Introduction to Arrays

Today

- Introduction to arrays
  - Syntax
  - Basic I/O
  - Simple usage

Why Use Arrays

Example : Maximum of 3 numbers

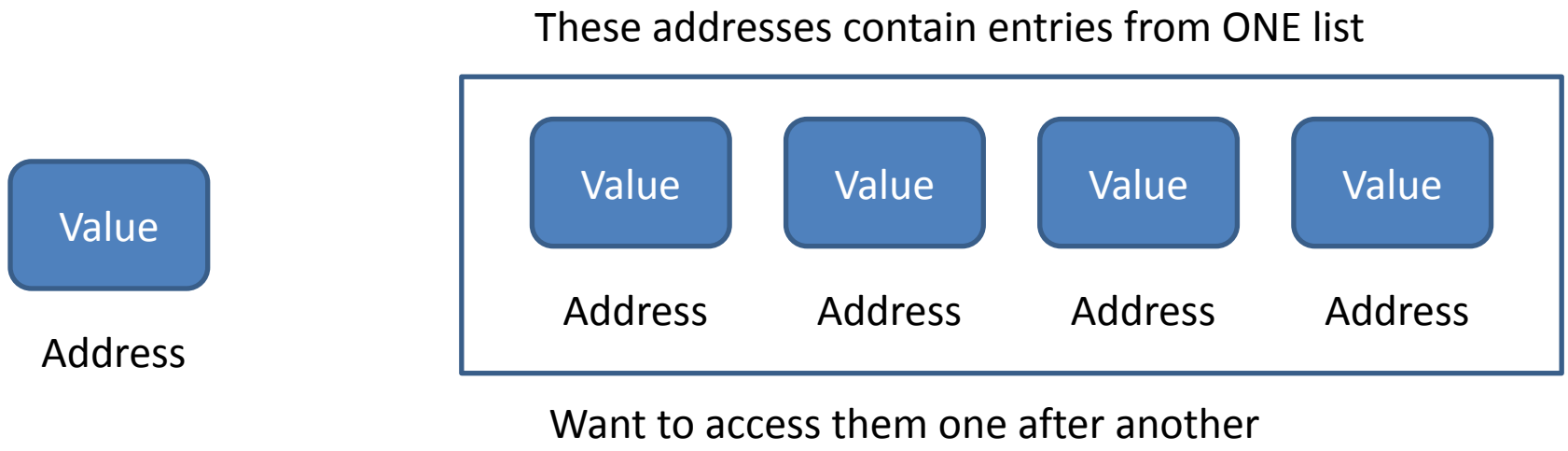
```
int main(){
   int a, b, c, m;

   /* code to read
    * a, b, c */

   if (a>b){
      if (a>c) m = a;
      else m = c;
   }
   else{
      if (b>c) m = b;
      else m = c;
   }

   /* print or use m */

   return 0;
}
```

```
int max(int a, int b){
   if (a>b)
     return a;
   else
     return b;
}

int main() {
   int a, b, c, m;

   /* code to read
    * a, b, c */

   m = max(a, b);
   m = max(m, c);
   /* print or use m */

   return 0;
}
```

How would you scale this code to handle large number of inputs (e.g.: max of 100 numbers!)

Operations on a List

- Take this list
- Do something to every element on this list
- Return the output for every element on the list



Arrays in C

An array in C is defined similar to defining a variable.

```
int a[5];
```

The square parenthesis [5] indicates that a is not a single integer but an array, that is a *consecutively allocated* group, of 5 integers.

It creates five integer boxes or variables

a[0] a[1] a[2] a[3] a[4]

Array elements are consecutively allocated in memory.

The boxes are addressed as a[0], a[1], a[2], a[3] and a[4]. These are called the **elements** of the array.

