Microbiol. Cult. Coll. Dec. 2000. p. 71-74 Vol. 16, No. 2

A Simple Program for Summarizing the Viability of Stored Resources

Toshirou Nagai*

National Institute of Agrobiological Resources, The Ministry of Agriculture, Forestry and Fisheries, Tsukuba, Ibaraki 305-8602, Japan

Many isolated cells are subjected to long-term storage in many institutes, and they are tested for viability periodically. Viability data is a good reference for preserving strains to be tested. The program described here facilitates such testing of preservation by tabulating it. The software : seizan.exe, is compatible with the Windows 95/98 operating system, and it can be downloaded from a WWW page (http : //www.gene.affrc.go.jp/micro/). This program was able to process data of 14,600 records in 2 seconds using a computer that had a 350 MHz Pentium II processor.

Key words : culture collection, preservation, software, viability

In culture collections, many cells are stored using customized methods (2), and the stored organisms are tested for viability periodically. The viability of cells after long-term storage gives a good reference for selecting a method for preserving an organism to be tested. Viability itself, however, cannot easily show how long an organism can be stored ; therefore, it is necessary to summarize the viability of stored resources as a table.

At the same time, the development of computer technology has given us powerful tools for programming on personal computers. Many compilers are now available on personal computers that allow us to build applications to help maintain stored resources.

This report describes a simple program that summarizes viability data of stored organisms, and that constructs a table showing the number of strains tested and strains viable after storage for each species. The table is output as a CSV file (Comma Separated Value, a commonly used format for spreadsheet files) or a text file. The program is simple to understand and modify.

The program described here (named seizan.exe ; seizan means "survival" in Japanese) is coded in the programming language C++ (3) and compiled with C++ Builder (Inprise Corporation) on the Windows 95/98 operating system (Microsoft Corporation). The program can be downloaded from a WWW page (http : //www.gene.affrc.go.jp/micro/).

Although the program seizan.exe is composed of several functions, this report describes only the GetState() function, which examines whether a strain is viable after a given storage period. The code of the GetState() function and a test program for it are presented in Fig. 1, and the meanings of variables used in the code are shown graphically in Fig. 2.

Before calling the GetState() function, data of viability of each strain is searched for the longest period of successful preservation (the variable alivePeriod in the code), and then, among periods longer than alivePeriod, the minimum value for deadPeriod is sought. Long-term storage over deadPeriod means serious impairment of the stored strain. Strains that pass all tests for viability have no value of deadPeriod, and those that pass none of the tests have no alivePeriod. In these cases, a negative value, e.g., −1, is assigned to each parameter.

The state of a strain after a certain period of storage, whose length is passed as a parameter : storagePeriod, is determined by calling the GetState() function. The value of storagePeriod can have a

* Correspondence author
E-mail : nag@affrc.go.jp

```
#include <iostream>

using namespace std;

enum State {UNKNOWN, DEAD, ALIVE};

State GetState(double lowerPeriod, double upperPeriod, double alivePeriod, double deadPeriod)
{
    if (alivePeriod >= 0) {
        if (deadPeriod >= 0) {
            // Case 1: Values for both alivePeriod and deadPeriod exist.
            if (lowerPeriod <= alivePeriod) return ALIVE;
            return ((upperPeriod >= deadPeriod) ? DEAD : UNKNOWN);        }
        else {
            // Case 2: A value exists only for alivePeriod.
            return ((lowerPeriod <= alivePeriod) ? ALIVE : UNKNOWN);
        }
    }
    else if (deadPeriod >= 0) {
            // Case 3: A value exists only for deadPeriod.
               return ((upperPeriod >= deadPeriod) ? DEAD : UNKNOWN);
          }
    else
            // Case 4: Values for neither alivePeriod nor deadPeriod exist.
        throw ("Invalid");
}

// A test program for the GetState() function
int main(void)
{
    try {
        double storagePeriod, margin;
        double alivePeriod, deadPeriod;
        char   cont;

        cout << "Input a storage period :" << endl;
        cin  >> storagePeriod;
        cout << "Input its margin :" << endl;
        cin  >> margin;
        double upperPeriod  = storagePeriod * (1 + margin / 100);
        double lowerPeriod  = storagePeriod * (1 - margin / 100);

        do {
            cout << "Input alivePeriod :" << endl;
            cin  >> alivePeriod;
            cout << "Input deadPeriod :"<< endl;
            cin  >> deadPeriod;

            switch (GetState(lowerPeriod, upperPeriod, alivePeriod, deadPeriod)) {
                case ALIVE:
                    cout << "ALIVE" << endl;
                    break;
                case DEAD:
                    cout << "DEAD" << endl;
                    break;
                default:
                    cout << "UNKNOWN" <<endl;
            }

            cout << " Continue? (Y/N):" << endl;
            cin  >> cont;
        } while (cont == 'Y' || cont == 'y');

        return 0;
    }
    catch (const char* message) {
        cout << "Invalid" <<endl;
        return 1;
    }
}
```

