Programming Techniques – 2501405

Introduction

Evaluation

- Take part in class
- Tests
  - *Regular test*
  - *Midterm*
  - *Final Exam*

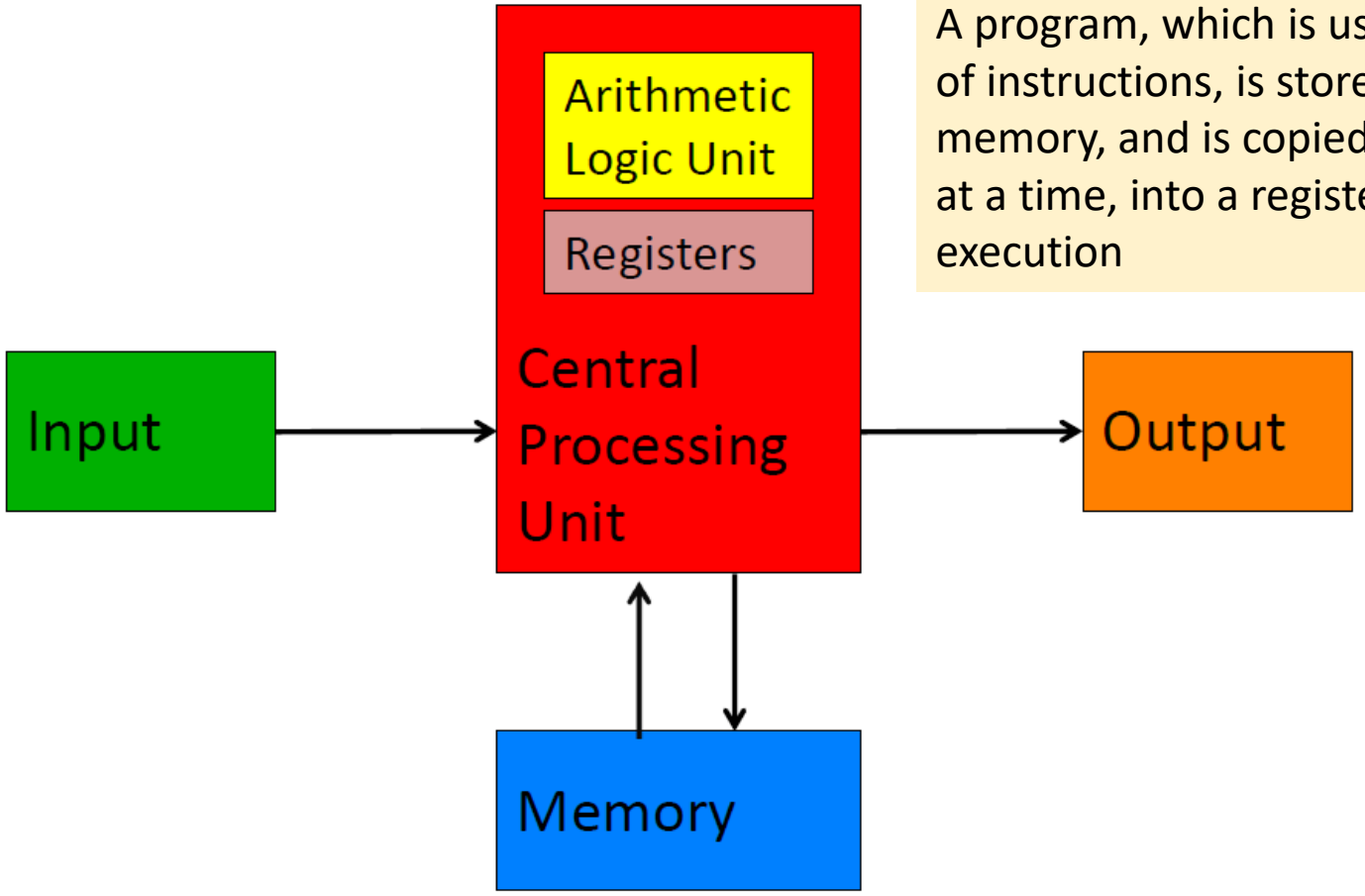
Materials

- Paul J. Deitel, Harvey M.Deitel. *C++ How to programme 9nd Edition*. Boston, Pea
- Brian W. Kernighan and Dennis M. Ritchie The C Programming Language