Max of N Numbers

```
int max(int a, int b){
   if (a>b)
     return a;
   else
     return b;
}

int main() {
   int a[100];

   /* code to read
    100 numbers/
m = max(a[0],a[1]);
for (i = 2; i<100;i++){
   m = max(a[i], m);
/* print or use m */

   return 0;
}
```

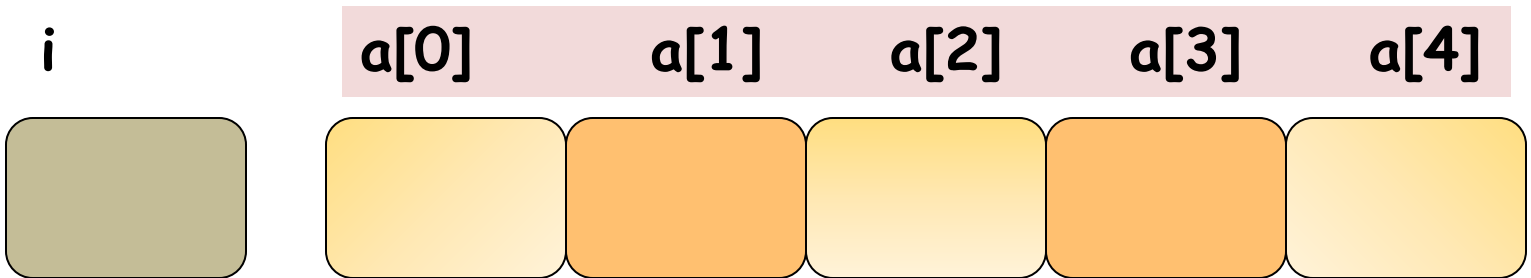
Arrays and loops are the bread and butter of basic programming

```
#include <stdio.h>
int main () {
      int i;
      int a[5];

       for (i=0; i < 5; i= i+1) {
         a[i] = i+1;

         printf("%d", a[i] );
       }
       return 0;
}
```

The program defines an integer variable called i and an integer array with name a of size 5

This is the notation used to address the elements of the array.

- The variable i is being used as an "index" for a.
- Similar to the math notation a_i



```
#include <stdio.h>
int main () {
      int a[5];
      int i;

      for (i=0; i < 5; i= i+1) {
         a[i] = i+1;
      }
      return 0; }
```

Let us trace the execution of the program.

i 5

a[0]	a[1]	a[2]	a[3]	a[4]
1	2	3	4	5

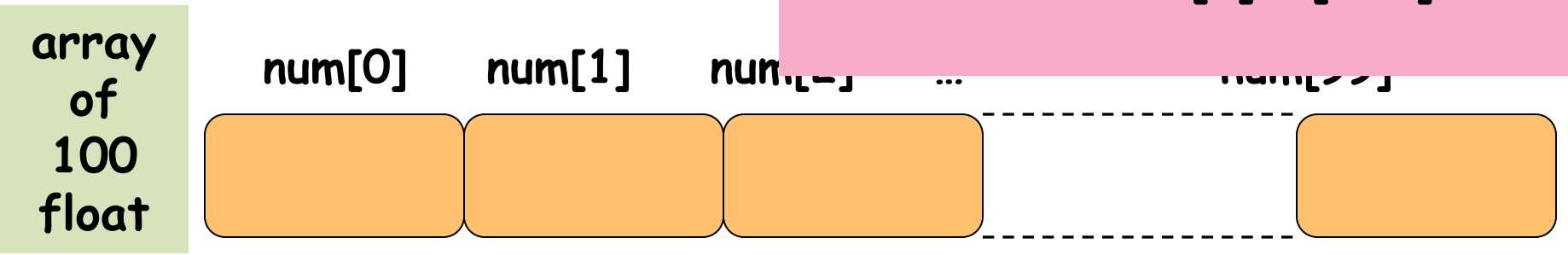
Statement becomes a[0] =0+1;
Statement becomes a[1] =1+1;
Statement becomes a[2] =2+1;
Statement becomes a[3] = 3+1;
Statement becomes a[4] = 4+1;

