

C LANGUAGE

A Short Course

Alvaro F. M. Azevedo

<http://www.fe.up.pt/~alvaro>

ANSI C

- Standard (ANSI, ISO)
- Compiled - efficient
- Low level / high level
- Other languages are based on the syntax of the C language
 - ◆ Example: C++, Java, C#, etc.

MY FIRST FULL PROGRAM

```
#include <stdio.h>

int main()
{
    printf("Hello world!\n");

    return 0;
}
```

♦ Recommended source file extension:  .c

Example: `my_first_test.c`

DECLARATIONS

```
/* This is a comment */
```

```
int k; /* k is a signed integer */
```

```
float x; /* x is a single precision real number */
```

```
double y; /* y is a double precision real number */
```

```
k = -3;
```

```
x = 8.19;
```

```
y = x / 7 + k;
```

```
printf("k = %4d, x = %6.2f, y = %12.7lf\n", k, x, y);
```



if ...

```
if (k > 3) printf("k is GREATER THAN 3\n");
```

```
if (k >= 3) printf("k is GREATER THAN OR EQUAL to 3\n");
```

```
if (k == 3) printf("k is EQUAL to 3\n");
```

```
if (k != 3) printf("k is NOT EQUAL to 3\n");
```

```
if (i < 0 && j < 0) printf("i AND j are negative\n");
```

```
if (i < 0 || j < 0) printf("i OR j is negative\n");
```

```
if ( ! (i < 0 && j < 0) ) printf("NOT(i < 0 and j < 0)\n");
```

if ...

```
int j = 6;
int kTest = 4;

if (kTest) /* 0 => FALSE; Not equal to 0 => TRUE */
{
    j++; /* j = j + 1 */
    printf("j = %d\n", j); /* Output: j = 7 */
}
else
{
    j--; /* j = j - 1 */
    printf("j = %d\n", j);
} /* if */
```

if ...

```
if (kTest > 0)
{
    j += 100; /* j = j + 100 */
    printf("j = %d\n", j);
}
else if (kTest < 0)
{
    j -= 100; /* j = j - 100 */
    printf("j = %d\n", j);
}
else
{
    printf("j unchanged\n");
} /* if */
```

OTHER COMPOUND OPERATORS

```
j *= i + 2; /* j = j * (i + 2) */
```

```
jNewCoeff /= a - b; /* jNewCoeff = jNewCoeff / (a - b) */
```

OTHER FORMAT SPECIFIERS

```
j = 37; printf("%5.5d\n", j); /* 00037 */
```

```
a = -0.00987; printf("%12.4le\n", a); /* -9.8700e-003 */
```

 Difference must be ≥ 8

INPUT

```
int myAge;

double todaysTemperature;

printf("How old are you? ");

scanf("%d", &myAge); /* Note the ampersand (&) */

printf("Tell me the temperature outside... ");

scanf("%lf", &todaysTemperature);

printf("I am %d years old and ", myAge);

printf("the temperature\noutside is %5.1lf degrees.\n",
      todaysTemperature);
```

ARRAYS

```
int v[3]; /* Array of 3 integers */  
  
v[0] = 8000; v[1] = 8001; v[2] = 8002;
```

```
double a[5][3]; /* 5x3 matrix of doubles */  
  
a[0][0] = 9.00; a[0][1] = 9.01; a[0][2] = 9.02;  
a[1][0] = 9.10; a[1][1] = 9.11; a[1][2] = 9.12;  
a[2][0] = 9.20; a[2][1] = 9.21; a[2][2] = -56.7952;  
a[3][0] = 9.30; a[3][1] = 9.31; a[3][2] = 9.32;  
a[4][0] = 9.40; a[4][1] = 9.41; a[4][2] = 9.42;
```

for LOOP

```
for (i = 1; i <= 6; i++)  
{  
    printf("%4d %4d\n", i, 100 * i);  
} /* for (i) */
```

```
for (i = -5, j = 1; j <= 12 && i != 2 * j; i -= 2, j += 3)  
{  
    printf("%4d %4d %4d\n", i, j, 2 * j);  
} /* for (i,j) */
```

for LOOP

```
/* Print a 5x3 matrix */
printf("Matrix a:\n");

for (i = 0; i < 5; i++)
{
    for (j = 0; j < 3; j++)
    {
        printf(" %10.4lf", a[i][j]);
    } /* for (j) */

    printf("\n");
} /* for (i) */
```

CHARACTERS

```
char gender; /* Male/Female => 'M'/'F' */

printf("Male (M) or Female (F) ? ");

gender = getchar();

if (gender != 'M' && gender != 'F')
{
    printf("ERROR!\n");
    exit(1); /* #include <stdlib.h> is required */
} /* if */

printf("The gender is: %c\n", gender);
```