Algorithms

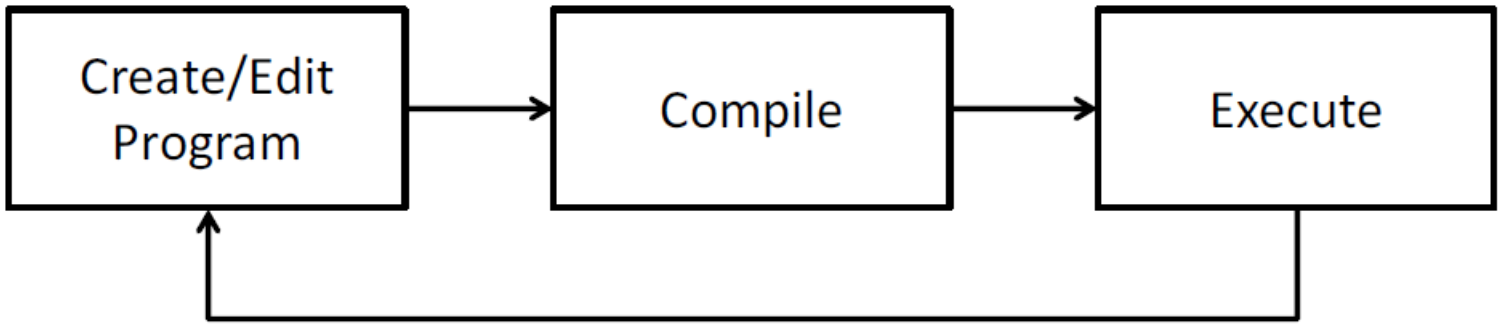
- **Definition:** An *algorithm* is a recipe for solving a problem.
- **Programs are ways of carrying out algorithms!!!**

Von Neumann Architecture



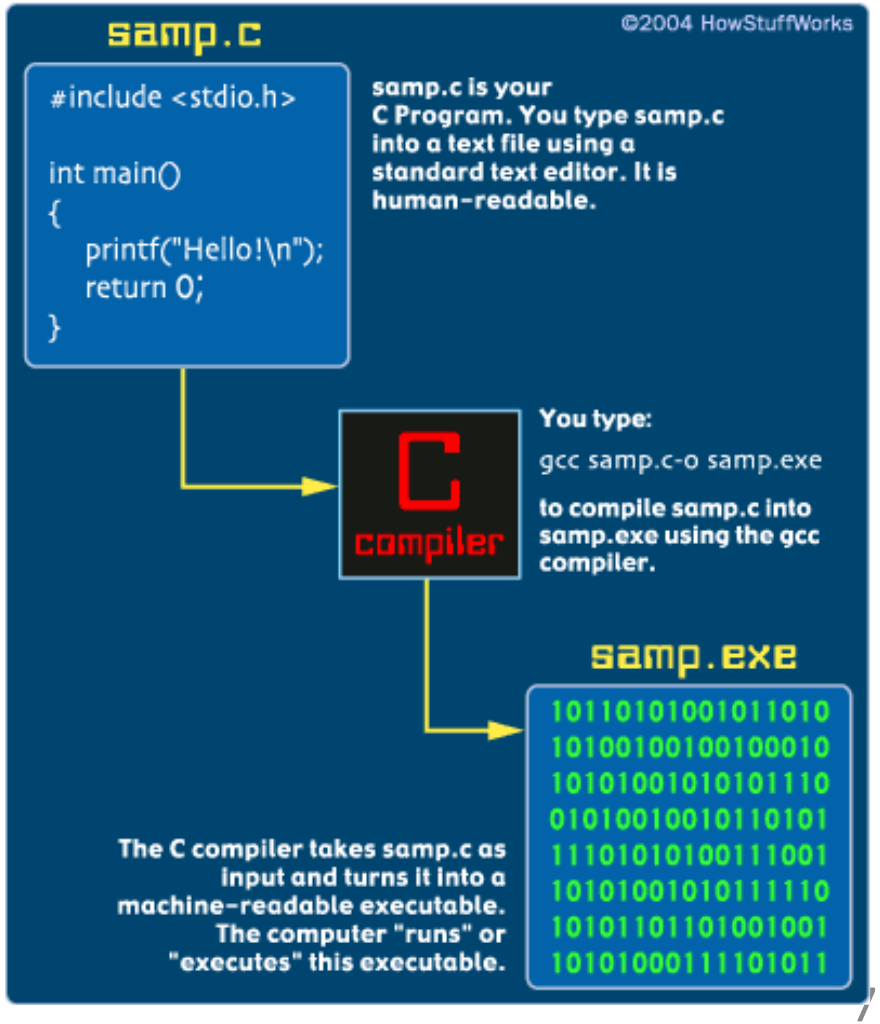
A program, which is usually a long list of instructions, is stored in the main memory, and is copied, one instruction at a time, into a register in the CPU for execution

