmetic
    printf("Address p is %d\n",p);
    printf("value at address p is %d\n",*p);
    printf("size of integer is %d bytes\n", sizeof(int));
    printf("Address p+1 is %d\n",p+1);
    printf("value at address p+1 is %d\n",*(p+1));
```

Outline

- 1 Introduction to Pointers
- 2 Pointers and Arrays
- 3 Arrrays as Function Arguments
- Character Arrays and Pointers
- 5 Character Arrays and Pointers Part II
- 6 Pointers and Multi-Dimensional Arrays

Pointers and Arrays int x=5 int A[5] int -> 4 bytes int *p Arol A → 5×4 bytes P = Ax Print P 1/ 300 = 20 bytes A[1] Print *P 115 A [2] P=P+1 //304 A[3] 200 204 208 212 216 300 304 A[4] A[0] A[1] A[2] A[3] A[4] Print P 1/304 Print *P

Pointers and Arrays int A[5] int A[5] int -> 4 bytes int +p A → 5×4 bytes A To 7 P = 4 A[0] = 20 bytes Print P 1/ 200 A[1] Print *P 1/2 A [2] A[3] 204 208 212 300 304 A[4] A[0] A[1] A[2] A[3] A[4] Print P+2 1/208 Print +(P+2) 115

```
Pointers and Arrays
                                        int A[5]
int ALS]
             int -> 4 bytes
                                        int *P
              A → 5×4 bytes
A [o]
                                        P = A
                    = 20 bytes
                                         Print A // 200
A[1]
                                         Print *A 1/2
A [2]
A[3]
         200
              204 208 212 216
                                        300 304
A[4]
         A[0] A[1] A[2] A[3] A[4]
                                         Print A+1 1/204
 Element at index i -
          Address - AA[i] or (A +i)
                                         Print *(A+1) 114
          Value - A[i] or * (A+i)
```

```
// Pointers and Arrays
#include<stdio.h>
int main()
    int A[] = \{2,4,5,8,1\};
    printf("%d\n",A);
    printf("%d\n",&A[0]);
    printf("%d\n",A[0]);
```

printf("%d\n",*A);

```
// Pointers and Arrays
#include<stdio.h>
int main()
     int A[] = \{2,4,5,8,1\};
                                                                              - 0 X
                                   C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\Debug\SampleApp2.exe
                                   2881420
     printf("%d\n",A);
                                   2881420
     printf("%d\n",&A[0]);
     printf("%d\n",A[0]);
     printf("%d\n",*A);
```

```
// Pointers and Arrays
#include<stdio.h>
int main()
    int A[] = \{2,4,5,8,1\};
    int i:
    for(i = 0; i < 5; i++)
        printf("Address = %d\n",&A[i]);
        printf("Address = %d\n",A+i);
        printf("value = %d\n",A[i]);
        printf("value = %d\n",*(A+i););
```

```
- 0 X
        C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\Debug\SampleApp2.exe
#incl Address = 3734748
       Address = 3734748
int m
       value = 2
       value = 2
       Address = 3734752
       Address = 3734752
      1 | value = 4
       Address = 3734756
       Address = 3734756
       value = 5
       value = 5
       Address = 3734760
       Address = 3734760
       value = 8
       value = 8
       Address = 3734764
```

```
// Pointers and Arrays
#include<stdio.h>
int main()
    int A[] = \{2,4,5,8,1\};
    int i; T
    int *p = A;
    A++;
    for(int i = 0; i < 5; i++)
        printf("Address = %d\n",&A[i]);
        printf("Address = %d\n",A+i);
        printf("value = %d\n",A[i]);
        printf("value = %d\n",*(A+i));
```

```
// Pointers and Arrays
#include<stdio.h>
int main()
    int A[] = \{2,4,5,8,1\};
    int i:
    int *p = A:
    A++;// invalid
    p++ // valid
    for(int i = 0; i < 5; i++)
        printf("Address = %d\n",&A[i]);
        printf("Address = %d\n",A+i):
        printf("value = %d\n",A[i]);
        printf("value = %d\n".*(A+i)):
```

```
// Pointers and Arrays
#include<stdio.h>
int main()
    int A[] = \{2,4,5,8,1\};
    int i:
    int *p = A;
    p++;
    for(int i = 0; i < 5; i++)
        printf("Address = %d\n",&A[i]);
        printf("Address = %d\n",A+i);
        printf("value = %d\n",A[i]);
        printf("value = %d\n",*(A+i));
```

Outline

- 1 Introduction to Pointers
- 2 Pointers and Arrays
- 3 Arrrays as Function Arguments
- 4 Character Arrays and Pointers
- 5 Character Arrays and Pointers Part II
- 6 Pointers and Multi-Dimensional Arrays

