Singapore Institute of Management
BSc in Computing & Information Systems
CS 210 Tutorial 11

---

1. What are the various categories of faults? Briefly describe with examples.

Answer :

- *Boundaries*
  - *Faults cluster near or on the boundaries imposed on the input domain by the input output relation.*
  - *E.g. The actual condition required "x<=y", however, the condition typed was "x<y". This means that the condition "x=y" is not captured and becomes a fault due to boundary condition*

- *Initialisation*
  - *Forgetting to initialize the value of a variable may/may not lead to error.*
  - *May not lead to error as the random value obtained form the uninitialised variable will allow the system to pass all our test cases.*
  - *For e.g. the variable "x" should be set to 0. However, as the variable was not initialized, the initial value could be set to "1". For a condition "if x < y" where y is set to 1, the condition will not be true and so the subsequent then statement is not executed.*

- *Scope*
  - *Name-scope clashes often occur when a program has global and local name spaces which becomes confused. Code in which a reference parameter shares the name of a global variable and in which the global is passed for the reference parameter exhibits this form of potential confusion*
  - *E.g. would be there is 2 variables "realValue" one of which is a global variable and another which is local to module "find-real-value". When setting the value of "realValue in the module, it is ambiguous as to which variable is being set.*

2. What is the recommended process for following when bugs are found?

Answer :

- *Avoid patching your way out of the bug and don't change the code at random.*
- *Once fault has been located and fix has been approved, the code must be altered.*
- *Keep an old version of the code which contains the bug and the i/o relation that demonstrates its presence. This allows you to go back to a previous version of the program if it turns out that your correction has introduced so many harmful ripple effects.*
- *Record time and date that bug was found, and the author of the change as a comment in the new version of the code.*
- *Record the location of the file containing the previous version of the code.*
- *Agree on standard format for this kind of comment with your colleagues and co-workers.*

3. What are the components of faults? Give examples to support your answer.

Answer :

- *Technical basis*
  - *Technical reason for the failure and the fault or collection of faults which directly lead to system failure*
  - *Pentium processor bug – where the computation of the following 4195835 – (4195835 / 3145727) * 3145727) was 256 instead of 0. This was due to technical error as a table of values was used in computation to speed up the process.*

- *Organisational basis*
  - *Damage not only caused by technical basis. It is exacerbated or even caused by organizational basis. The occurrence and severity of failure is thus conditioned by the organizational envt in which the software is developed and used*
  - *The organizational issue due to the above mentioned technical error was that Intel refused to replace the chips unless users proved they needed a replacement. PR disaster. Caused dissatisfaction among users who paid a lot of money for chip.*
  - *Intel estimated that normal user would encounter error once every 27000 years.*
  - *Public perception of Intel's conclusion that users needed to demonstrate need for upgrade was not favourable. Failed to recognized the potential of organizational failure. Public believed that had a faulty product and were entitled to replacement.*

4. What is assertion-based programming? When would this type of programming be useful?

Answer :

Assertion-based programming Is one approach to trapping defects which has proved successful. It is a basis for self-testing code. The intended behaviour of a part of a program is described using logical notation and in the popular version of the approach, these assertions are inserted as program code. The assertions are normally expressed in the form of pre and post conditions.

For e.g.

```
 I=0;
While (A[i] != Findme) i++;
Return I;
```

Suitable pre-condition will be There exist i.A[i} = Findme
Suitable post-condition will be A[i] = Findme

Assertion programming is useful

- *to check the intuition behind programming*
- *to Act as a link between code and specification and as a form of documentation.*
- *Is Useful to strengthen the robustness of the system*

5. The following is a sample of C code. Identify the faults and show how to rectify the problems.

```
#include <stdio.h>
#include<string.h>

#define MAX_A 10
#define MAX LEN 40
#define NBR_MEDALS 3

struct athlete {
   char name [MAX_LEN + 1];
   float jump1;                    /* his/her first jump in decimal */
   float jump2;                    /* his/her second jump in decimal */
   float jump3;                    /* his/her third jump in decimal */
   float max_jump;                 /* his/her highest jump in decimal
}
struct athlete *na;

void Init(struct athlete *na)      /*function to initialize athlete's information*/
int GetInput (struct athlete *na); /* function to Get the input and store them */
void CalcHigh(struct athlete *na, int nbr);   /* function to calculate the best jump among the 3
                                                 jumps*/


void SortHeight (struct athlete na, int nbr); /* function to sort them by the best jump */
void PrintResults(struct athlete *na);        /* function to print results */

main()
{
   struct athlete nameath[MAX_A];
   float nbr;
   char sTemp[81];

   na = &nameath [0];              /* points to first element in the array */
   nbr = GetInput(na);
   CalcHigh(na, nbr);              /* call to calculate best of 3 jumps */
   SortHeight(na, nbr);            /* call to sort them by descending order */
   PrintResults;                   /* call to print the final results */
   Gets(sTemp);
}

void Init(struct athlete *na)
{
   for (j=0;j<=MAX_A;j++,na++) {
            *(na>name) = '\0';
            na->jump1 = 0.0;
            na->jump2 = 0.0;
            na->jump3 = 0.0;
            na->maxjump = 0.0;
   }
}
```

Answer :

```
#include <stdio.h>
#include<string.h>

#define MAX_A 10
#define MAX LEN 40
#define NBR_MEDALS 3

struct athlete {
    char name [MAX_LEN + 1];
    float jump1;                    /* his/her first jump in decimal */
    float jump2;                    /* his/her second jump in decimal */
    float jump3;                    /* his/her third jump in decimal */
    float max_jump;                 /* his/her highest jump in decimal  */
}

void Init(struct athlete *na)       /*function to initialize athlete's information*/
int GetInput (struct athlete *na);  /* function to Get the input and store them */
void CalcHigh(struct athlete *na, int nbr);   /* function to calculate the best jump among the 3
                                                 jumps*/
struct athlete na;          /* so that not global variable */


void SortHeight (struct athlete na, int nbr); /* function to sort them by the best jump */
void PrintResults(struct athlete *na);        /* function to print results */

main()
{
    struct athlete nameath[MAX_A];
    float int nbr;
    char sTemp[81];
    struct athlete *na;

    na = &nameath [0];              /* points to first element in the array */
    Init(na);                       /* insert to initialize the athlete variable */
    nbr = GetInput(na);
    CalcHigh(na, nbr);              /* call to calculate best of 3 jumps */
    SortHeight(na, nbr);            /* call to sort them by descending order */
    PrintResults (na);                      /* call to print the final results */
    Gets(sTemp);
}

void Init(struct athlete *na)
{
    for (j=0; j<MAX_A; j++,na++) {
            *(na>name) = '\0';
            na->jump1 = 0.0;
            na->jump2 = 0.0;
            na->jump3 = 0.0;
            na->maxjump = 0.0;
    }
}
```