STRINGS

```
/* A string is an array of char */
char t[4];
strcpy(t, "ABC"); /* #include <string.h> is required */

printf("t[0] = %3d\n", t[0]); /* 65 => ASCII code of A */
printf("t[1] = %3d\n", t[1]); /* 66 => ASCII code of B */
printf("t[2] = %3d\n", t[2]); /* 67 => ASCII code of C */
printf("t[3] = %3d\n", t[3]); /* 0 => All strings are
                               terminated by a NULL character. */

/* printf("t[4] = %3d\n", t[4]); => ERROR */

printf("The string is: %s\n", t); /* The string is: ABC */
```



POINTERS

```
double z; /* z is a double */
double* p; /* p is a pointer to a double */

z = 5.87; /* z is initialized with the value 5.87 */
p = &z; /* p is initialized with the address of z */

printf("z = %6.2lf\n", z); /* z = 5.87 */
printf("p = %d\n", p); /* p = 1245048 */
printf(">>> *p = %6.2lf\n", *p); /* >>> *p = 5.87 */
```

Notes:

& is the address-of operator

* is the indirection or dereferencing operator

FILES (input)

```
FILE* f; /* #include <stdio.h> is required */

f = fopen("my_file.txt", "r"); /* "r" means "read" */
if (f == NULL)
{
    fprintf(stderr, "File not found\n");
    exit(1); /* Program terminates */
} /* if */

fscanf(f, "%lf", &z); /* z must be declared as a double */

printf("z = %lg\n", z);

fclose(f); /* The file is closed */
```

FILES (output)

```
FILE* f; /* #include <stdio.h> is required */

f = fopen("my_file.txt", "w"); /* "w" means "write" */
if (f == NULL)
{
    fprintf(stderr, "Unable to open the output file\n");
    exit(1); /* Program terminates */
} /* if */

fprintf(f, "z = %6.2lf\n", z); /* z is a double */

fclose(f); /* The file is closed */
```

break AND continue

```
int i, j;

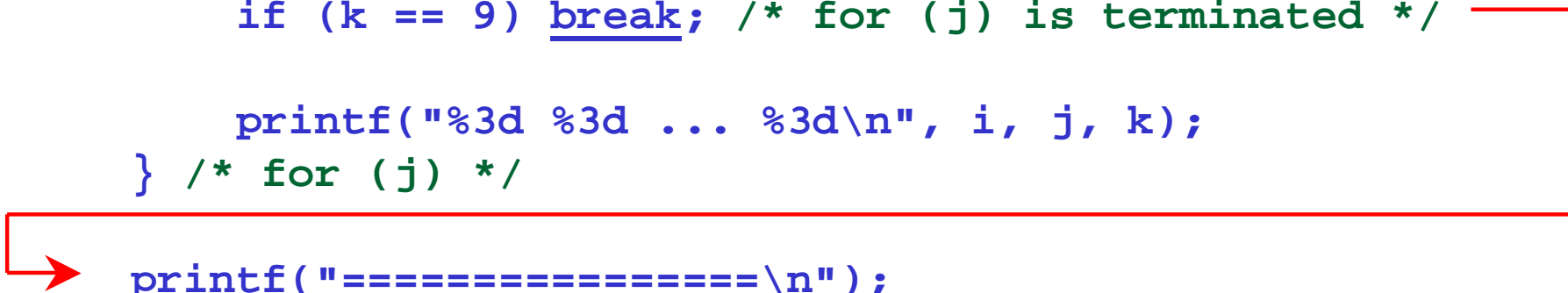
for (i = 20; i <= 40; i++)
{
    j = i / 6; /* Integer division */
    if (j == 5) break; /* for loop is terminated */
    printf("%2d %2d\n", i, j);
} /* for (i) */
```

```
for (i = 20; i <= 40; i++)
{
    j = i % 6; /* Remainder of the integer division */
    if (j == 0) continue; /* "goto" the beginning */
    printf("%3d %3d\n", i, j);
} /* for (i) */
```

break WITH NESTED FOR LOOPS

```
for (i = 1; i <= 8; i++)
{
    for (j = 1; j <= 6; j++)
    {
        k = i + 2 * j;
        if (k == 9) break; /* for (j) is terminated */

        printf("%3d %3d ... %3d\n", i, j, k);
    } /* for (j) */
} /* for (i) */
```



```
printf("=====\n");
```

OTHER STATEMENTS

```
for (;;) /* Infinite loop (forever) */
{
    ...
    if (condition) break;
    ...
} /* for (;;) */
```

```
while (condition)
{
    ...
} /* while */
```

```
do
{
    ...
}
while (condition);
```

switch

```
switch (nChoice)
{
case 1:
    ... /* Executed when nChoice == 1 */
    break;

case 5:
case 8:
    ... /* Executed when nChoice == 5 || nChoice == 8 */
    break;

default:
    fprintf(stderr, "Valid choices are: 1, 5 or 8\n");
    exit(1);

} /* switch (nChoice) */
```

