

**COMP2300**

- [Home](#)
- [StudyAt](#)
- [Assessment](#)
- [Schedule](#)
- [Lecture Notes](#)
- [Tute / Labs](#)
- [Assignments](#)
- [Announcements](#)
- [Discussion](#)
- [Reading material](#)
- [Old Exams](#)
- [Links](#)
- [PeANUt@Home](#)
- [Getting Help](#)

Quick Links

- [ANU Home](#)
- [FEIT Home](#)
- [DCS Home](#)
- [Search](#)



Notes for Tutorial/Laboratory Session 03: C I/O, Pointers and Structures

Semester 1, 2009

Week 4 (16 - 20 March)

As well as the preparation exercises required for this session, you should read through the tutorial exercises beforehand, and revise any relevant lecture notes and/or sections of your text book before your session (this in fact applies to all sessions). Note also that there is submittable work which is due by 00 am Tuesday 24 March (week 5), which will contribute up to 1% of your assessment (in the Tute/Lab mark).

Preparation Exercises

Complete the following questions on a separate sheet of paper, with your name and student number clearly written. Please ensure your writing is legible. Hand in to your tutor at the *beginning of your tutorial / laboratory session*. The following questions should serve as revision for you C programming from the previous session.

1. Write a statement that declares an integer x and initializes it to 0.
2. Write a statement to assign the integer variable x that value of the integer variable y plus 1.
3. Write a statement to print out the value of the integer variable x.
4. Write a statement to scan the read an integer value from standard input into the integer variable x.
5. Write a for loop which prints out the values 1 to 10 on separate lines.
6. Write a while loop which does the same as the above.
7. Write a if-else statement which assigns the integer variable y to the absolute value of x.
8. Write a statement to declare in integer array a which holds 4 elements.
9. Write a for loop to set the ith element in the array a above to have the value two times i.
10. Write a function which takes a single string parameter and prints it out.

Tutorial Questions

Work through the following questions in your session. Finish any uncompleted exercises for homework.

1. Consider the declaration `int a[8], b[4], *c;`. How many bytes of memory would be allocated by this declaration on a 32-bit machine? Draw a diagram to illustrate this. Would the assignment statement `b = a;` be legal, and if so describe what effect it would have. How about `c = a;`?
2. **Pointers:** true or false? (and why?)
 1. A variable in a C program, regardless of its type, must have an address.
 2. A variable of any type can be used to store the address of a variable.
 3. The address of a variable is expressed in hexadecimal form; therefore, only an integer type variable can be used to store the address of a variable.
 4. A pointer variable of any type can be used to store the address of a double variable.
 5. The indirection operator `*` may appear only on the left side of an assignment statement.
 6. The indirection operator `*` may appear on both sides of an assignment statement.
 7. The address of operator `&` may appear only on the left side of an

assignment statement.

8. The address of operator & may appear on both sides of an assignment statement.
 9. For an int variable aa, its address &aa is a constant, not a variable.
 10. Given char aa, *bb; and sizeof(aa) is 1 byte, then sizeof(bb) is also 1 byte.
3. Consider the following code:

```
#include <stdio.h>
int main(void) {

    char str[80];
    char *p;
    int i;

    printf("Enter a string:\n");
    scanf("%80s", str);
    p = str;

    while(*p) *p++ -= 32;
    printf("%s\n", str);

    return 0;
}
```

Rewrite it (on paper) using array notation instead of pointers, i.e. the str[i] notation. Now state what the program does.

What function from <ctype.h> could you use to achieve the same effect as p++ -= 32; ?

4. In the following code:

```
#include <stdio.h>
void funct(int *p);

int main(void) {
    int a[] = {4, 8, 10, 7, 9, -6, 3, 0, 7, 22, 1};

    funct(a+1);
    funct(a+2);
    funct(a+8);

    return 0;
}

void funct(int *p) {
    int x = 0, y = 0;
    int *pstart, *pend;
    pstart = p;

    while (*p >= 0) {
        x += *p++;
        y++;
    }

    pend = p;
    printf("y=%d x=%d pend-pstart=%d\n", y, x, pend-pstart);
}
```

1. What kind of argument is passed to funct()?
2. What kind of information is returned by funct()?
3. What information is actually passed to funct()?
4. What is the purpose of the while loop that appears within funct()?
5. What values are displayed by the printf() statement within funct()?

Laboratory Exercises