Belajaritma

Algoritma dan Pemrograman

Sesi III: Control Flow
(PDF Example)

There are 5 types of control flow in a typical C Language program, which are :

1. **If-else Statement**
2. **For Loop**
3. **While & do-while Loop**
4. **Break & Continue**
5. **Switch Statement**

If-Else Statements

If-else statements are the most commonly used flow control. A basic structure of an if-else statement is :

```
if (test expression1)
{
   // statements to be executed if the first test expression is true
}
else if (test expression2)
{
  // statements to be executed if the second test expression is true
}
else
{
  // statements to be executed if both test expressions are false
}
```

Nested if-else statements are if-else inside another if-else statement. For example :

```c
#include <stdio.h>
int main()
{
    int number1, number2;
    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    if (number1 >= number2) // checks if number 1 is equal or larger than number 2
   {
      if (number1 == number2) // if equal
      {
        printf("Result: %d = %d",number1,number2);
      }
      else // if larger
      {
        printf("Result: %d > %d", number1, number2);
      }
   }
    else // this means number 1 is smaller than number 2
    {
        printf("Result: %d < %d",number1, number2);
    }
    return 0;
}
```

For-Loop

For-loops are used to repeat a block of code until the specified condition is met. A basic structure of for-loop is :

```c
for (initializationStatement; testExpression; updateStatement)
//example : (i=0;i<5;i++)
{
    // statements inside the body of loop
}
```

An example for a for-loop is :

```c
// Program to calculate the sum of first n natural numbers
// Positive integers 1,2,3...n are known as natural numbers

#include <stdio.h>
int main()
{
    int num, count, sum = 0;

    printf("Enter a positive integer: ");
    scanf("%d", &num);

    // for loop terminates when num is less than count
    for(count = 1; count <= num; ++count)
    {
        sum += count;
    }

    printf("Sum = %d", sum);

    return 0;
}
```

This program asks the user to input a number, then it will count itself from 1 to the inputted number.

EX :

input = 10, then program will run 10 times

1 + 2 +3 +4 .....+10 = 55

While and Do-while Loops

Do-while and while loops are essentially the same loop, with a single difference;

While loops will evaluate the test expression inside the parentheses first, before executing it.

Do-while loops will execute it at least once, then evaluate the test expression.

Basic structures for both loops :

```
while (testExpression) //while loop
{
    // statements inside the body of the loop
}

do
{
   // statements inside the body of the loop
}
while (testExpression); // do-while loop
```

an example of a while loop :

```
// Print numbers from 1 to limit

#include <stdio.h>
int main()
{
    int i = 1;
    int limit;
    scanf ("%d",&limit);
    while (i <= limit)
    {
        printf("%d\n", i);
        ++i;
    }

    return 0;
}
```

an example of a do-while loop :

```c
// Program to add numbers until the user enters zero

#include <stdio.h>
int main()
{
    double number, sum = 0;

    // the body of the loop is executed at least once
    do
    {
        printf("Enter a number: ");
        scanf("%lf", &number);
        sum += number;
    }
    while(number != 0.0);

    printf("Sum = %.2lf",sum);

    return 0;
}
```

Break and Continue

Self explanatory. Break == ending the code, Continue == proceed Example of break :

```c
// Program to calculate the sum of numbers (10 numbers max)
// If the user enters a negative number, the loop terminates

#include <stdio.h>
int main() {
   int i;
   double number, sum = 0.0;

   for (i = 1; i <= 10; ++i)
  {
      printf("Enter a n%d: ", i);
      scanf("%lf", &number);

      // if the user enters a negative number, break the loop
      if (number < 0.0) {
         break;
      }

      sum += number; // sum = sum + number;
   }

   printf("Sum = %.2lf", sum);

   return 0;
}
```

If the user enters a negative number, end the loop, regardless if i hasn't reached 10 or not.

Example of continue :

```
// Program to calculate the sum of numbers (10 numbers max)
// If the user enters a negative number, it's not added to the result

#include <stdio.h>
int main() {
   int i;
   double number, sum = 0.0;

   for (i = 1; i <= 10; ++i) {
      printf("Enter a n%d: ", i);
      scanf("%lf", &number);

      if (number < 0.0)
      {
         continue;
      }

      sum += number; // sum = sum + number;
   }

   printf("Sum = %.2lf", sum);

   return 0;
}
```

In this program, if the input is lower than 0, then continue (it will not be added to the sum).

Switch Statements

It is an alternative (subjectively, clearer and more-readable) version of an if-else statement.

Basic structure of a switch statement :

```
switch (expression)
{
    case constant1:
      // statements
      break;

    case constant2:
      // statements
      break;
    .
    .
    .
    default:
      // default statements
}
```

Mechanics of a switch statement :

The expression is evaluated once and compared with the values of each case label.

- If there is a match, the corresponding statements after the matching label are executed. For example, if the value of the expression is equal to constant2, statements after case constant2: are executed until break is encountered.
- If there is no match, the default statements are executed.
- Remember to use break. If not, all statements after the matching label are executed.

A basic example of a switch statement can be applied in a simple calculator :

```c
// Program to create a simple calculator
#include <stdio.h>
int main() {
    char oper;
    double n1, n2;

    printf("Enter an operator (+, -, *, /): ");
    scanf("%c", &oper);
    printf("Enter two operands: ");
    scanf("%lf %lf",&n1, &n2);

    switch(oper) //will check oper input
    {
        case '+': // if oper == '+'
            printf("%.1lf + %.1lf = %.1lf",n1, n2, n1+n2);
            break;

        case '-': // if oper == '-'
            printf("%.1lf - %.1lf = %.1lf",n1, n2, n1-n2);
            break;

        case '*': // if oper == '*'
            printf("%.1lf * %.1lf = %.1lf",n1, n2, n1*n2);
            break;

        case '/': // if oper == '/'
            printf("%.1lf / %.1lf = %.1lf",n1, n2, n1/n2);
            break;

        // operator doesn't match any case constant +, -, *, /
        default:
            printf("Error! operator is not correct");
    }

    return 0;
}
```