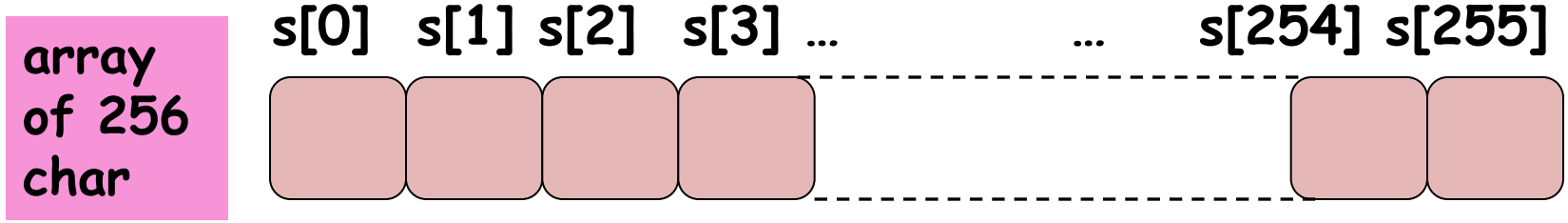
One can define an array of float or an array of char, or array of any data type of C. For example

```
int main() {
        float num[100];

        char s[256];

 /* some code here */
}
```

This defines an array called num of 100 floating point numbers indexed from 0 to 99 and named num[0]… num[99]

This defines an array called s of 256 characters indexed from 0 to 255 and named s[0]…s[255].

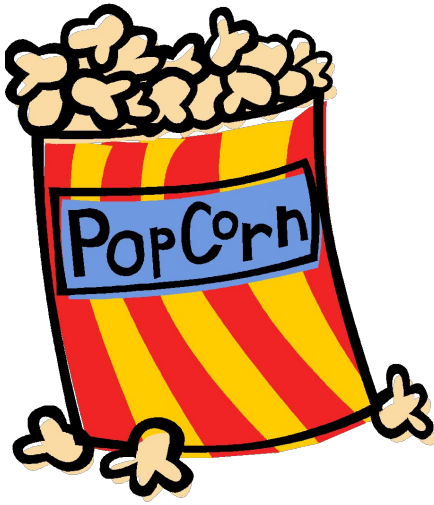




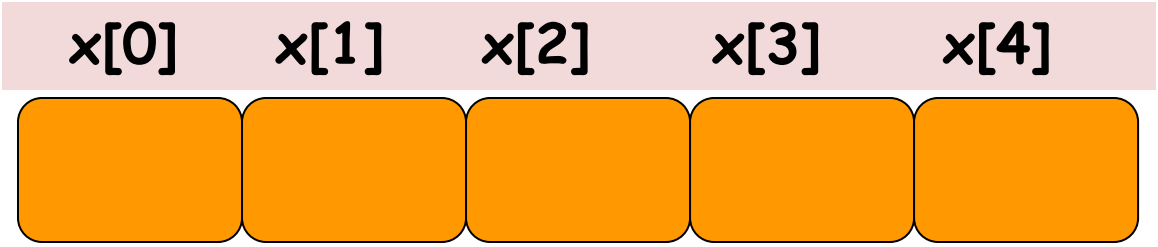
Mind the Size (of Array)

```
int f() {
    int x[5];
    …
}
```

This defines an integer array named x of size 5.



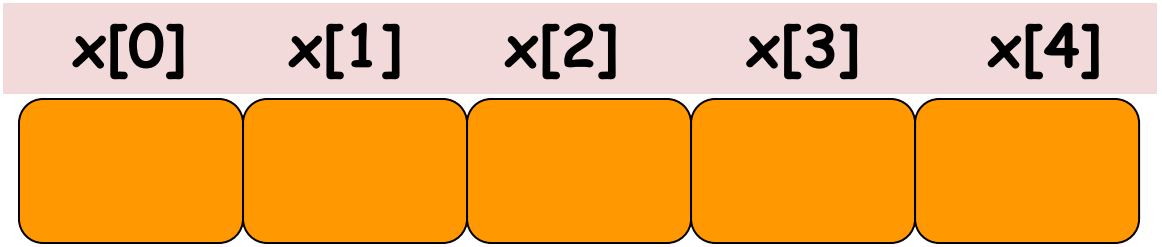
Five integer variables named x[0] x[1] … x[4] are allocated.