The Programming Process in C



The Programming Process in C

- A compiler generates a file containing the translation of the program into the machine's native code.
  - The compiler does not actually execute the program!
  - Instead, you first execute the compiler to create a native executable, and then you execute the generated executable.



The Programming Process in C

- After creating a C program, executing it is a two step process:

```
me@computer:~$ gcc my_program.c –o my_program
me@computer:~$ ./my_program
```

The Programming Process in C

me@computer:~$ gcc my_program.c –o my_program

me@computer:~$ **./my_program**

- invokes the compiler, named *gcc*
- The compiler reads the source file my_program.c containing the C codes
- It generates a new file named my_program containing a translation of this code into the binary code used by the machine

The Programming Process in C

me@computer:~$ gcc my_program.c –o my_program

me@computer:~$ **./my_program**

- tells the computer to execute this binary code.

Revision

1. How many data types are there?
2. How many bits does a byte occupy? For example.
3. How many basic data types are there in the C language?
4. A *data structure?*
5. *What is an array? For example.*

Quiz

- **Bộ phận nào của máy tính chứa dữ liệu và mã thực thi khi chạy chương trình:**

A. Màn hình

B. Bộ nhớ RAM

C. Bàn phím

D. CPU

Quiz

- Trong chương trình viết bằng C, biểu thức **2*13%7 + 19/4** có giá trị bằng

A. 6

B. 10

C. 9

D. 9.75

Quiz

- **Máy tính có thể thực thi trực tiếp các đoạn mã được viết bằng:**

A. Ngôn ngữ máy
B. Ngôn ngữ cao cấp
C. Ngôn ngữ Assembly
D. Tất cả đều sai

Quiz

- Có mấy kiểu chú thích trong chương trình C

A. 1

B. 2

C. 3

D. 4

Quiz

- Câu lệnh **printf("Tab\\tExample");** xuất ra màn hình:

A. Tab\\tExample
B. Tab\\tExample
C. Tab Example
D. Tab\tExample

Quiz

- **Hãy cho biết sau khi thực hiện đoạn chương trình sau, x có giá trị bằng bao nhiêu**

```
int     x = 2, y = 7;
x *=(y + 2) + 3;
```