```
// Arrays as function arguments
#include<stdio.h>
int SumOfElements(int A[], int size)
    int i, sum = 0;
    for(i = 0; i < size; i++)
        sum+=A[i];
    return sum;
int main()
    int A[] = \{1,2,3,4,5\};
```

int size = sizeof(A)/sizeof(A[0]);
int total = SumOfElements(A,size);

printf("Sum of elements = %d\n",total);

```
// Arrays as function arguments
#include<stdio.h>
int SumOfElements(int A[], int size)
    int i, sum = 0;
                                       C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\Debuq\S
    for(i = 0;i< size;i++)</pre>
                                       Sum of elements = 15
        sum+=A[i];
    return sum;
int main()
    int A[] = \{1,2,3,4,5\};
    int size = sizeof(A)/sizeof(A[0]);
    int total = SumOfElements(A, size);
    printf("Sum of elements = %d\n",total);
```

```
// Arrays as function arguments
#include<stdio.h>
int SumOfElements(int A[])
    int i, sum = 0;
    int size = sizeof(A)/sizeof(A[0]);
    for(i = 0;i< size;i++)</pre>
        sum+= A[i];
    return sum;
int main()
```

int $A[] = \{1,2,3,4,5\};$

int total = SumOfElements(A);

printf("Sum of elements = %d\n",total);

```
// Arrays as function arguments
#include<stdio.h>
int SumOfElements(int A[])
    int i, sum = 0;
    int size = sizeof(A)/sizeof(A[0]);
    for(i = 0; i < size; i++)
                                         C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\
                                         Sum of elements = 1
        sum+= A[i];
    return sum;
int main()
    int A[] = \{1,2,3,4,5\};
    int total = SumOfElements(A);
    printf("Sum of elements = %d\n",total);
```

```
#include<stdio.h>
int SumOfElements(int A[])
    int i, sum = 0;
    int size = sizeof(A)/sizeof(A[0]);
    printf("SOE - Size of A = %d, size of A[0] = %d", sizeof(A), sizeof(A[0]));
    for(i = 0:i< size:i++)</pre>
        sum+= A[i];
    return sum:
int main()
    int A[] = \{1,2,3,4,5\};
    int total = SumOfElements(A);
    printf("Sum of elements = "%d\n", total);
    printf("Main - Size of A = %d, size of A[0] = %d", sizeof(A), sizeof(A[0])):
```

```
#include<stdio.h>
int SumOfElements(int A[])
    int i, sum = 0;
    int size = sizeof(A)/sizeof(A[0]);
    printf("SOE - Size of A = %d, size of A[0] = %d\n", sizeof(A), sizeof(A[0]));
    for(i = 0;i< size;i++)</pre>
                                    C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\Debug\SampleApp2.exe
                                    SOE - Size of A = 4, size of A[0] = 4
        sum+= A[i];
                                    Sum of elements = 1
                                    Main - Size of A = 20, size of A[0] = 4
    return sum;
int main()
    int A[] = \{1,2,3,4,5\};
    int total = SumOfElements(A);
    printf("Sum of elements = %d\r" !
    printf("Main - Size of A = %d, size of A[0] = %d\n".sizeof(A).sizeof(A[0]):
```

Arrays as function arguments

```
#include<stdio.h>
int SumOfElements(int A[])
                                                                                 Applications
                                                        Stack
                                                                                   memory
    int i, sum = 0;
    int size = sizeof(A)/sizeof(A[0]);
    for(i = 0;i< size;i++)</pre>
                                                                                        Heap
        sum+= A[i];
    return sum;
int main()
                                                                                      Static/Global
    int A[] = \{1,2,3,4,5\};
                                                                                      Code (Text)
    int total = SumOfElements(A);
    printf("Sum of elements = %d\n".total);
```

Arrays as function arguments

```
#include<stdio.h>
int SumOfElements(int A[])
                                                                                Applications
                                                       Stack
                                                                                  memory
   int i, sum = 0;
   int size = sizeof(A)/sizeof(A[0]);
   for(i = 0;i< size;i++)</pre>
                                                                                       Heap
       sum+= A[i];
                                           SOE()
   return sum;
int main()
                                             Main()
                                                                                    Static/Global
   int A[] = \{1,2,3,4,5\};
                                                                                     Code (Text)
 wint total = SumOfElements(A);
   printf("Sum of elements = %d\n".total);
```

Arrays as function arguments