The variables x[0],x[1] … x[4] are integers, and can be assigned and operated upon like integers! OK, so far so good!

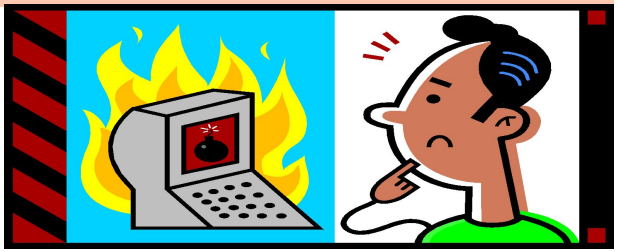
Mind the Size (of Array)

```
int f() {
    int x[5];
    …
}
```



But what about x[5], x[6], … x[55]?
Can I assign to x[5], increment it, etc.?

NO! Program may crash!



Why?

x[5], x[6], and so on are undefined. These are names but no storage has been allocated. Shouldn't access them!

Q: Shouldn't I or couldn't I access array elements outside of the array range declared?


```
int f() {
      int x[5];
      x[0] =0;
      x[1] =1;
        …
      x[4] =4;

      x[5] = 5;
      x[6] = 6;
}
```

All good

not recommended.

Will it compile? Yes, it will compile. C compiler may give a warning.

Program may give:
1) "segmentation fault: core dumped"
2) it may run correctly

Ans: You can but shouldn't. Program may crash.

Reading Directly into Array

Read N numbers from user directly into an array

```
#include <stdio.h>
int main() {
    int num[10];
    for (i=0; i<10; i=i+1) {
        scanf("%d", &num[i]);
   }
   return 0;
}
```

scanf can be used directly, treat an array element like variable of the same data type.

1. For integers, read as scanf("%d", &num[i]);
2. For reading elements of a char array s[], use scanf("%c", &s[j]).

In the previous slide, we had the statement:

```
scanf("%d", &num[i]      );
```

What does &num[i] mean?

&num[i]: two operators & and [].
gives address of array element num[i].

& is the "address-of" operator.

1. It can be applied to any defined variable.
2. It returns the location (i.e., address) of this variable in the program's memory.

[ ]
array indexing operator
e.g, num[i].

&num[i] is evaluated as:

&(num[i])

NOT as : (&num)[i]

Array Example: Print Backwards

Problem:

1. Define a character array of size 100 (upper limit)
2. read the input character by character and store in the array until either
   - 100 characters are read or
   - EOF (End Of File) is encountered
3. Now print the characters backwards from the array.

Example Input 1

Me or Moo

Output 1

ooM roa eM

Example Input 2

Eena Meena Dika

Output 2

akiD aneeM aneE

EOF (end of file)

- EOF is a distinctive value that is a non-character. This is to distinguish “an input” from “no more input”.
- In integers, it’s -1. Not to be confused with “\n” or “ ” (space).
- stdio.h contains the integer value of EOF.

Read and Print in Reverse

1. We will design the program in a top down fashion, using just main() function.
2. There will be two parts to main: read_into_array and print_reverse.
3. read_into_array will read the input character-by-character up to 100 characters or until the end of input.
4. print_reverse will print the characters in reverse.

Overall design

```
int main() {
        char s[100]; /* to hold the input */
        /* read_into_array */
        /* print_reverse */
        return 0;
}
```

Let us design the program fragment read_into_array.

Keep the following variables:

1. int count: count the number of characters read so far.
2. int ch: to read the next character using getchar().

