CS11001/CS11002
Programming and Data Structures (PDS)
(Theory: 3-1-0)

Conditions and Branching

Nested If else

- Suppose that we want to compute the absolute value |xy| of the product of two integers x and y and store the value in z. Here is a possible way of doing it:
- if (x >= 0)

```
    { z = x;
        if (y >= 0)  z *= y;
        else z *= -y; }
 else { z = -x;
        if (y >= 0) z *= y;
        else z *= -y; }
```

- This can also be implemented as:

```
if (x >= 0) z = x; else z = -x;
if (y >= 0) z *= y; else z *= -y;
```

Here is a third way of doing the same:

- if (((x >= 0)&&(y >= 0)) || ((x < 0)&&(y < 0)))

```
        z = x * y;
else z = -x * y;
```

Repeated if-else statements

- A structure of the last figure can be translated into C as:
- if (Condition 1)
  { Block 1 }
  else if (Condition 2)
  { Block 2 }
  else if ... ... }
  else if (Condition n)
  { Block n }
  else
  { Block n+1 }

Example

- Here is a possible implementation of the assignment y = |x|:
- scanf("%d",&x);
  if (x == 0) y = 0;
  else if (x > 0) y = x;
  else y = -x;

The Switch Statement

```
switch (E)
  { case val1 : Block 1 break;
    case val2 : Block 2 break;
     ...
   case valn : Block n break;
   default: Block n+1
  }
```

Example

- char lang; ...
  switch (lang) {
  case ‘B’: printf(“Dhanyabad\n”); break;
  case 'E' : printf("Thanks\n"); break;
  case 'F' : printf("Merci\n"); break;
  case 'G' : printf("Danke\n"); break;
  case 'H' : printf("Shukriya\n"); break;
  case 'I' : printf("Grazie\n"); break;
  case 'J' : printf("Arigato\n"); break;
  case 'K' : printf("Dhanyabaadagaru\n"); break;
  default : printf("Thanks\n"); }

Switch is strange

- Switch statements are strange.
- It checks for the satisfying value of the condition it is checking.
- Once a match is found, further checks are disabled and all the subsequent statements are done one after the other, irrespective of the condition.

Example

- There are, however, situations where this odd behavior of switch can be exploited. Let us look at an artificial example. Suppose you want to compute the sum
- n + (n+1) + ... + 10

Using the strangeness of Switch

```
switch (n) {
   case 0  :
   case 1  : sum += 1;
   case 2  : sum += 2;
   case 3  : sum += 3;
   case 4  : sum += 4;
   case 5  : sum += 5;
   case 6  : sum += 6;
   case 7  : sum += 7;
   case 8  : sum += 8;
   case 9  : sum += 9;
   case 10 : sum += 10;
             break;
   default : printf("n = %d is not in the desired range...\n", n);
}
```

Displaying a menu and using Switch

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

  printf("Choice of destination:\n");
  printf("\t1 - Mercury\n");
  printf("\t2 - Venus\n");
  printf("\t3 - Mars\n");
  printf("Enter the number corresponding to your choice: ");
  scanf("%d",&choice);

  switch(choice)
  {
    case 1:
     puts("Mercury is closest to the sun.");
     puts("So, the weather may be quite hot there.");
     puts("The journey will cost you 10000 IGCs.");
     //break;
   case 2:
     puts("Venus is the second planet from the sun.");
     puts("The weather is probably hot and poisonous.");
     puts("The journey will cost 5000 IGCs.");
     break;
```

The output menu

```
case 3:
     puts("Mars is the closest planet to earth in the solar system.");
     puts("There is probably some form of life there.");
     puts("The journey will cost 3000 IGCs.");
     break;
  default:
    puts("Unknown destination.\n");
    break;
 }
 puts("\n Note: IGC = Inter Galactic Currency\n");