Fig. 1. The GetState() function and a test program

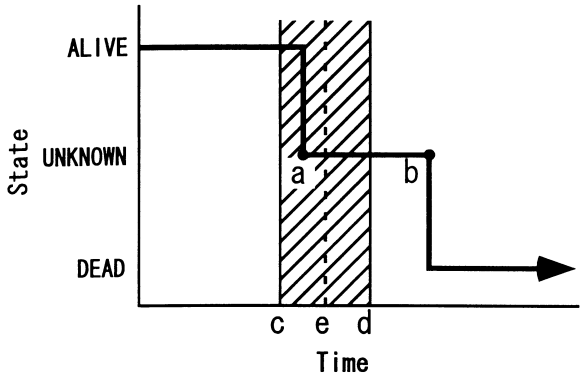

Fig. 2. Variables used in the program

A typical change in the state of a stored organism during preservation is shown. The hashed area indicates a range, within which the state of an organism after a given period of preservation is determined. a, alivePeriod ; b, deadPeriod ; c, lowerPeriod ; d, upperPeriod ; e, storagePeriod. Values for lowerPeriod and upperPeriod are calculated using values for variables storagePeriod and margin (see text), which are given by the user.

margin, whose size is passed to the GetState() function as the parameter *margin*. A value of storagePeriod of 10 months with a margin of 20 % represents a period ranging from 8 to 12 months. In the GetState() function, the state of a stored strain is determined by using the algorithm "If the value of storagePeriod is larger than that of deadPeriod, the strain is not alive. Conversely, if the value of storagePeriod is smaller than that of alivePeriod, the strain is alive. If neither case applies, the state of the strain cannot be determined." Codes for this algorithm are shown in Fig. 1, in which data are processed following the cases below :

Case 1 : Values for both alivePeriod and deadPeriod exist.

Case 2 : A value exists only for alivePeriod.

Case 3 : A value exists only for deadPeriod.

Case 4 : Values for neither alivePeriod nor deadPeriod exist. This means an error has occurred somewhere.

Next, we will briefly examine how to use seizan.exe. Input data need to be described in a text file according to a fixed form : *species name*$*strain number*$*date for starting storage*$*date for viability test*$*viability*. Each field is separated by a separator code, such as "$" employed in the Genetic Resources

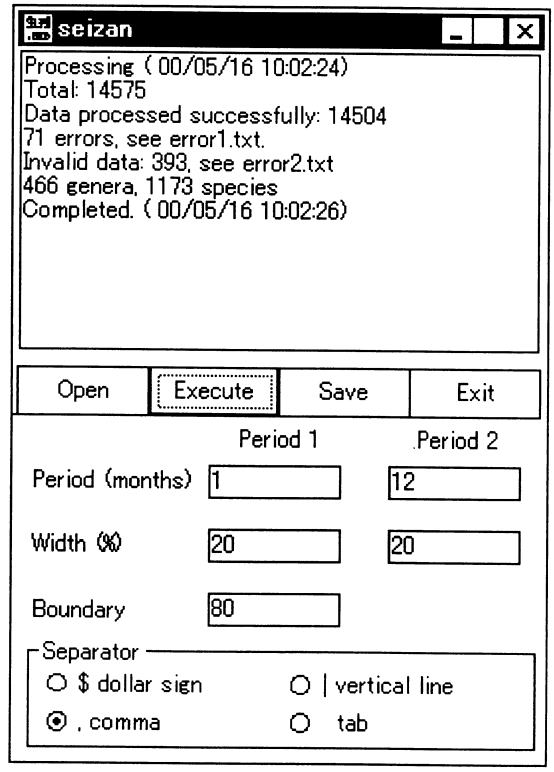

Fig. 3. Sample display of seizan.exe

Center for microorganisms of the Ministry of Agriculture, Forestry and Fisheries (MAFF). A CSV file is available. The date is in the form of *year/month/day* (e.g., 1999/10/2). Note that 99/10/2 is not interpreted as 1999/10/2. The data to be processed should not be mixed with the data from experiments using any other preservation methods.