Note that getchar() has prototype int getchar()
since getchar() returns all the 256 characters and the integer EOF

```
int ch;
int count = 0;
read the next character into ch using getchar();
while (ch is not EOF AND count < 100) {
    s[count] = ch;
        count = count + 1;
        read the next character into ch using getchar();
}
```

An initial design
(pseudo-code)

```
int ch;
int count = 0;
read the next character into ch using getchar();
while (ch is not EOF AND count < 100) {
    s[count] = ch;
        count = count + 1;
        read the next character into ch using getchar();
}
```

initial design pseudo-code

```
int ch;
int count = 0;
ch = getchar();
while ( ch != EOF && count < 100) {
    s[count] = ch;
        count = count + 1;
        ch = getchar();
}
```

Overall design

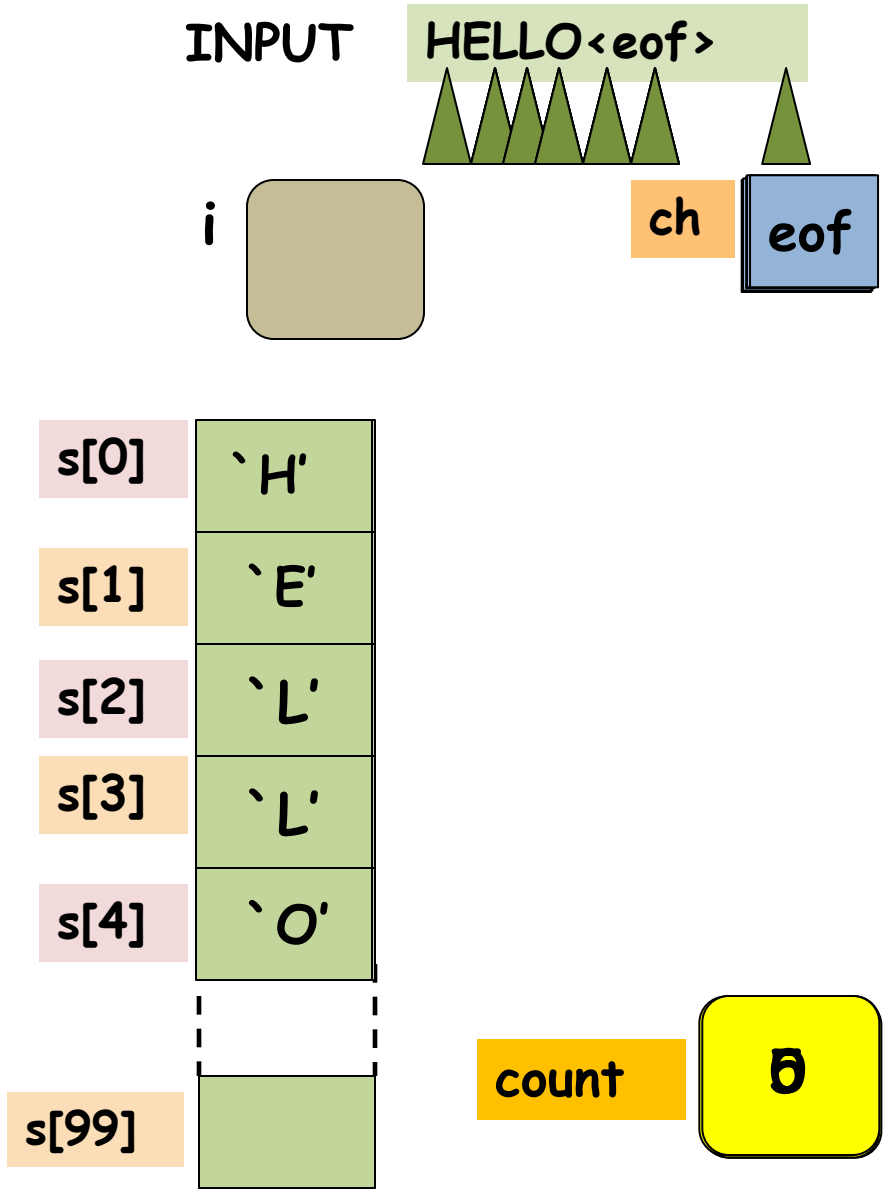
```
int main() {
    char s[100];
 /* read_into_array */
 /* print_reverse */
    return 0;
}
```

Translating the read_into_array pseudo-code into code.

