Nested If/Else If/Switch Case in C

if-else statement

- **else** part is optional
- The **else** is associated with closest **else**-less **if**

-
  ```
  if(n>0)
          if(a>b) z=a;
  else z=b;
  ```

Even though the else is indented with the first if, by rule it will be associated with the nearest if

- Braces must be used to force association with the first

-
  ```
  if(n>0)
  {
          if(a>b) z=a;
  }
  else z=b;
  ```

Nested if

```
#include<stdio.h>
int main(void)
{
    int id;
    printf("Please enter last 3 digits of your
    id:\n");
    scanf("%d", &id);
    printf("You are in ");
    if(id%2)
    {
            if(id<60)
                        printf("A1\n");
            else
                        printf("A2\n");
    }
```

```
else
{
            if(id<61)
                        printf("B1\n");
            else
                        printf("B2\n");
}
   return 0;
}
```

Blocks of Code

- Surround the statements in a block with opening and ending curly braces.
- One indivisible logical unit
- Can be used anywhere a single statement may
- Multiple statements
- Common programming error:
  - Forgetting braces of compound statements/blocks

Blocks of Code

- if(*expression*) {

```
if(expression) {
     statement1;
     statement2;
     ...
     statementN;
}
else {
     statement1;
     statement2;
     ...
     statementN;
}
```

- If *expression* is **true** all the statements with if will be executed
- If *expression* is **false** all the statements with else will be executed

if-else if statement

- if(*expression*)

        statement;

  else if (*expression*)

        statement;

  else if (*expression*)

        statement;

  else

        statement;

if-else if statement

- Multi-way decision
- *expressions* are evaluated in order
- If the *expression* of any **if** is *true*
  - the *statement* associated with it is executed
    - Multiple *statements* can be associated using curly braces
  - the whole chain is terminated
- If none of the *expressions* are true
  - **else** part is executed
  - Handles none of the above/ default case
  - Optional

if-else if statement

```
#include<stdio.h>
int main( )
{
   int num;
   scanf("%d", &num);
   if(num>=80)
          printf("5.0\n");
   else if(num>=75)
          printf("4.75\n");
   else if(num>=70)
          printf("4.50\n");
   else
          printf("0.0");
   return 0;
}
```

Use of logical operator

```
#include<stdio.h>
    int main( ){
    char ch;
    scanf("%c", &ch);
    if(ch>='A' && ch<='Z')
           printf("%c\n", ch+('a'-'A'));
    else if(ch>='a' && ch<='z')
           printf("%c\n", ch-('a'-'A'));
    else
           printf("Invalid\n");
    return 0;
}
```

Short Circuit Evaluation

```
if(a!=0 && num/a)
{
}
```

- If first operand of '&&' is zero, the 2nd operand is not evaluated
- If first operand of '||' is nonzero, the 2nd operand is not evaluated

Conditional Expressions

- Uses **ternary** operator "?:"
- *expression1?expression2*:*expression3*;
- z= (a>b)? a: b; /* z=max(a,b);*/
- Can be used anywhere an expression can be

Switch Case

```
switch (expression) {
        case constant: statements
        case constant: statements
        default: statements
}
```

- Use of break

Switch Case

```
switch (month) {
        case 1: printf("January\n");
        case 2: printf("February\n");
        default: printf("Invalid\n");
}
```

Switch Case

```
#include<stdio.h>
int main()
{
    char ch;
    scanf(“ %c” , &ch);
    switch (i) {
            case '0': case '1': case '2': case '3': case '4': case '5': case '6': case '7': case '8': case '9':
                        printf("digit\n");
                        break;
            default:  printf("non digit\n");}
    return 0;
}
```

- Use of break

Switch Case (use of break)

```
int x, a, b;
scanf("%d %d", &a, &b);
switch (b) {
        case 0:
        printf("divide by zero error\n");
        default: x=a/b;
}
```

Switch Case (use of break)

```
int x, a, b;
scanf("%d %d", &a, &b);
switch (b) {
        case 0:
        printf("divide by zero error\n");
        break;
        default: x=a/b;
}
```

Symbolic Constant

- A name that substitutes for a sequence of characters
- #define *name replacement*
- Any occurrence of *name* (not in quotes and not part of another name) will be replaced by corresponding *replacement*
- #define PI 3.141593
- #define TRUE 1
- #define FALSE 0