A. 12

B. 21

C. 24

D. 4096

Quiz

- **Câu lệnh nào sau đây KHÔNG tăng giá trị của biến a lên 1**

A. a++

B. a += 1

C. a = a + 1

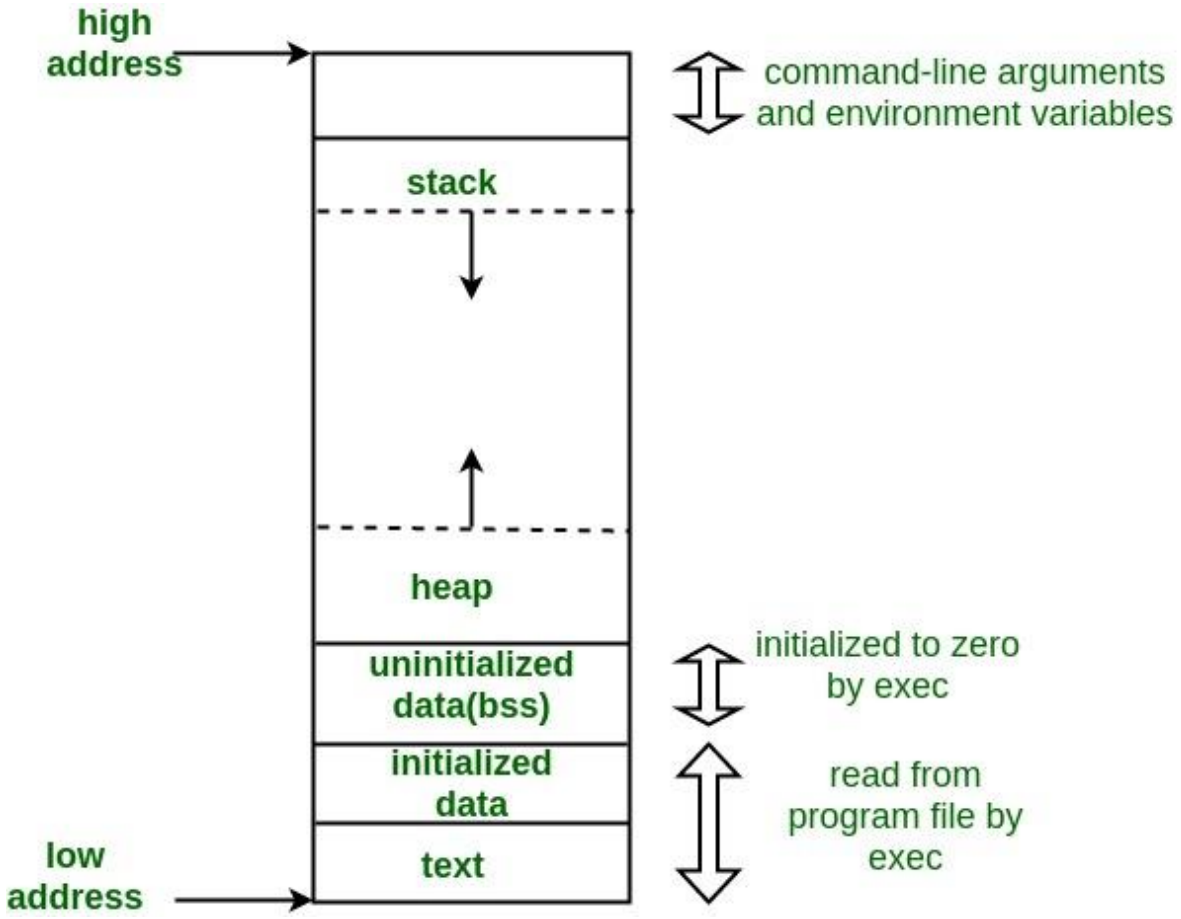
D. a += a + 1

Arithmetic Operators

Operators						Associativity	Type
++	--	+	-	!	(type)	right to left	unary
*	/	%				left to right	multiplicative
+	-					left to right	additive
<	<=	>	>=			left to right	relational
==	!=					left to right	equality
&&						left to right	logical AND
||						left to right	logical OR
?:						right to left	conditional
=	+=	-=	*=	/=	%=	right to left	assignment
,						left to right	comma

Operator precedence and associativity.

Memory Layout of C Program



Memory Layout of C Program

- **Text or Code Segment:** Text segment contains machine code of the compiled program
- **Initialized Data Segment:** Initialized data stores all global, static, constant, and external variables (declared with extern keyword)
- **Uninitialized Data Segment:** all global variables and static variables that are initialized to 0 or do not have explicit initialization in source code.
- **Heap:** *dynamic memory allocation*

Scope of a variable

- Local variables
- Global variables
- Static variables

Static Variable

Lifetime: A static variable has a lifetime that extends throughout the entire execution of the program.

