Programming in C

Chapter -4

Decisions and Loops

by, Santa Basnet

Decisions and Loops

Block/Compound statement

- C has braces { and }, are used to group declarations and statements together making a compound statement or block.
- All blocks are syntactically equivalent to a single statement.
- Example:

```
/* Operators demonstration program. */
#include <stdio.h>
int main()
{
    unsigned int x, y, z;
    int a = 2, b, c;
    {
        x = 7;
        y = 16;
        x &= y;
        c = ((b=2) == a);
    }
    printf("c = %d & x = %d\n", c, x);
    return(1);
}
```

Decisions and Loops

if-else

- C language allows you to write if and if-else statements to express the decisions.
- Formal syntax:

```
if (expression)
   statement1
else
   statement2
```

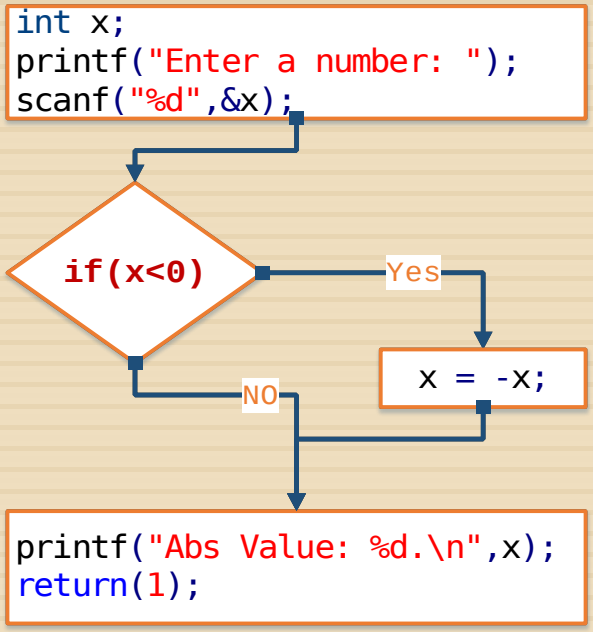
- The else part is optional.
- The expression is evaluated and if the result is non-zero(true), the $statement_1$ is executed otherwise the $statement_2$ is executed.

Decisions and Loops

if-else

- if Statement: Example

```c
/*if Statement: Absolute value of an integer*/
#include<stdio.h>
int main()
{
    int x;
    printf("Enter a number: ");
    scanf("%d",&x);
    if(x<0) {
        x = -x;
    }
    printf("The absolute value is %d.\n",x);
    return(1);
}
```

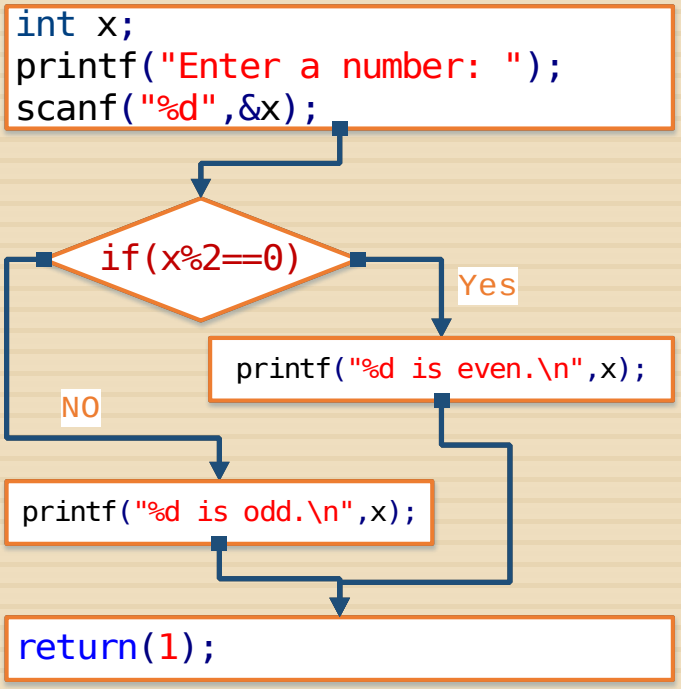


Decisions and Loops

if-else

- If-else Statement: Example

```
/*if-else statement: Odd or Even Number*/
#include<stdio.h>
int main()
{
    int x;
    printf("Enter a number: ");
    scanf("%d",&x);
    if(x % 2 == 0) {
        printf("%d is even.\n",x);
    } else {
        printf("%d is odd.\n",x);
    }
    return(1);
}
```



Decisions and Loops

if-else if … else

- C supports multi-way selection of more than one expressions using if-else if … else statement and the syntax is:

```
if (expression)
    statement
else if (expression)
    statement
else if (expression)
    statement
else if (expression)
    statement
else
    statement
```

When expression is true(non-zero), the statement block associated with it is executed and terminates the sequence. The else handles the default or otherwise case and is optional.