#define AND const

```
#define PI 3.14159265359 /* Simple text replacement */
```



```
const double PI = 3.14159265359; /* PI is a const variable */
```

sscanf AND sprintf

```
char t[20];
```

```
double x = -2.87;
```

```
double y;
```

```
sprintf(t, "%6.2lf", x); /* Output goes to the string t */
```

```
strcat(t, "654"); /* String concatenation */
```

```
sscanf(t, "%lf", &y); /* Input from the string t */
```

```
printf("y = %12.8lf\n", y); /* y = -2.87654000 */
```

malloc AND free

```
#include <stdlib.h> /* malloc, free */
...
int i, n;
double* vect;
printf("n ? "); scanf("%d", &n);

/* Allocate memory: */
vect = (double*)malloc(n * sizeof(double));

for (i = 0; i < n; i++)
{
    vect[i] = (double)i / 3; /* i is converted into a double */
}
...
free(vect); /* Free memory */
```


FUNCTIONS

 → Memory address

```
char calc(int k, double x, double* result )
{
    k -= 3; /* Modifying a copy of the original k */
    if (k == 0) return 1;
    *result = x / k;
    printf("Function calc: k = %d\n", k);
    return 0;
}
int main()
{
    int k = 8; double x = 5.55; double result; char c;
    c = calc(k, x, &result ); /* c == 0 indicates success */
    if (c == 0) printf("result = %lg\n", result);
    printf("main: k = %d\n", k); /* k = 8 (unchanged) */

    return 0;
}
```

argc AND argv

```
int main(int argc, char* argv[])
{
    /* argc is the n. of command line arguments */
    if (argc != 3) /* The program name is also counted */
    {
        fprintf(stderr, "When this program is executed,\n");
        fprintf(stderr, "two arguments must follow the\n");
        fprintf(stderr, "program name.\n\n");
        fprintf(stderr, "Usage:  show_args  arg1  arg2\n");
        exit(1);
    } /* if */

    printf("argv[0] = %s\n", argv[0]);
    printf("argv[1] = %s\n", argv[1]);
    printf("argv[2] = %s\n", argv[2]);

    return 0;
}
```

Note: save this file as
"show_args.c"

struct

```
struct Point
{
    double x;
    double y;
    int size;
    char color;
}; /* Semicolon (;) required */

int main()
{
    struct Point A, B;
    A.x = 70.1; A.y = 70.2; A.size = 5; A.color = 'R';
    B = A;
    printf("B = ( %lg , %lg ) -> %d/%c\n",
           B.x, B.y, B.size, B.color);
    return 0;
}
```

struct AS A FUNCTION ARGUMENT

```
void PrintPoint(char* description, struct Point K)
{
    /* Copies of the data members of */
    /* the struct are printed. */
    printf("\n%s: ( %lg , %lg ) -> %d/%c\n",
           description, K.x, K.y, K.size, K.color);
}

int main()
{
    struct Point A, B;
    A.x = 70.1; A.y = 70.2; A.size = 5; A.color = 'R';
    B = A;
    PrintPoint("Point A", A);
    PrintPoint("Point B", B);
    return 0;
}
```

struct MODIFIED BY A FUNCTION

```
void ReadPoint(char* description, struct Point* pK)
{
    printf("\n%s (x,y,size,color)\n", description);
    printf("x ? ");          scanf("%lf", &pK->x);
    printf("y ? ");          scanf("%lf", &pK->y);
    printf("size ? ");       scanf("%d", &pK->size);
    printf("color [1-5] ? "); scanf("%d", &pK->color);
    pK->color += 64; /* [1-5] -> [A-E] */
}
int main()
{
    struct Point A, B;
    ReadPoint("Point A ?", &A);
    ReadPoint("Point B ?", &B);
    PrintPoint(">>> Point A", A);
    PrintPoint(">>> Point B", B);
    return 0;
}
```

struct CONTAINING AN ARRAY

```
struct Vector
{
    int size;
    double* vect;
};
int main()
{
    struct Vector A, B;
    A.size = 3; A.vect = (double*)malloc(3 * sizeof(double));
    A.vect[0] = 7.00; A.vect[1] = 7.01; A.vect[2] = 7.02;
    B = A;
    printf("A => %d, %8.2lf\n", A.size, A.vect[1]);
    printf("B => %d, %8.2lf\n", B.size, B.vect[1]);
    A.vect[1] = 5555.55; /* Why was this ignored ? */
    B.vect[1] = 8888.88;
    printf("A => %d, %8.2lf\n", A.size, A.vect[1]);
    printf("B => %d, %8.2lf\n", B.size, B.vect[1]);
    ...
}
```

NEXT CHAPTER:

C++

...