Day04 A

Young W. Lim

2017-10-07 Sat

Outline

1. Based on

2. Structured Programming (1)
   - Algorithms and Flowcharts
   - Examples

Based on

"C How to Program",
Paul Deitel and Harvey Deitel

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

Algorithms

- execution of a series of actions in a specific order
- a procedure for solving problem in terms of
  the actions to be executed, and
  the order in which these actions are to be executed
- pseudocode
  - artificial and informal language
  - not an actual programming language
  - helps you to think out a program
  - consists of only action statements not definitions

Control Structures

- sequential execution: statements are executed one after the other in the given order
- transfer of control : out of this "one after the other" sequence can execute a statement that is not the next following statement
- goto statement used in the early ages
  difficult to manage (spaghetti codes)
- three control structures
  - sequence control structure : one after the other execution
  - selection control structure : `if`, `if ... else` statements
  - repetition control structure : `while`, `do ... while`, `for`

Flow Charts

- rectangle : action symbols (calculation, input, output)
- rounded rectangles : Begin, End
- small circles : connecting symbols
- diamonds : decision symbols
- flowlines : the order of the actions

- structured programming and flowcharts
  - stacked building blocks
    flowchart segments can be attached to one another
    connect the exit of one segment to the entry of the other
  - nested building blocks
    any rectangle can be replaced by any control statement

Selection Control Structures

- single selection statment `if`
  - either performs/selects an action if a condition is met
  - or skips/ignores the action if the codition is false
- double selection statement `if ... else`
  - either performs/selects an action if a condition is met
  - or performs/selects the other action if the codition is false
- multiple selection statement `switch`
  - performs/selects one of many different actions depending on the *value* of the expression

Repetion Control Structures

- repetition / iteration / loop statement
- C provides : (a) `while` (b) `do ... while` (c) `for`
- counter controlled repetition
  definite repetition : the number of repetition is known in advance
- sentinel controlled repetition
  indefinite repetition : the number of repetition is not know in advance
- control variable : incremented / decremented each time
- sentinel value : denotes the end of data
  regular data and the end of data (sentinel value)

Examples (1)

```
#include <stdio.h>

int main(void) {
  int S;

  S = 1+2+3+4+5;

  printf("S= %d \n", S);

}

---
$ gcc -Wall t.c
$ ./a.out
S= 15
```

- not useful when the numbers are many.

Examples (2)

```
#include <stdio.h>

int main(void) {
  int S;

  S = (((((1)+2)+3)+4)+5);

  S = 0;
  S = S + 1;    // S = 1;
  S = S + 2;    // S = (1) + 2
  S = S + 3;    // S = ((1)+2) + 3
  S = S + 4;    // S = (((1)+2)+3) + 4
  S = S + 5;    // S = ((((1)+2)+3+4) + 5

  printf("S= %d \n", S);
}

---
$ gcc -Wall t.c
$ ./a.out
S= 15
```

- accumuation variable S
- add one number at a time
- can think in this way
  `S = S + i, (i=1, ..., 5)`
- then we have the same
  statement `S = S+i`

Examples (3)

```
#include <stdio.h>

int main(void) {
  int i, S;

  i = 0;   S = 0;
  i = i+1; S = S + i;
  i = i+1; S = S + i;
  i = i+1; S = S + i;
  i = i+1; S = S + i;
  i = i+1; S = S + i;

  printf("S= %d \n", S);

}
```

```
---
$ gcc -Wall t.c
$ ./a.out
S= 15
```

- accumulation variable S
- loop variable i
- increment i by 1
  to make (i=1, 2, 3, 4, 5)
- the same statements are
  repeated 5 times
- i can start from 1, also.

Examples (4)

```
#include <stdio.h>

int main(void) {
  int i, S;

  S = 0;      i = 1;
  S = S + i; i = i+1;
  S = S + i; i = i+1;
  S = S + i; i = i+1;
  S = S + i; i = i+1;
  S = S + i; i = i+1;

  printf("S= %d \n", S);

}
```

```
---
$ gcc -Wall t.c
$ ./a.out
S= 15
```

- if i starts from 1, then i = i+1 statement must used after S = S + i
- Note that alogrithm = actions + orders
- Now, let's use the if statement

Examples (5)

```
#include <stdio.h>

int main(void) {
  int i, S;

  i = 0;   S = 0;
  if (i<5) { i = i+1; S = S + i; }
  if (i<5) { i = i+1; S = S + i; }
  if (i<5) { i = i+1; S = S + i; }
  if (i<5) { i = i+1; S = S + i; }
  if (i<5) { i = i+1; S = S + i; }

  if (i<5) { i = i+1; S = S + i; }
  if (i<5) { i = i+1; S = S + i; }
  printf("S= %d \n", S);

}
```

- as long as the condition is met, the same statement is executed
- the first 5 statements are executed
- the last 2 statements are not executed because the condition is not met.

```
---
$ gcc -Wall t.c
$ ./a.out
S= 15
```

Examples (6)

```
#include <stdio.h>

int main(void) {
  int i, S;

  S = 0;      i = 1;
  if (i<=5) { S = S + i; i = i+1; }
  if (i<=5) { S = S + i; i = i+1; }
  if (i<=5) { S = S + i; i = i+1; }
  if (i<=5) { S = S + i; i = i+1; }
  if (i<=5) { S = S + i; i = i+1; }

  if (i<=5) { S = S + i; i = i+1; }
  if (i<=5) { S = S + i; i = i+1; }

  printf("S= %d \n", S);

}
```

```
---
$ gcc -Wall t.c
$ ./a.out
S= 15
```

- note the condition when `i` starts from 1
- Now, we can use the `while` statement

Examples (7)

```
#include <stdio.h>

int main(void) {
  int i, S;

  i = 0;   S = 0;
  while (i<5) { i = i+1; S = S + i; }

  printf("S= %d \n", S);

}

---
$ gcc -Wall t.c
$ ./a.out
S= 15
```

- i starts from 0
- as long as the condition (i<5) is met
- the same statements are executed
  - increments i by 1
  - accumulates S by adding i
- after 5 repeatitions, i becomes 5
- the condition does not be met
- no more repeatition, escape, exit, break

Examples (8)

```
#include <stdio.h>

int main(void) {
  int i, S;

  S = 0;      i = 1;
  while (i<=5) { S = S + i; i = i+1; }

  printf("S= %d \n", S);

}

---
$ gcc -Wall t.c
$ ./a.out
S= 15
```

- i starts from 1
- as long as the condition (i<=5) is met
- the same statements are executed
  - accumulates S by adding i
  - increments i by 1
- after 5 repeatitions, i becomes 6
- the condition does not be met
- no more repeatition, escape, exit, break

Examples (9)

```
#include <stdio.h>

int main(void) {
  int i=0, S=0;

  while (i<5) {
    i = i + 1;
    S = S + i;
  }

  printf("S= %d \n", S);

}

---
$ gcc -Wall t.c
$ ./a.out
S= 15
```

Examples (10)

```
#include <stdio.h>

int main(void) {
  int i=0, S=0;

  while (i<6) {
    S = S + i;
    i = i + 1;
  }

  printf("S= %d \n", S);

}

---
$ gcc -Wall t.c
$ ./a.out
S= 15
```