Scope: A static variable can have either global or local scope. If declared inside a function, it has local scope and if declared outside of any function, it becomes a global variable.

```
void test(){
    static int count = 0;
    count = count + 1;
    printf("%d\n", count);
}
```

```
int main(){
    test();
    printf("Hello World\n");
    test();
    return 0;
}
```

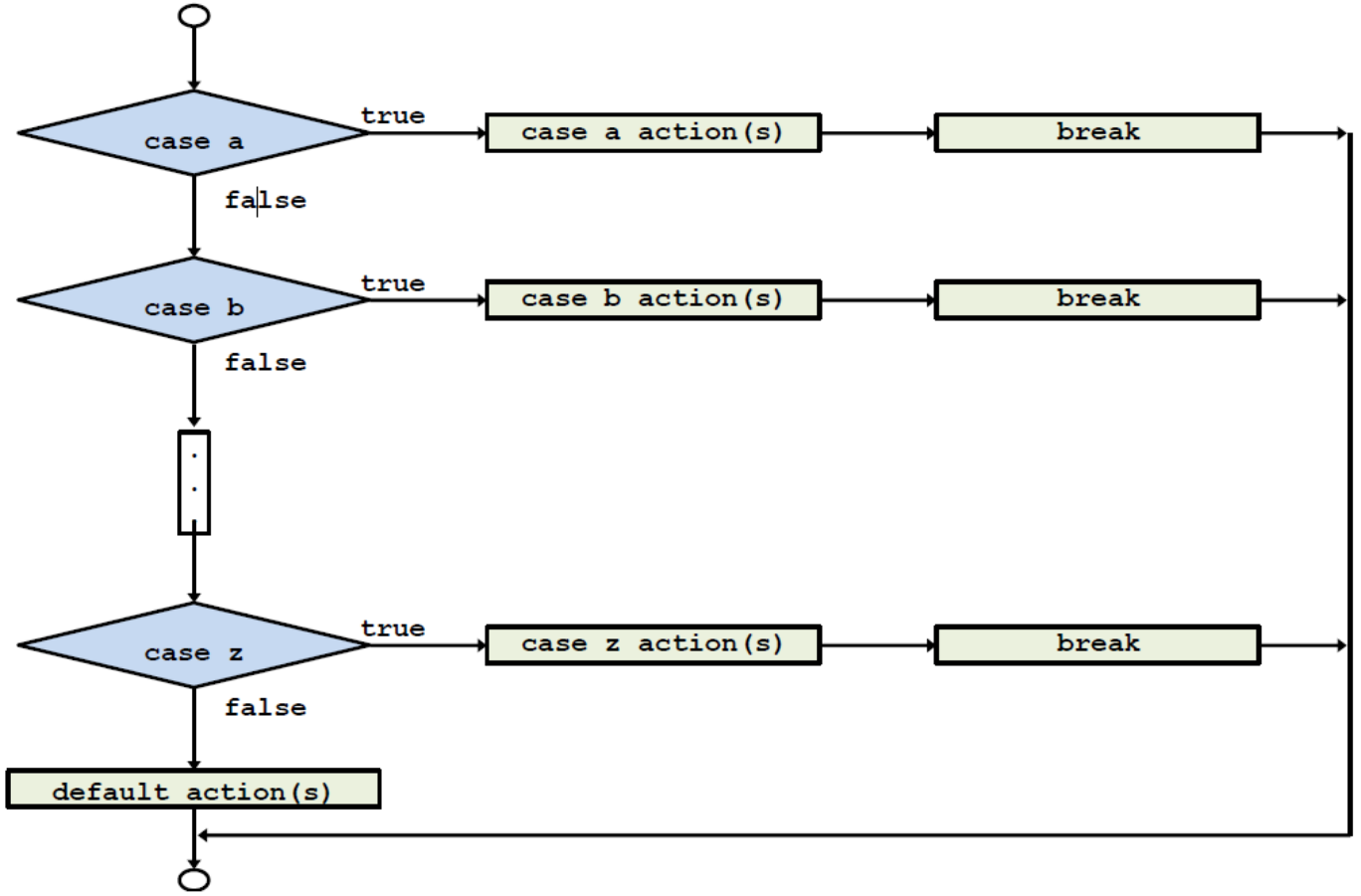
Quiz

```
int a = 5, b = 6, c = 10;
float p;
p = ((a == 5) + (b < 10) + (c > 15) ) * (c++)/ (++b);
printf("p = %f\n", p);

p = ((a == 5) + (b < 10) + (c > 15) ) * (c++)/ (float)(++b);
printf("p = %f\n", p);
```