```
#include<stdio.h>
int SumOfElements(int A[])
                                                                                Applications
                                                       Stack
                                                                                  memory
   int i, sum = 0;
   int size = sizeof(A)/sizeof(A[0]);
   for(i = 0;i< size;i++)</pre>
                                                                                       Heap
       sum+= A[i];
                                           SOE()
   return sum;
int main()
                                            Mainl)
                                                                                    Static/Global
                                                          AV
   int A[] = \{1,2,3,4,5\};
                                                                                     Code (Text)
 wint total = SumOfElements(A);
   printf("Sum of elements = %d\n".total);
```

```
Arrays as function arguments
```

```
#include<stdio.h>
int SumOfElements(int A[]) (int + A)
                                                                               Applications
                                                      Stack
                                                                                 memory
    int i, sum = 0;
    int size = sizeof(A)/sizeof(A[0]);
    for(i = 0;i< size;i++)</pre>
                                                                                     Heap
        sum+= A[i];
                                           SOE()
    return sum;
                                                         A= 200
int main()
                                                       total
                                                                                   Static/Global
                                            Main()
                                                         AV
    int A[] = \{1,2,3,4,5\};
                                                                                   Code (Text)
 wint total = SumOfElements(A);
    printf("Sum of elements = %d\n".total);
```

```
#include<stdio.h>
int SumOfElements(int A[])
    int i, sum = 0;
    int size = sizeof(A)/sizeof(A[0]);
    printf("SOE - Size of A = %d, size of A[0] = %d\n", sizeof(A), sizeof(A[0]));
    for(i = 0:i< size:i++)</pre>
        sum+= A[i];
    return sum;
int main()
    int A[] = \{1,2,3,4,5\};
    int total = SumOfElements(A);
    printf("Sum of elements = %d\n",total);
    printf("Main - Size of A = %d, size of A[0] = %d\n", sizeof(A), sizeof(A[0]));
```

```
#include<stdio.h>
int SumOfElements(int* A)I
    int i, sum = 0;
    int size = sizeof(A)/sizeof(A[0]);
    printf("SOE - Size of A = %d, size of A[0] = %d\n", sizeof(A), sizeof(A[0]));
    for(i = 0;i< size;i++)</pre>
        sum+= A[i];
    return sum;
int main()
    int A[] = \{1,2,3,4,5\};
    int total = SumOfElements(A);
    printf("Sum of elements = %d\n",total);
    printf("Main - Size of A = %d, size of A[0] = %d\n", sizeof(A), sizeof(A[0]));
```

```
#include<stdio.h>
int SumOfElements(int* A, int size)// "int* A" or "int A[]" ..it's the same..
    int i, sum = 0;
    printf("SOE - Size of A = %d, size of A[0] = %d\n", sizeof(A), sizeof(A[0]));
    for(i = 0; i < size; i++)
        sum+= A[i]; // A[i] is *(A+i)
    return sum;
int main()
    int A[] = \{1,2,3,4,5\};
    int size = sizeof(A)/sizeof(A[0]);
```

printf("Main - Size of A = %d, size of A[0] = %d\n", sizeof(A), sizeof(A[0]));

int total = SumOfElements(A, size); // A can be used for &A[0]

printf("Sum of elements = %d\n",total);

1

```
#include<stdio.h>
void Double(int* A, int size)// "int* A" or "int A[]" ..it's the same..
    int i, sum = 0;
    for(i = 0;i< size;i++)
        A[i] = 2*A[i];
int main()
    int A[] = \{1,2,3,4,5\};
    int size = sizeof(A)/sizeof(A[0]);
    int i;
    Double(A, size);
```

for(i = 0;i< size;i++)</pre>

1

printf("%d ",A[i]);

```
#include<stdio.h>
void Double(int* A, int size)// "int* A" or "int A[]" ..it's the same..
                              int i, sum = 0;
                             for(i = 0;i< size;i++)</pre>
                                                          A[i] = 2*A[i];
                                                                                                                                                                                                                                                                                                                                      C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\SampleApp2\Debug\Sample
int main()
                              int A[] = \{1,2,3,4,5\};
                              int size = sizeof(A)/sizeof(A[0]);
                              int i;
                             Double(A, size);
                             for(i = 0;i< size;i++)
                                                           printf("%d ",A[i]);
```

1

Outline

- 1 Introduction to Pointers
- 2 Pointers and Arrays
- 3 Arrrays as Function Arguments
- Character Arrays and Pointers
- 5 Character Arrays and Pointers Part II
- 6 Pointers and Multi-Dimensional Arrays

String: - group of characters
eg: - "John"

"Hello World"

"I am feeling lucky"

.....

) How to store Strings

Size of array 7 no. of characters in string + 1

"John" Size >,5

) HOW to store strings

Size of array > no. of characters in string + 1

"John" Size >,5

) How to store strings

"John" +Size >,5 01234567 Char C[8]; C J O H N

C[0] = 'J'; C[1] = '0'; C[2] = 'H'; C[3] = 'N';

1) How to store strings

Size of array > no. of characters in string + 1 "John" Size >,5 01234567 Char C[8]; C J OHN 10 11 11

C[0] = 'J'; C[1] = '0'; C[2] = 'H'; C[3] = 'N'; C[4] = 101;