Decisions and Loops

if-else if … else

- C supports multi-way selection of more than one expressions using if-else if … else statement and the Example is:

```
/*if-else if ... else statement: Grade Calculator*/
#include<stdio.h>
int main()
{
    int x;
    printf("Enter mark of a subject :");
    scanf("%d",&x);
    if(x < 0 || x > 100) {
        printf("Invalid input !\n");
    } else if (x >= 90) {
        printf("Grade - A\n");
    }else if (x >= 80) {
        printf("Grade - B\n");
    }else if (x >= 70) {
        printf("Grade - C\n");
    }else if (x >= 60) {
        printf("Grade - D\n");
    }else {
        printf("Grade - F\n");
    }
    return(1);
}
```

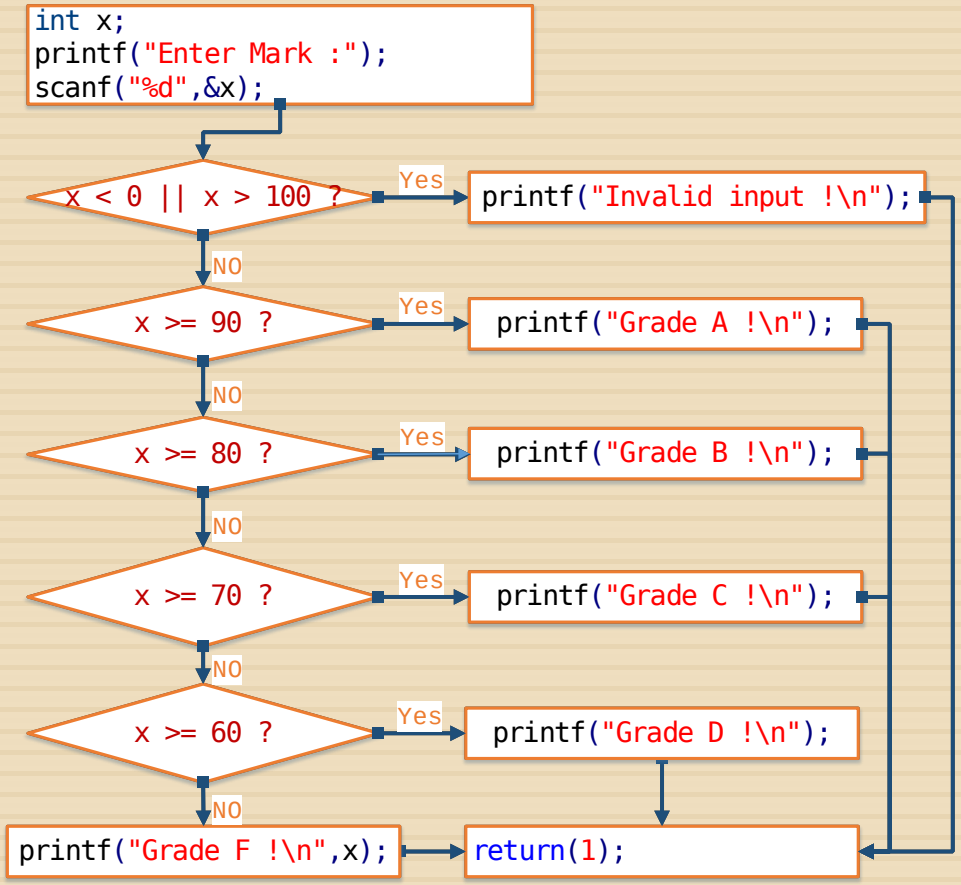
Grade Selection:

```
90 to 100 -> A
80 to  90 -> B
70 to  80 -> C
60 to  70 -> D
 0 to  60 -> F
```

Decisions and Loops

if-else if … else

- if-else if … else statement in a chart:



Grade Selection:

```
90 to 100 -> A
80 to  90 -> B
70 to  80 -> C
60 to  70 -> D
 0 to  60 -> F
```

Decisions and Loops

if-else if … else

- Even more way of selections:

```
if(expr_1){
    if(expr_2){
        statement_1
    }else{
        if(expr_3){
            statement_2
        }else{
            statement_3
        }
    }
}
else{
    if(expr_4){
        statement_4
    }else{
        if(expr_5){
            statement_5
        }else{
            statement_6
        }
    }
}
```

Decisions and Loops

for, while & do-while

- Loops allows you to execute certain statements repeatedly until thee evaluation of expression becomes zero.
- C have three constructs of writing loops: while, for and do-while.

1. while:

```
while(expression){
    statement;
}
```

```
/*while-example: Print first n natural numbers*/
#include <stdio.h>
int main(){
    int n, iter = 1;
    printf("How many : ");
    scanf("%d", &n);
    while(iter <= n){
        printf("%d\n", iter);
        iter++;
    }
    return(0);
}
```

Decisions and Loops

for, while & do-while

- Loops allows you to execute certain statement repeatedly until the expression becomes zero.
- C have three constructs of writing loops: while, for and do-while.