What is the value of count at the end of read_into_array?

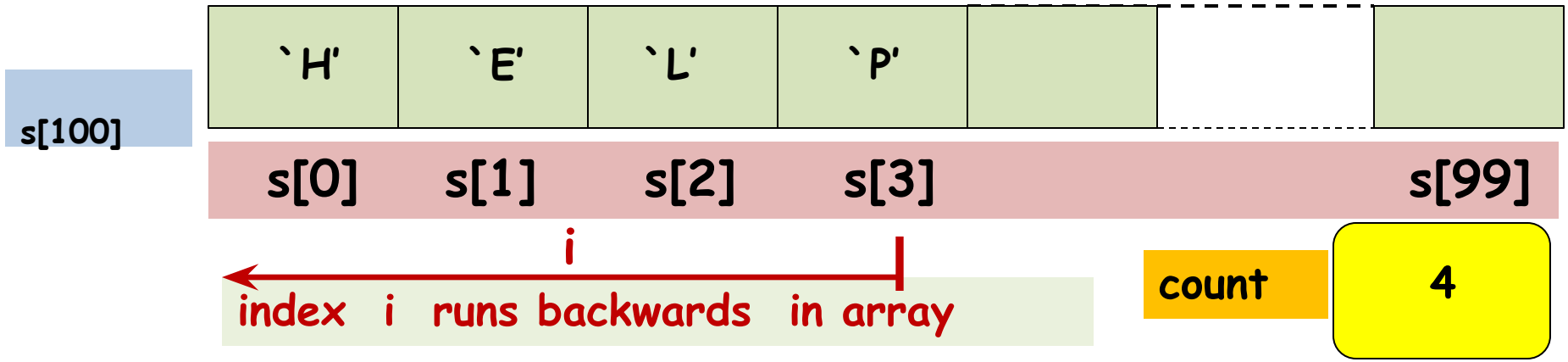
Let us trace the execution.
We will do this for part
read_into_array

```
#include <stdio.h>
int main() {
    char s[100];
    int count = 0;
    int ch, i;
    ch = getchar();

    while ( ch != EOF &&
            count < 100) {
        s[count] = ch;
        count = count + 1;
        ch = getchar();
    }
    ...
```



Now let us design the code fragment print_reverse

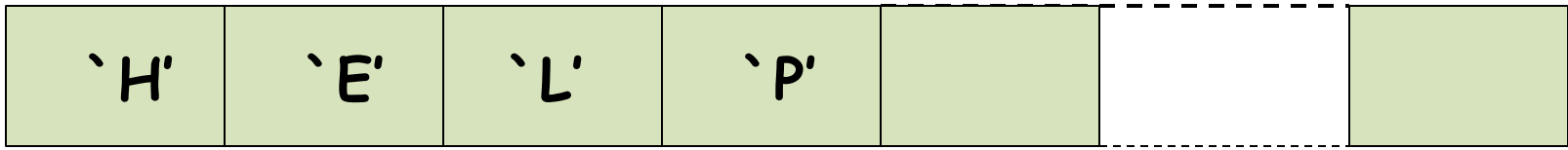
Suppose input is: HELP<eof>




```
int i;
set i to the index of  last  character read.

while (i >= 0) {
    print s[i]
    i = i-1;      /* shift array index one to  left */
}
```

PSEUDO CODE

The array char s[100]



count 4

```
int i;
set i to index of the last character read.

while (i >= 0) {
    print s[i]
    i = i-1;
}
```

PSEUDO CODE

```
int i;
i = count-1;

while (i >=0) {
        putchar(s[i]);
        i=i-1;
}
```