```
//character arrays and pointers
#include<stdio.h>
int main()
{
    char C[4];
    C[0] = 'J';
    C[1] = 'O';
```

C[2] = 'H';
C[3] = 'N'; I
printf("%s",C);

```
//character arrays and pointers
#include<stdio.h>
int main()
     char C[4];
                            C:\Users\animesh\documents\visual studio 2010\Projects
     C[0] = 'J';
     C[1] = '0';
     C[2] = 'H';
     C[3] = 'N';
     printf("%s",C);
```

```
//character arrays and pointers
#include<stdio.h>
int main()
                          C:\Users\animesh\documents\visual studio
    char C[5];
    C[0] = 'J';
    C[1] = '0';
    C[2] = 'H';
    C[3] = 'N';
    C[4] = ' \ 0';
    printf("%s",C);
```

```
//character arrays and pointers
#include<stdio.h>
int main()
     char C[20];
                            C:\Users\animesh\documents\visual studio 2010\Pro
    C[0] = 'J';
    C[1] = '0';
    C[2] = 'H';
    C[3] = 'N';
    C[4] = ' (0';
     printf("%s",C);
```

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
int main()
    char C[20];
    C[0] = 'J';
    C[1] = '0';
    C[2] = 'H';
    C[3] = 'N';
    C[4] = ' (0';
    int len = strlen(C);
    printf("Length = %d\n",len);
1
```

```
//character arrays and pointers
#include<stdio.h>
                                  C:\Users\animesh\documents\visual studio 2010\Proj
#include<string.h>
                                  .ength
int main()
     char C[20];
    C[0] = 'J';
    C[1] = '0';
    C[2] = 'H';
    C[3] = 'N';
    C[4] = ' \backslash 0';
     int len = strlen(C);
     printf("Length = %d\n",len);
1
```

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
int main()
{
    char C[20] = "JOHN";
    int len = strlen(C);
```

printf("Length = %d\n",len);

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
int main()
{
    char C[20];
    C = "JOHN"; I
```

printf("Length = %d\n",len);

int len = strlen(C);

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
int main()
    char C[] = "JOHN";
    printf("Size in bytes = %d\n", sizeof(C));
    int len = strlen(C);
```

printf("Length = %d\n",len);

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
int main()
     char C[] = "JOHN";
     printf("Size in bytes = %d\n", sizeof(C));
     int len = strlen(C);
                                           C:\Users\animesh\documents\visual studio 2010\Projects\SampleApp2\Debu
     printf("Length = %d\n",len);
                                           Size in bytes = 5
                                           Length = 4
```

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
int main()
     char C[4] = "JOHN";
      printf("Size Error: a value of type "const char [5]" cannot be used to initialize an entity of type "char [4]"
      int len = strlen(C);
     printf("Length = %d\n",len);
```

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
int main()
    char C[5] = "JOHN";
    printf("Size in bytes = %d\n", sizeof(C));
    int len = strlen(C);
    printf("Length = %d\n",len);
```

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
int main()
    char C[5] = {'J','O','H','N','\0'};
    printf("Size in bytes = %d\n", sizeof(C));
    int len = strlen(C);
    printf("Length = %d\n",len);
```

2) Arrays and pointers are different types that are used in Similar manner

2) Arrays and pointers are different types that are used in Similar manner

2) Arrays and pointers are different types that are used in Similar manner

Char C1[6] = "Hello"; C1 Hello 1000

```
2) Arrays and pointers are different types that are used in Similar manner

Char C1[6] = "Hello"; C1 Hello";

Char+ C2;

C2 = C1;

400 200
```

Print (2[1]; // ((2[0] = 'A'; // "Ael(0" (2[i] is *(c2+i)

```
2) Arrays and pointers are different types that
   are used in Similar manner
                                 200 201 202 . . .
 Char C1[6] = "Hello";
 Chart (2)
  C2 = C1;
                             . C2
  Print (2[1]; 1/ 1
  C2[0] = 'A'; // "Aello"
```

C2[i] is *(c2+i)C1[i] or *(c1+i)

