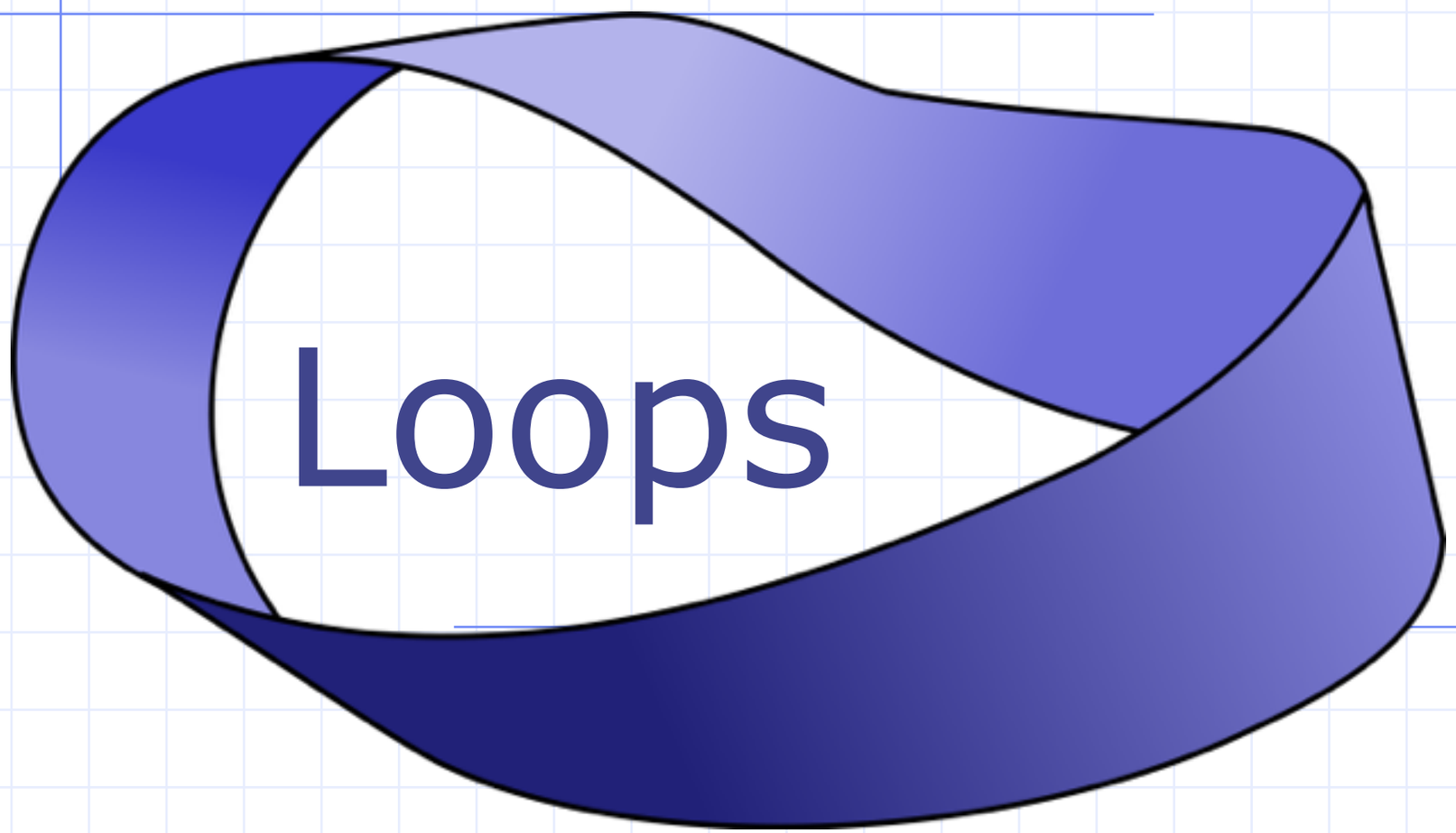
ESC101: Introduction to Computing



Example: Geometric Progression

- Given positive real numbers r and a, and a positive integer, n, the nth term of the geometric progression with a as the first term and r as the common ratio is ar^{n-1}.
- Write a program that given r, a, and n, displays the first n terms of the corresponding geometric progression.

```
#include<stdio.h>
int main(){
    int n, i;     float r, a, term;

    // Reading inputs from the user
    scanf("%f", &r);
    scanf("%f", &a);
    scanf("%d", &n);
    term = a;
    for (i=1; i<=n; i=i+1) {
        printf("%f\n", term); // Displaying i^th term
        term = term * r;      // Computing (i + 1)^th term
    }
    return 0;
}
```

Nested Loops

- Loop with in a loop
- Many iterations of inner loop ⇒ One iteration of outer loop

An Example

```
#include<stdio.h>
int main(){
    int i, j;
    int nrows=2, ncols=2;
    for (i=0; i<nrows; i=i+1) {
        for (j=0; j<ncols; j=j+1) {
            printf("%d  ",i*ncols+j);
        }
        printf("\n"); // Move to the next line
    }
  return 0;
}
```

Output

```
0  1
2  3
```

An Example

```
#include<stdio.h>
int main(){
    int i, j;
    int nrows=2, ncols=2;
    for (i=0; i<nrows; i=i+1) {
        for (j=1; j<=ncols; j=j+1) {
            printf("%d  ",i*ncols+j);
        }
        printf("\n"); // Move to the next line
    }
  return 0;
}
```

Output

```
1  2
3  4
```

Example

- Write a program that displays the following pattern