There are some buttons and check boxes on the form of seizan.exe (Fig. 3).

The buttons are :

Open : to open a data file.

Execute : to start processing data.

Save : to save a table of viability after data processing.

Exit : to end the program.

The editable boxes and the radio box are :

Period : to set a storage period.

Margin : to set a margin of the storage period, expressed as a percentage of the storage period.

Boundary : to set a minimum viable level as viability. Having viability over the boundary is defined as being alive after storage.

Separator : to set a separator used in input data.

The program is used as follows : 1) run seizan.

exe, 2) click the "Open" button and select a data file, 3) set parameters, 4) click the "Execute" button, 5) click the "Save" button and save the result, and 6) click the "Exit" button to terminate seizan.exe. Details of errors occurring during data processing are recorded in error log files in the current directory.

An example of input data and a table of processed data, which are output in a file, is shown in Fig. 4. The data concerned the viability of fungal strains stored in the MAFF Genebank by freezing preservation in an atmosphere over liquid nitrogen (1), and consisted of 14,600 records. The program seizan.exe processed the data in 2 seconds on a computer with a 350 MHz Pentium II CPU. The complete list is published elsewhere (4).

A

```
Abortiporus biennis$420311$94/12/08$95/01/30$100
Abortiporus biennis$420311$94/12/08$95/12/04$100
Achlya americana$305720$96/04/18$97/04/10$0
                    ●
                    ●
                    ●
Xylobolus frustulatus$420277$93/10/22$94/10/26$100
Xylobolus spectabilis$420297$93/10/13$93/11/25$100
Xylobolus spectabilis$420297$93/10/13$94/10/17$100
```

B

Abortiporus biennis	2	2	2
Absidia corymbifera	1	1	1
Absidia macrospora	1	1	1
●			
●			
●			
Xylobolus frustulatus	1	1	1
Xylobolus spectabilis	1	1	1
Total	5569	5490	5406

Fig. 4. Input data for seizan.exe and tabulated results

A) input data, B) tabulated results. The columns show species name, the number of strains tested, surviving strains after storage period 1, and surviving strains after storage period 2.

REFERENCES

1. Miki, N. and Kubomura, Y. Preliminary trials of frozen-preservation of fungi for long-term storage. Misc. Publ. Natl. Inst. Agrobiol. Resour. **5**: 1-29 (1993) (in Japanese).
2. Sakai, A.(ed.). Touketsu Hozon. Asakura Shoten, Tokyo (1987) (in Japanese).
3. Stroustrup, B. The C++ programming language. 3rd ed., Addison-Wesley, Mass. (1997).
4. Nagai, T., Ideno, A., Tsuge, M., Oyanagi, C., Oniki, M., Kita, K., Horita, M., Aoki, T., Kobayashi, T. and Tsuchiya, K. Preservation of fungi in an atmosphere over liquid nitrogen after uncontrolled freezing. Microbiol. Cult. Coll. **16**: 13-22 (2000).

生物資源の生残結果要約プログラム

永井利郎

農業生物資源研究所遺伝資源第二部

菌株の保存方法を選択する際の指標として，菌株の保存後の生残数をまとめた表は有用である．本報告ではある期間保存した後の生残菌株数をとりまとめる表を作成するソフトウェア seizan.exe の作成とその使用法について述べた．本ソフトウェアは Windows 95/98 をオペレーティングシステムとしているパーソナルコンピュータで稼働し，インターネット (http：//www.gene.affrc.go.jp/micro/) を通じて公開されている．Pentium II (350 MHz) プロセッサを実装するコンピュータでは，本プログラムは 2 秒で 14,600 件のデータを処理することができた．