```
2) Arrays and pointers are different types that
   are used in Similar manner
                                200 201 202 . . .
 Char C1[6] = "Hello;
 chart (2)
  C2 = C1; V
  Print (2[1]; // (
                              C1 = C2; X
  c2[0] = 'A'; // "Aello"
```

c2[i] is *(c2+i)

c1[i] or *(c1+i)

C1 = C1+1; X

C2++;

3) Arrays are always passed to function by reference

```
//character arrays and pointers
#include<stdio.h>
void print(char* C)
{
    int i = 0;
    while(C[i] != '\0')
        printf("%c",C[i]);
        i++;
    printf("\n");
int main()
```

char C[20] = "Hello";

print(C);

}

```
//character arrays and pointers
#include<stdio.h>
void print(char* C)
    int i = 0;
    while(C[i] != '\0')
        printf("%c",C[i]);
        i++;
    printf("\n");
int main()
    char C[20] = "Hello";
    print(C);
```



```
//character arrays and pointers
#include<stdio.h>
void print(char* C)
{
    int i = 0;
    while(*(C+i)!= '\0')
        printf("%c",C[i]);
        i++;
    printf("\n");
int main()
```

char C[20] = "Hello";

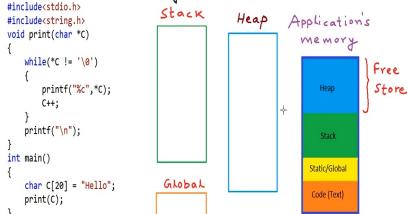
print(C);

```
//character arrays and pointers
#include<stdio.h>
void print(char* C)
    while(*C != '\0')
                                  C:\Users\ar
        printf("%c",*C);
        C++;
    printf("\n");
int main()
    char C[20] = "Hello";
    print(C);
```

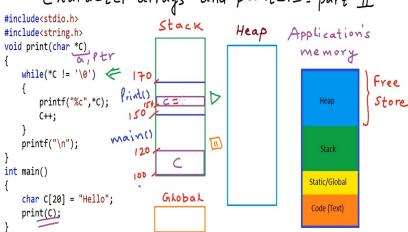
Outline

- 1 Introduction to Pointers
- 2 Pointers and Arrays
- 3 Arrrays as Function Arguments
- Character Arrays and Pointers
- 5 Character Arrays and Pointers Part II
- 6 Pointers and Multi-Dimensional Arrays

Character arrays and pointers - part I Stack Heap Application's



Character arrays and pointers - part I



Character arrays and pointers - part I

```
#include<stdio.h>
                               Stack
#include<string.h>
                                                100 101 102 103 104 105 $
void print(char *C)
   while(*C != '\0')
                          170
                        (inte)
                        15054
      C++;
                                                    100
                     main()
                                          1
                                                     154
   printf("\n");
                           120
int main()
                          100
                                Global
   char C[20] = "Hello";
   print(C);
```

Character arrays and pointers - part II

```
#include<stdio.h>
                              Stack
#include<string.h>
                                              100 101 102 103 104 105 3
void print(char *C)
   while(*C != '\0')
      printf("%c",*C);
                        150
      C++;
                                                 105
                    main()
   printf("\n");
                         120
int main()
                                               output: Hello
                         100
                              Global
   char C[20] = "Hello";
   print(C);
```

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
void print(char *C)
{
    while(*C != '\0')
        printf("%c",*C);
        C++;
    printf("\n");
}
int main()
    char *C = "Hello";
    print(C);
```

}

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
void print(char *C)
    while(*C != '\0')
                                  C:\Users\animesh\documents\visual stuc
         printf("%c",*C);
         C++;
    printf("\n");
int main()
    char *C = "Hello":
    print(C);
```

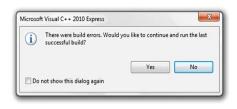
```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
void print(char *C)
   while(*C != '\0')
       printf("%c",*C);
       C++;
   printf("\n");
int main()
   //char C[20] = "Hello"; // string gets stored in the space for array
    char *C = "Hello"; // string gets stored as compile time costant
   printf("Hello World");
   print(C);
```

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
/oid print(char *C)
    while(*C != '\0')
                                     Microsoft Visual C++ 2010 Express
                                          Unhandled exception at 0x00dd2458 in SampleApp2.exe: 0xC0000005: Access
         printf("%c",*C);
                                         violation writing location 0x00dd7884.
         C++:
    printf("\n");
int main()
                                                       Break
                                                                  Continue
                                                                              Ignore
    //char C[20] = "Hello"; // string gets stored in the space for array
    char *C = "Hello"; // string gets stored as compile time costant
    C[0] = 'A';
    printf("Hello World");
    print(C);
```

```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
void print(char *C)
{
    C[0] = 'A';
    while(*C != '\0')
        printf("%c",*C);
        C++;
    printf("\n");
int main()
    char C[20] = "Hello";
    print(C);
}
```

```
C:\Users\animesh\documents\visual studio
```