The **switch** Structure

```
switch ( a_variable or expr ){
    case value1:
        actions
    case value2 :
        actions
        …
    default:
        actions
}
```

The **switch** Structure

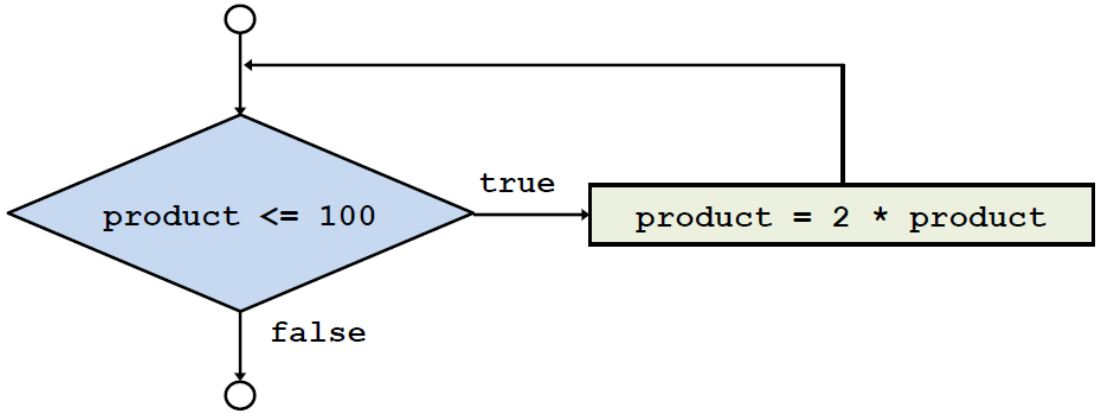


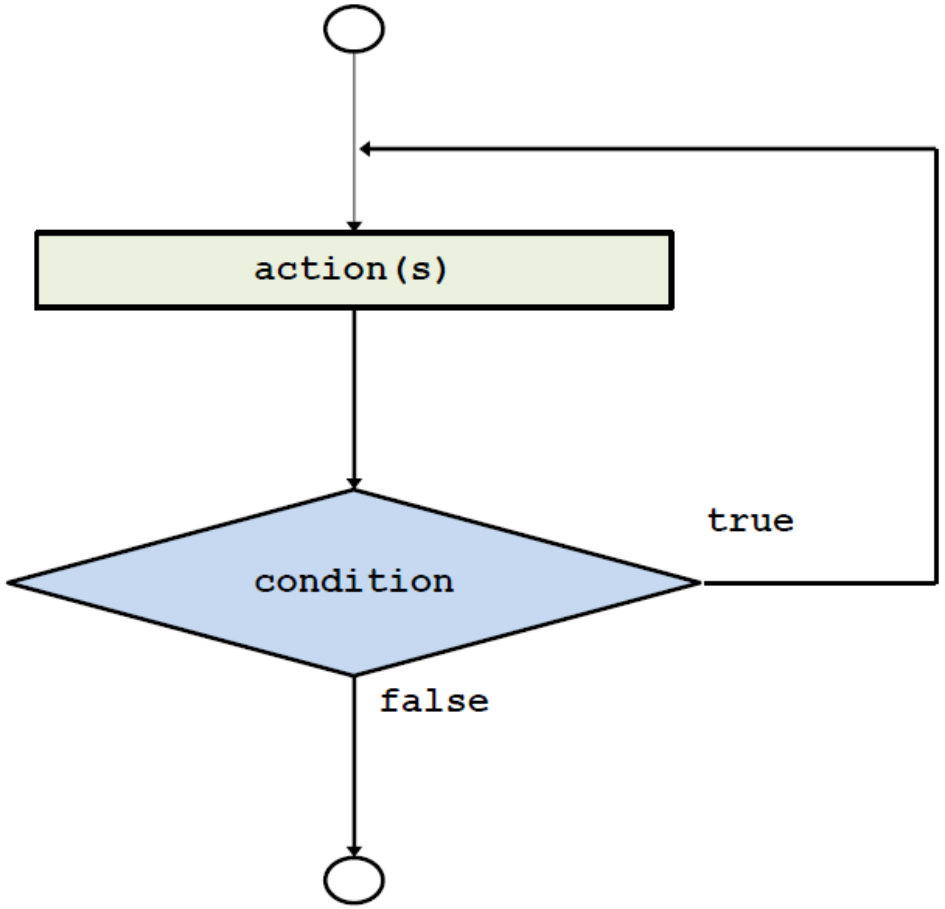
while & do-while

```
initialization;
while ( loopContinuationTest ) {
        statement;
        increment;
}

int product = 2;
while ( product <= 100 )
        product = 2 * product;
```