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25
6	12	18	24	30
7	14	21	28	35
8	16	24	32	40

integers are printed in 5 columns each

```
#include<stdio.h>
int main(){
    int i, j;

    for (i=1; i<=8; i=i+1) {
        for (j=1; j<=5; j=j+1) {

            ____________________

        }
        printf("\n"); // Move to the next line
    }

    return 0;
}
```

```
#include<stdio.h>
int main(){
    int i, j;

    for (i=1; i<=8; i=i+1) {
        for (j=1; j<=5; j=j+1) {
            printf("%4d", i*j); // Displaying i, 2i, …, 5i
        }
        printf("\n"); // Move to the next line
    }

    return 0;
}
```

Practice Problem

- Write a program to use for loop to take input a 3x3 matrix and output its trace

- Trace of a matrix is the sum of its diagonal elements

Write a program that computes the trace of a matrix using for loop

```
#include <stdio.h>
int main()
{
   int num, trace=0;;
   int i,j;
   for( i=0; i<3; i=i+1)
      for( j=0; j<3; j=j+1)
      {
         scanf("%d",&num);

      }
  return 0;
}
```

Write a program that computes the trace of a matrix using for loop

```
#include <stdio.h>
int main()
{
   int num, trace=0;;
   int i,j;
   for( i=0; i<3; i=i+1)
      for( j=0; j<3; j=j+1)
      {
         scanf("%d",&num);
         if(i == j)
            trace = trace+num;
      }
   printf("trace = %d\n",trace);
   return 0;
}
```

Displaying a pattern

```
#include <stdio.h>
int main(){
  int i,j;
  for (i=1; i<=5; i=i+1){
    for (j=i; j<2*i; j=j+1){
      printf("%d ",j);
    }
    printf("\n");
  }
  return 0;
}
```