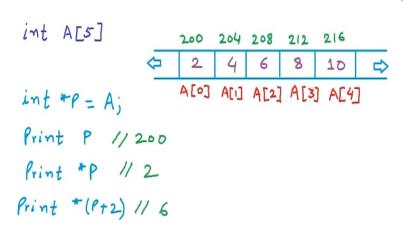
```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
void print(const char *C)
   C[0] = 'A';
    while(*C != '\0')
        printf("%c",*C);
        C++;
    printf("\n");
int main()
    char C[20] = "Hello";
    print(C);
```



```
//character arrays and pointers
#include<stdio.h>
#include<string.h>
void print(const char *C)
    while(*C != '\0')
                                   C:\Users\animesh\documents\visual studio 2010
                                 Hello
         printf("%c",*C);
         C++;
    printf("\n");
int main()
    char C[20] = "Hello";
    print(C);
```

Outline

- 1 Introduction to Pointers
- 2 Pointers and Arrays
- 3 Arrrays as Function Arguments
- Character Arrays and Pointers
- 5 Character Arrays and Pointers Part II
- 6 Pointers and Multi-Dimensional Arrays



int A[5]
$$200\ 204\ 208\ 212\ 216$$
 $= 2\ 4\ 6\ 8\ 10\ = 5$

int *P = A;

Print A // 200

Print *A // 2

Print *(A+2) // 6 *(A+i) is Same as A[i]

(A+i) is Same as PA[i]

int
$$A[5]$$

200 204 208 212 216

2 2 4 6 8 10 D

int *P = A;

Print A // 200

Print *A // 2

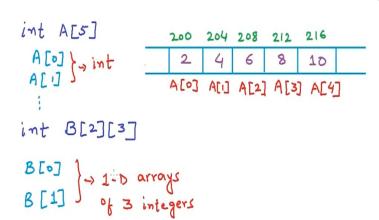
Print *(A+2) // 6

P = A;

A = P; X

Print A = P; X

mycc



int B[2][37 400 404 408 412 416 420 B[1]] - 1-D'arrays = 2.

B[1] of 3 integers B[1] BLOT B[17 int (*p)[3] = B; Print B or &B[0] 1/400 Print *B or B[0] or 4B[0][0] // 400 Print 8+1 //

int B[2][37 400 404 408 412 416 420 B[1]] -> 1-D arrays
B[1] of 3 integers B[a][a] B[1] int (*P)[3] = B; Print B or &B[0] 1/400 Print *B or B[0] or 4B[0][0] // 400 Print 8+1 // 400+12 = 412 &B[17

```
int B[2][37
                          400 404 408 412 416 420
B[1] | 1-D arrays of 3 integers
                     (2 3 B[0]
                                        B/17
  int (+p)[3] = B;
  Print B or &B[0] 1/400
  Print *B or B[0] or 4B[0][0] // 400
  Print 8+1 or 4B[1] // 412
  Print *(8+1) or B[1] or &B[1][0] 11412
```

int B[2][37 400 404 408 412 416 420 B[o] $\rightarrow 1-D$ arrays B[1] $\rightarrow 6$ 3 integers B[0][0] 8[1][0] B[O] B[1] int (*P)[3] = B; Print B or &B[0] // 400 Print *B or B[0] or 4B[0][0] // 400 Print 8+1 or 4B[1] // 412 Print *(8+1) or B[1] or &B[1][0] 11412 >> returning int* Print *(B+1),+2 or B[1]+2 or 4B[1][2] 1/ 420

```
int B[2][37
B[1] ) -> 1-D arrays
B[1] of 3 integers
                                   404 408 412 416 4
                              BT07507
                                           8[1][0]
int (*P)[3] = B;
                                  B[D]
                                                B[1]
Print B or &B[0] 1/400
Print *B or B[0] or 4B[0][0] // 400
Print B+1 or &B[1] // 412
Print *(8+1) or B[1] or 48[1][0] 11412
Print *(B+1)+2 or B[1]+2 or 4B[1][2] 1/ 420
Print * (*B+1) B \rightarrow int (*)[3] B[0] \rightarrow int *
```

int B[2][37 B[1] \rightarrow 1-D arrays
B[1] of 3 integers 400 404 408 412 416 4 B[1][0] int (+p)[3] = B; BLOJ B/17 Print B or &B[0] 1/400 Print *B or B[0] or 4B[0][0] // 400 Print B+1 or &B[1] // 412 Print *(8+1) or B[1] or 4B[1][0] 11412 Print *(B+1)+2 or B[1]+2 or 48[1][2] 1/ 420 Print * (*B+1) B -> int (*)[3] B[0] -> int +