Translating pseudo code to C code: print_reverse

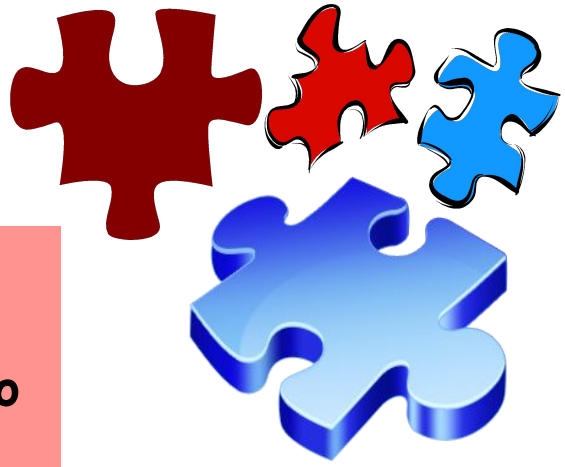
Code for printing characters read in array in reverse

Putting it Together

Overall design

```
int main() {
        char s[100];
        /* read_into_array */
        /* print_reverse */
        return 0;
}
```

The code fragments we have written so far.



read_into_array code.

```
int count = 0;
int ch;
ch = getchar();
while ( ch != EOF && count < 100) {
     s[count] = ch;
        count = count + 1;
        ch = getchar();
}
```

print_reverse code

```
int i;
i = count-1;
while (i >=0) {
        putchar(s[i]);
        i=i-1;
}
```

```
#include <stdio.h>
int main() {
    char s[100];
    int count = 0;
    int ch;
    int i;
```

```
/* the array of 100 char */
/* counts number of input chars read */
/* current character read  */
/* index for printing array backwards */
```

```
ch = getchar();
while ( ch != EOF && count < 100) {
    s[count] = ch;
        count = count + 1;
        ch = getchar();
}
```

```
/*read_into_array */
```

```
i = count-1;
while (i >=0) {
        putchar(s[i]);
        i=i-1;
}
```

```
/*print_in_reverse */
```

```
    return 0;
}
```

```
#include <stdio.h>
int main() {
      char s[100];
      int count = 0;
      int ch;
      int i;
```

/*read_into_array */

```
      while ( (ch=getchar()) != EOF &&
                  count < 100 )
      {   s[count] = ch;
            count = count + 1;
      }
```

```
      i = count-1;
      while (i >=0) {
            putchar(s[i]);
            i=i-1;
      }
```

/*print_in_reverse */

```
      return 0;
}
```

Practice Problem

Write a program to read in an array of 5 integers. Compute and print the running total of the integers.

Input: 3 1 5 2 9

Output: 3 4 9 11 20

Solution for Practice Problem

```
#include <stdio.h>
int main()
{
   int arr[5];
   //read input
   for(int i=0; i<5; i++)
      --------------;

   //compute running total
   for(int i=1; i<5; i++)
      ----------------;

   //obtain output
   for(int i=0; i<5; i++)
      printf("%3d",arr[i]);

   printf("\n");
   return 0;

}
```

Solution for Practice Problem

```
#include <stdio.h>
int main()
{
   int arr[5];
   //read input
   for(int i=0; i<5; i++)
      scanf("%d",&arr[i]);

   //compute running total
   for(int i=1; i<5; i++)
      ----------------;

   //obtain output
   for(int i=0; i<5; i++)
      printf("%3d",arr[i]);

   printf("\n");
   return 0;

}
```

Solution for Practice Problem

```
#include <stdio.h>
int main()
{
   int arr[5];
   //read input
   for(int i=0; i<5; i++)
      scanf("%d",&arr[i]);

   //compute running total
   for(int i=1; i<5; i++)
      arr[i] = arr[i-1]+arr[i];

   //obtain output
   for(int i=0; i<5; i++)
      printf("%3d",arr[i]);

   printf("\n");
   return 0;

}
```

Next Class

- Parameter passing using arrays