- Output

```
1
2 3
3 4 5
4 5 6 7
5 6 7 8 9
```

increment/decrement operator

- Two very common actions in C

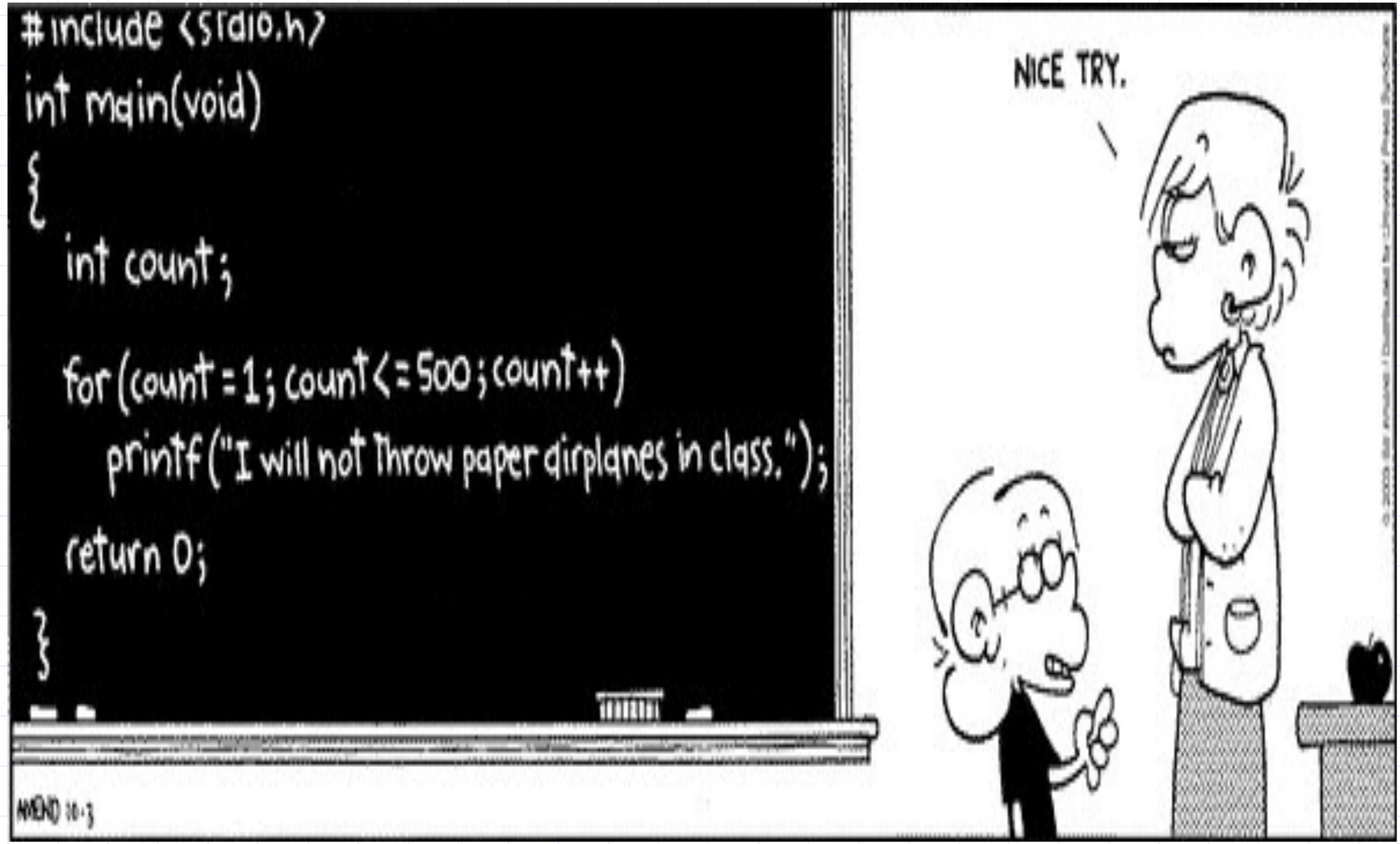
```
i = i + 1;
i = i – 1;
```

- These can be written in short as:

```
i++     // increment
i--     // decrement
```

- Complete semantics are bit involved
  - Not covered in this course
  - Advise: Do not use them other than:
    - in update_expr of for/while loops
    - Standalone statements: i++;

When not advisable to avoid repetitive job



For loop in C

- General form

```
for (init_expr; test_expr;
update_expr)
    statement;
```

- init_expr is the initialization expression.
- update_expr is the update expression.
- test_expr is the expression that evaluates to either TRUE (non-zero) or FALSE (zero).
- statement is the work to repeat (can be multiple statements in {…})

```c
#include <stdio.h>
int main(){

 for (int i=1;i<=2;i++)
     printf("%d\n",i);

 return 0;
}
```

Output:
1
2

```c
#include <stdio.h>
int main(){

 for (int i=1;i<=2;i++)
     printf("%d\n",i);
 printf("%d\n",i);
 return 0;
}
```

Output:

Compile time error — Variable i undeclared.