&BEOTE 17

int B[2][37 400 404 408 412 416 4 B[1]) -> 1-D arrays
B[1] of 3 integers 8[1][0] int (+p)[3] = B; B[0] B[1] Print B or &B[0] 1/400 Print *B or B[0] or 4B[0][0] // 400 Print B+1 or &B[1] // 412 Print *(8+1) or B[1] or 4B[1][0] 11412 Print *(B+1)+2 or B[1]+2 or 48[1][2] 1/ 420 Print * (+B+1) B[O][1]

int B[2][37 400 404 408 412 416 4 B[1]) -> 1-D arrays
B[1] of 3 integers 8[1][0] int (+p)[3] = B; B[0] B[1] Print B or &B[0] 1/400 Print *B or B[0] or 4B[0][0] // 400 Print B+1 or &B[1] // 412 Print *(8+1) or B[1] or 4B[1][0] 11412 Print *(B+1)+2 or B[1]+2 or 48[1][2] 1/ 420 Print * (+B+1) B[O][1]

int
$$B[2][3]$$

400 404 408 412 416 420

For 2-D array

 $B[3][3] = {B[1][2]}$
 $B[3][3] = {B[1][2]}$
 $B[3][3] = {B[1][2]}$

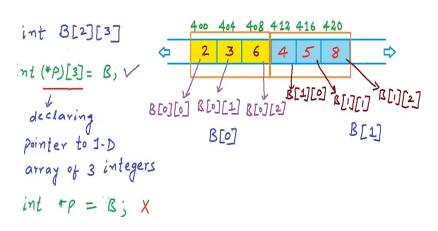
int B[2][3]

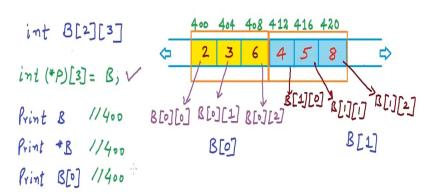
\$\frac{400 404 408 412 416 420}{2 3 6 4 5 8} \Rightarrow

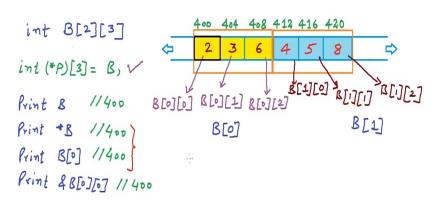
B[0][0] R[0][1] R[0][2]

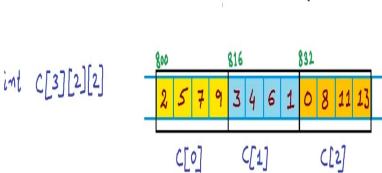
B[0]

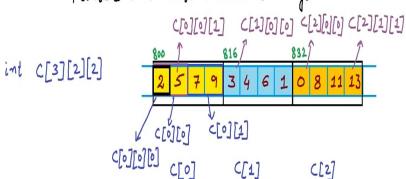
B[0]

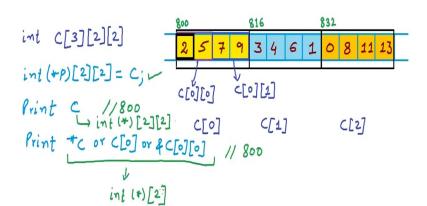












Pointers and multi-dimensional arrays

int
$$C[3][2][2]$$

int $C[3][2][2]$

2 5 7 9 3 4 6 1 0 8 11 13

int $(*P)[2][2] = C_j$

C[0][0] $C[0][1]$

Print C

int $(*)[2][2]$

C[0] $C[1]$

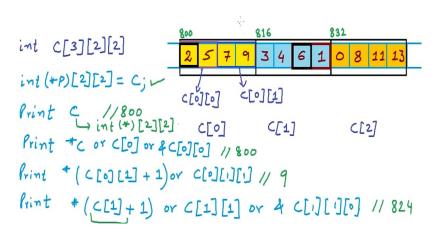
C[1]