-bash-3.2$ ./a.out
Choice of destination:
        1 - Mercury
        2 - Venus
        3 - Mars
Enter the number corresponding to your choice:
```

Loops in C

Loops

- This is the first time we are going to make an attempt to move backward in a program. Loops make this backward movement feasible in a controlled manner. This control is imparted by logical conditions.
- Consider the computation of the **harmonic number**:
  - $H_n = 1/1 + 1/2 + 1/3 + ... + 1/n$.
- Initialize sum to 0. for each i in the set {1,2,...,n} add 1/i to sum. Report the accumulated sum as the output value.

Recursive definitions

- Use mathematical definition:
  - strong form
  - weak form
- Very useful technique to design algorithms.
- Consider the problem of generating all possible permutations of n numbers.
  - With n=1, there is just one possibility.
  - With other values of n:
    - Imagine that you have the permutations for n-1 numbers.
    - Insert the nth number into all the positions of each of these permutations.

Computing the gcd of a and b, two positive integers

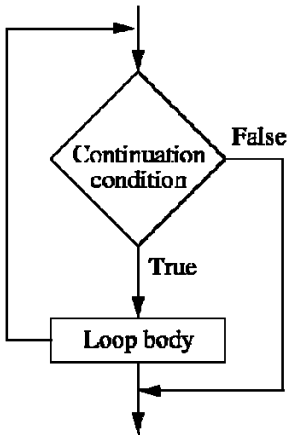
- gcd(a,b) is also computable in a recursive fashion.
- If a=0 or b=0, gcd(a,b)=0
- If a=b, gcd(a,b)=a
- If a > b, and r = a rem b. Then gcd(a,b) = gcd(b,r).

Iterative definition

- As long as b is not equal to 0 do the following:
  - Compute the remainder r = a rem b.
  - Replace a by b and b by r.
  - Report a as the desired gcd.

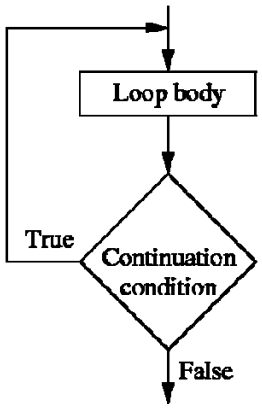
Pre-test loops

- The condition is checked first.
- If yes, then enters the body.
- After the loop body is executed, the control comes unconditionally to the start of the loop.
- But, now the condition might not hold anymore as the loop body may have changed it.
- So, the condition is again checked.
- If it is satisfied, enters the loop body again.
- If no, it goes beyond the end of the loop.



Post Test loops

- The control of execution enters the loop body unconditionally.
- After the entire body is executed, the loop condition is checked.
- If it is satisfied, control goes back to the top of the loop, the body is again executed and the continuation condition is again checked.
- This process is repeated until the continuation condition becomes false.
- In that case, control leaves the loop and proceeds further down the code.



The while loop

- while (the continuation condition is true) { execute loop body; }
- Example: gcd(a,b)

```
while (b > 0)
{
      r = a % b; /* Compute the next remainder */
     a = b; /* Replace a by b */
      b = r; /* Replace b by r */
 }
       printf("gcd = %d\n",a);
```

Example: Harmonic Number

```
float i, H;
unsigned int n;
 ... //Read n etc.
i = 0; H = 0;
while (i < n)
{ ++i; /* Increment i */
 H += 1.0/i; /* Update the harmonic number
    accordingly */
}
printf("H(%d) = %f\n", n, H);
```

Example: Fibonacci Number

```
 i = 1; /* Initialize i to 1 */
F = 1; /* Initialize Fi */
 p1 = 0; /* Initialize Fi-1 */
 while (i < n)
 {
    ++i; /* Increment i */
     p2 = p1;  /* The old Fi-1 now becomes Fi-2 */
     p1 = F; /* The old Fi now becomes Fi-1 */
     F = p1 + p2; /* Compute Fi from Fi-1 and Fi-2 */
  }
printf("F(%d) = %d", n, F);
```

The do-while loop

- The do-while loop of C is a post-test loop. It has the following syntax:
- do { execute loop body; }
  while (continuation condition is true);

The gcd using do-while

```
do {
   r = a % b; /* Compute the next remainder */
  a = b; /* Replace a by b */
  b = r; /* Replace b by r */
  }
  while (b > 0);
printf("gcd = %d\n",a);
```

Note that here b cannot be 0.