Block scope of a variable

```
#include <stdio.h>
int main(){

 for (int i=1;i<=2;i++){
     printf("%d\n",i);
 }
 return 0;
}
```

Output:
1
2

Block scope of a variable

```
#include <stdio.h>
int main(){

 { //start block
 int i;
 for (i=1;i<=2;i++)
     printf("%d\n",i);
 } //end block

 return 0;
}
```

- Output

  1

  2

Block scope of a variable

```
#include <stdio.h>
int main(){

 {
 int i;
 for (i=1;i<=2;i++)
     printf("%d\n",i);
 }
 printf("outside %d\n",i);

 return 0;
}
```

- Output

Compile time error: 'i' undeclared

Block scope of a variable

```
#include <stdio.h>
int main(){
 int i;
 for (i=1;i<=2;i++){
     printf("%d\n",i);
     int j=0;
     printf("j=%d\n",j+1);
 }

 return 0;
}
```

- Output?

1

j=1

2

j=1

Back to Break

- Used for exiting a loop forcefully
- Example Program:

Read at most 100 integer inputs from a user. Print the sum of inputs until a negative input is found (excluding the negative number) or sum of all 100 inputs.

```
int value;
int sum = 0;
int i;
for (i = 1; i <= 100; i++) {
     scanf("%d", &value);
     if (value < 0) {
          //-ve number: no need to go
          // around the loop any more!!
          break;
     }
     sum = sum + value;
}
printf("%d", sum);
```

To break or not to!

- Use of break sometimes can simplify exit condition from loop.
- However, it can make the code a bit harder to read and understand.
- Tip: if the loop terminates in at least two ways which are sufficiently different and requires substantially different processing then consider the use of termination via break for one of them.

Continue

- Used for skipping an iteration of a loop
- The loop is NOT exited.
- Example Program:

Read 100 integer inputs from a user. Print the sum of only positive inputs.

```
int sum = 0;
int i, value;
for (i = 0; i < 100; i++) {
     scanf("%d", &value);
     if (value < 0) {
          //-ve number; no need to add it
          // to the sum. Go ahead and
          // check the next input.
          continue;
     }
     sum = sum + value;
}
printf("%d", sum);
```

Break and Continue

- if there are nested loop: break and continue apply to the nearest enclosing loop only.

```
for (i = 0; i < 100; i++) {
    for (j = 0; j < 100; j++) {
        if (…) break;
    }
    …
}
```

Continue

```
i = 0;
while (i < 100) {
   scanf("%d", &value);
   if (value < 0) continue;
   sum = sum + value;
   i++;
}
```

i is never incremented
potentially infinite loop!!

- Make sure continue does not bypass update-expression for loops
  - Specially for while and do-while loops

Continue and Update Expr

- Correct Code:

```
i = 0;
while (i < 100) {
   i++;
   scanf("%d", &value);
   if (value < 0) continue;
   sum = sum + value;
}
```

Continue and Update Expr

- Correct Code:

```
i = 0;
while (i < 100) {
   scanf("%d", &value);
   if (value < 0) {
      i++;
      continue;
   }
   sum = sum + value;
   i++;
}
```

Quick question

```
int a =4;
while (a < 10) {
        if (a = 5)  {
                printf("%d\n", a);
        }
        a=a+1;  }
```

Probable intention:

```
int a =0;
while (a < 10) {
        if (a == 5)  {
                printf("%d", a);
        }
        a=a+1;  }
```

```
5
5
5
5
5
5
```

Practice Problem

- Write a program to read upto a maximum of 20 integers. Sum all the odd integers. Stop reading the sequence if you encounter a negative integer.

- Input: 2 3 8 1 5 -1
- Output: 9