C[1][j][k] = * (C[i][j]+k) = *(*(c[i]+j)+k)

= * (*(c[i]+j)+k)

Pointers and multi-dimensional arrays

Pointers and multi-dimensional arrays



```
Pointers and multi- dimensional arrays
#include<stdio.h>
int main()
     int C[3][2][2]=\{\{\{2,5\},\{7,9\}\},
                           \{\{3,4\},\{6,1\}\},
                           {{0,8},{11,13}}};
     printf("%d %d %d %d", C, *C, C[0], &C[0][0]);
                 C:\Users\animesh\Documents\Visual Studio 2010\Projects\SampleApp5\Debug\SampleApp5.exe
                 4192172 4192172 4192172 4192172_
```

```
Pointers and multi- dimensional arrays
#include<stdio.h>
int main()
    int C[3][2][2]=\{\{\{2,5\},\{7,9\}\},
                     \{\{3,4\},\{6,1\}\},
                      {{0,8},{11,13}}};
    printf("%d %d %d %d", C, *C, C[0], &C[0][0]);
    printf("%d",*(C[0][0]+1));
```

```
Pointers and multi- dimensional arrays
#include<stdio.h>
int main()
    int C[3][2][2]={{{2,5},{7,9}}
                                           3078752 3078752 3078752
                     \{\{3,4\},\{6,1\}\}
                     {{0,8},{11,13
    printf("%d %d %d %d\n", C, *C
    printf("%d\n",*(C[0][0]+1));
```

```
// Pointers and multi- dimensional arrays
#include<stdio.h>
void Func(int *A) // Argument: 1-D array of integers
int main()
    int C[3][2][2]=\{\{\{2,5\},\{7,9\}\},
                      \{\{3,4\},\{6,1\}\},
                      {{0,8},{11,13}}};
    int A[2] = \{1,2\};
```

int $B[2][3] = \{\{2,4,6\},\{5,7,8\}\};$

Func(A);

```
// Pointers and multi- dimensional arrays
#include<stdio.h>
void Func(int (*A)[3]) // Argument: 2-D array of integers
int main()
    int C[3][2][2]=\{\{\{2,5\},\{7,9\}\},
                      \{\{3,4\},\{6,1\}\},
                      {{0,8},{11,13}}};
    int A[2] = \{1,2\};
    int B[2][3] = \{\{2,4,6\},\{5,7,8\}\}; // B \text{ returns int } (*)[3]
    Func(A);
                                                                   myce
```

```
// Pointers and multi- dimensional arrays
#include<stdio.h>
void Func(int A[][3],) // Argument: 2-D array of integers
int main()
    int C[3][2][2]=\{\{\{2,5\},\{7,9\}\},
                      \{\{3,4\},\{6,1\}\},
                      {{0,8},{11,13}}};
    int A[2] = \{1,2\};
    int B[2][3] = \{\{2,4,6\},\{5,7,8\}\}; // B \text{ returns int } (*)[3]
    Func(A);
```

```
Pointers and multi- dimensional arrays
#include<stdio.h>
void Func(int A[][3]) // Argument: 2-D array of integers
int main()
    int C[3][2][2]=\{\{\{2,5\},\{7,9\}\},
                      \{\{3,4\},\{6,1\}\},
                      \{\{0,8\},\{11,13\}\}\};
    int A[2] = \{1, 2\};
    int B[2][3] = \{\{2,4,6\},\{5,7,8\}\}; // B returns int (*)[3]
    Func(B);
```

```
// Pointers and multi- dimensional arrays
#include<stdio.h>
void Func(int A[][3]) // Argument: 2-D array of integers
int main()
    int C[3][2][2]=\{\{\{2,5\},\{7,9\}\},
                      \{\{3,4\},\{6,1\}\},
                      {{0,8},{11,13}}};
    int A[2] = \{1,2\};
    int B[2][3] = \{\{2,4,6\},\{5,7,8\}\}; // B returns int (*)[3]
    int X[2][4];
```

Func(X); // X returns int (*)[4]

```
// Pointers and multi- dimensional arrays
#include<stdio.h>
void Func(int (*A)[2][2]) // Argument: 2-D array of integer
int main()
    int C[3][2][2]=\{\{\{2,5\},\{7,9\}\},
                     \{\{3,4\},\{6,1\}\},
                     {{0,8},{11,13}}};
    int A[2] = \{1,2\};
    int B[2][3] ={{2,4,6},{5,7,8}}; // B returns int (*)[3]
    int X[5][3];
```

Func(C);