2. for:

```
for(expr; expr; expr){
    statement;
}
```

```
/*for: Print first n natural numbers*/
#include <stdio.h>
int main(){
    int n, iter;
    printf("How many : ");
    scanf("%d", &n);
    for(iter = 1; iter <= n; iter++){
        printf("%d\n",iter);
    }
    return(0);
}
```

Decisions and Loops

for, while & do-while

- Loops allows you to execute certain statement repeatedly until the expression becomes zero.
- C have 3 constructs of writing loops: while, for and do-while.

3. do-while:

```
do{
    statement;
} while(expr);
```

```
/*do-while: Print first n natural numbers*/
#include <stdio.h>
int main()
{
    int n, iter = 1;
    printf("How many : ");
    scanf("%d", &n);
    do{
        printf("%d\n",iter);
        iter++;
    }while(iter <= n);
    return(0);
}
```

Decisions and Loops

for, while & do-while

- Loop for-ever: when the evaluation of the expression always become non-zero, the loop goes forever i.e. an infinite loop.

```
1. for(;;){
      statement;
   }

2. do{
      statement;
   } while(1);

3. while(1){
      statement;
   }

4. for(i=1; i<5; i--){
      statement;
   }
```

Decisions and Loops

for, while & do-while

- Loop for-ever: an example, false infinite.

```
/*for: Infinite loop but stops*/
#include <stdio.h>
int main(){
    short n = 10, i;
    for(i = 1; i < n; i--){
        printf("%d\n",i);
    }
    return(0);
}
```

Decisions and Loops

for, while & do-while

- Equivalency in for and while loop:

```
1. for(expr1;expr2;expr3){
        statement;
   }

2. expr1;
   while(expr2){
        statement;
        expr3;
   }
```

```
#include <stdio.h>
int main()
{
    short n = 100, i;
    for(i=1; i < n; i++){
        printf("%d\n",i);
    }
    return(0);
}
```

```
#include <stdio.h>
int main()
{
    short n = 100, i = 1;
    while(i < n){
        printf("%d\n", i);
        i++;
    }
    return(0);
}
```

Decisions and Loops

break – continue

- break statement allows early exit to you from for, while and do-while loop.
- It causes innermost loop and switch to be exited immediately.

```
for(expr1;expr2;expr3){
    if(expr4){
        statement;
        break;
    }
    statement;
}
```

```
#include <stdio.h>
int main()
{
    short n = 10, i;
    for(i=1; i < n; i--){
        printf("%d\n",i);
        if(i < -10){
            break;
        }
    }
    return(0);
}
```

Decisions and Loops

break – continue

- continue statement allows you to skip execution of statement from the for, while and do-while loop.
- It causes next iteration of the enclosing for, while and do-while loop to begin.

```
for(expr1;expr2;expr3){
    if(expr4){
        statement;
        continue;
    }
    statement;
}
```

```
/*continue: Number Division*/
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char ch = 'y'; float x, y;
    while(ch == 'y' || ch == 'Y'){
        system("cls");
        printf("Enter two numbers: ");
        scanf("%f%f",&x,&y);
        if(y == 0) continue;
        printf("X/Y = %6.2f\n",(x/y));
        printf("Do you want to continue(y|Y):");
        scanf(" %c", &ch);
    }
}
```

Decisions and Loops

switch – case

- The switch statement allows you to make is a multi-way decision with the evaluated value of an expression, i.e. a number of constant integer.

```
switch(expression){
    case const_expr_1:
        statement_1;
    case const_expr_2:
        statement_2;
    case const_expr_3:
        statement_3;
    default:
        statement_4;
}
```

Decisions and Loops

switch – case

```
/*switch - example: An arithmetic operator*/
#include <stdio.h>
int main(){
    int x, y; char op;
    printf("Enter two numbers: ");
    scanf("%d%d", &x, &y);
    printf("Enter operator (+, -, *, /) : ");
    scanf("%c",&op);
    switch(op) {
        case '+':
            printf("%d + %d = %d\n", x, y, x+y); break;
        case '-':
            printf("%d - %d = %d\n", x, y, x-y); break;
        case '*':
            printf("%d * %d = %d\n", x, y, x*y); break;
        case '/':
            if(y == 0) printf("Undefined !\n");
            else printf("%d / %d = %d", x, y, x/y);
            break;
        default: printf("Invalid operator !\n");
        }
}
```

Decisions and Loops

goto – label

- goto statement allows you to jump to the label statement and is infinitely-abusable within program.
- We can write source code without using goto statement.

With goto

```
for (i=0; i<n;i++)
    for (j=0;j<m;j++)
        if (a[i] == b[j])
        goto found;
...
found:
...
```

Without goto

```
found = 0;
for (i=0; i<n;i++)
    for (j=0;j<m;j++)
        if (a[i] == b[j])
        found = 1;
...
if(found){
}else{
}
...
```

Thank you.

Questions ?