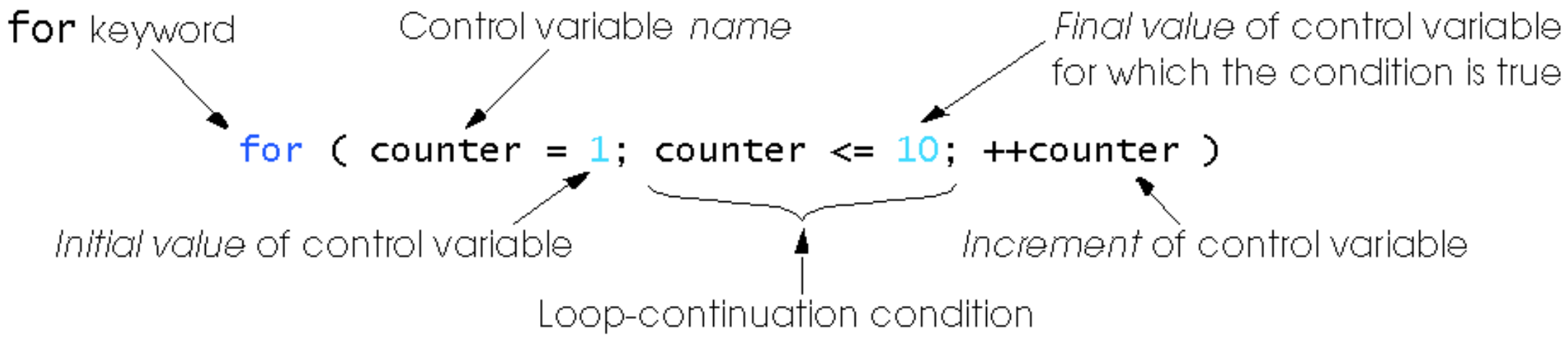
```
initialization;
do {
        statement;
} while ( condition );

int counter = 1;
do {
        printf( "%d ", counter );
        counter = counter + 1;
} while ( counter <= 10 );
```

Flowchart


product <= 100
true
product = 2 * product
false


action(s)
condition
true
false

for Repetition




```
for(counter = 1; counter <= 10; counter++ )
    printf( "%d\n", counter );
```

– Prints the integers from one to ten

Reminder: Arrays

- An array in C cannot grow or shrink —its size is fixed at the time of creation.
- Example

```
double pops[50];
pops[0] = 897934;
pops[1] = pops[0] + 11804445;
```

- Another way to make an array, if you know all the elements

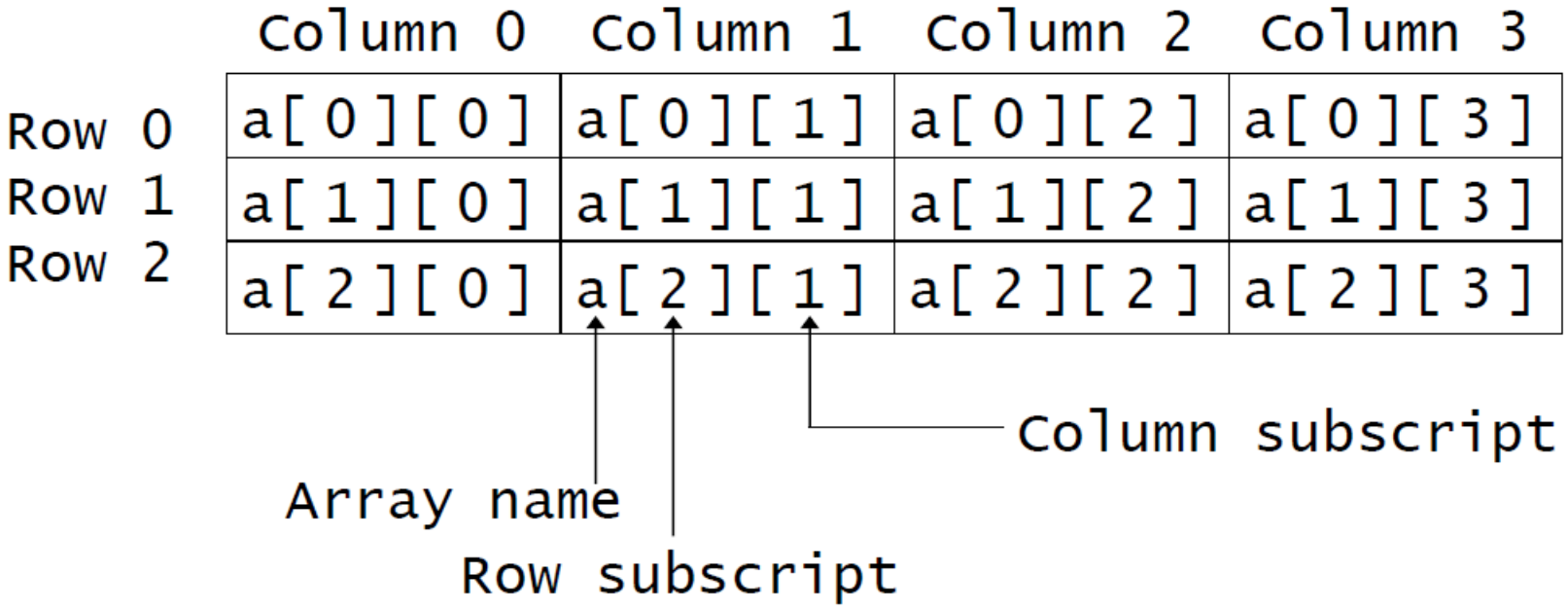
```
char vowels[6] = {'a', 'e', 'i', 'o', 'u', 'y'};
```

Array Exercise

- Write your first C program myFirstC.c that examines every elements stored in an array named **myArray** and returns the number of:
  - *Different element*
  - *Those divisible to 5*
  - *Those whose square root is equal to or greater than 4*
- Only 10 elements are allowed to be stored int **myArray**

Multi-Dimensional Arrays

- Multiple subscripted arrays
  - Tables with rows and columns (**m** by **n** array)
  - Like matrices: specify row, then column



Multi-Dimensional Arrays

- Initialization
  - **int b[2][2] = {{1,2},{3,4}};**
- Referencing elements
  - Specify row, then column

    printf("%d", b[0][1]);

Multi-Dimensional Arrays

- Takes a square matrix as input (NxN), contains only 1's and 0's

1	0	1	0	0	0	0
0	1	0	1	0	1	0
1	0	1	0	1	0	0
0	0	0	1	0	1	0
0	1	0	1	1	0	1
0	0	1	1	0	1	0
0	1	0	1	1	1	1

- Find a special pattern. X symbol, defined as 3x3 matrix

1	(0 or 1)	1
(0 or 1)	1	(0 or 1)
1	(0 or 1)	1

- Output

```
Pattern found at 1,1
Pattern found at 2,4
Pattern found at 4,4
Pattern found at 5,2
Pattern found at 5,5
```

Multi-Dimensional Array Exercise

- Takes a square matrix as input (NxN)
  - Find index of max value in matrix
  - Sum of elements on edge of matrix