

FISH MANAGEMENT REPORT 124

April 1985

Bureau of Fish Management • Wisconsin Department of Natural Resources, Madison, Wisconsin

A COMPUTER PROGRAM SYSTEM
TO ALLOCATE THE ANNUAL STOCKING
OF SALMONIDS IN THE WISCONSIN
WATERS OF LAKE MICHIGAN

by Terrence R. Dehring and Charles C. Krueger

ABSTRACT

A system of computer programs is described that allocates the annual stocking of brook, brown and rainbow trout, and coho and chinook salmon in the Wisconsin waters of Lake Michigan. The program is written in Apple Pascal and operates on an Apple II+ or IIe microcomputer. The user enters data on the keyboard in response to questions by the program. Sociological, physical, and ecological parameters are used to compute (1) the recommended number and size of each species to stock by management zone, (2) the predicted catch by species and fishery within zones, and (3) the predicted costs involved in propagation and distribution of recommended stocking. Output is available on a video screen or in printed form.

CONTENTS

INTRODUCTION	3
METHODS	4
PROGRAM DESCRIPTION	4
EXAMPLE OPERATION	13
LITERATURE CITED	19
APPENDIX 1 - Program listings	21
APPENDIX 2 - Program structure by program name and procedure	59
APPENDIX 3 - Contents and size of the programs and files included on the diskettes	65
APPENDIX 4 - Procedure for making program changes	69
APPENDIX 5 - Flow diagram of main computational program (Rationale)	75

"APPLE COMPUTER, INC. AND THE REGENTS OF THE UNIVERSITY OF CALIFORNIA MAKE NO WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING THE ENCLOSED COMPUTER SOFTWARE PACKAGE, ITS MERCHANTABILITY OR ITS FITNESS FOR ANY PARTICULAR PURPOSE. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME STATES. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY PROVIDES YOU WITH SPECIFIC LEGAL RIGHTS. THERE MAY BE OTHER RIGHTS THAT YOU MAY HAVE WHICH VARY FROM STATE TO STATE."

SYSTEM.APPLE, SYSTEM.PASCAL, SYSTEM.MISCINFO, and SYSTEM.LIBRARY are copyrighted programs of Apple Computer, Inc. licensed to Terrence R. Dehring and/or Charles C. Krueger to distribute for use only in combination with the Wisconsin DNR Lake Michigan Stocking Allocation Program. Apple Software shall not be copied onto another diskette (except for archive purposes) or into memory unless as part of the execution of the Wisconsin DNR Lake Michigan Stocking Allocation Program. When the Wisconsin DNR Lake Michigan Stocking Allocation Program has completed execution, Apple Software shall not be used by any other program.

INTRODUCTION

The Wisconsin Department of Natural Resources (WDNR) has been actively stocking salmonids in Lake Michigan since 1963. Rainbow trout (Salmo gairdneri), brown trout (Salmo trutta), brook trout (Salvelinus fontinalis), lake trout (Salvelinus namaycush namaycush), coho salmon (Oncorhynchus kisutch), and chinook salmon (Oncorhynchus tshawytscha) have been stocked to provide a sport fishery as well as to provide predatory controls for an abundant alewife population. Stocking has been required to maintain the fishery because natural reproduction of the salmonids does not occur to any appreciable extent. Salmonid stocking has steadily increased since the 1960s to the current annual levels of 5.5 to 7.5 million fish. These fish are distributed along the entire Lake Michigan shoreline in Wisconsin from Marinette to Kenosha.

The dependence of the fishery on stocking presents an unusual management opportunity to potentially impact the fisheries in local areas by altering the numbers and species stocked each year. In Wisconsin, stocking decisions have been made on an informal basis by local fishery managers within the WDNR. Decisions about stocking were intended to optimize angling opportunities in the trolling, pier, shore, and stream fisheries. As the fishery expanded, however, the complexity of the variables that impact stocking decisions were similarly increased. This increased complexity resulted in a need to identify formal procedures for developing stocking recommendations for Lake Michigan salmonids (e.g., Schultz 1979).

In response to this need, Krueger and Dehring (in prep.) described a process to develop annual salmonid stocking recommendations for geographically specific zones in the Wisconsin waters of Lake Michigan. The allocation process uses sociological parameters (catch objectives, angler species preferences), physical parameters (availability of fishery facilities such as number of boat ramps, length of streams, etc.), and ecological parameters (predator pressure on the forage base) to develop stocking recommendations. The procedure was designed to optimize angler catch for all species except lake trout. Lake trout stocking recommendations were determined by a separate procedure and were based on the objective of population rehabilitation (Krueger and Dehring, in prep.).

In this paper, a computer program is described that performs the salmonid stocking procedure described by Krueger and Dehring (in prep.). This program is interactive with the user and requires information about a catch objective, angler species preferences, number of lake trout available for stocking, and the average weights of coho and chinook salmon in the past two years. Output includes tables of recommended stocking by species within management zones; predicted catch by species within zones; predicted catch by species for each fishery; projected cost of the recommended stocking; and the recommended lake trout stocking by zone. The program described in this document is available on magnetic medium (5 1/4-inch floppy diskette) from the authors at cost.

This paper should be used in conjunction with the document (Krueger and Dehring, in prep.) that describes the procedure and should not be relied upon to provide the computational details used in the stocking rationale procedure. The computer model and how it implements the procedure will be discussed here, rather than the procedure itself.

METHODS

The computer version of the stocking procedure consists of eight programs and six data files (Append. 1 and 2). The computer programs were written in Apple Pascal and compiled using an Apple Pascal System ver. 1.1 software package (Apple Computer, Inc. 1980). The compiled versions of the programs and the data files are on one diskette (STOCK:), the non-compiled programs are on another diskette (STOCK1:) (Append. 3). Although there are two diskettes in the model package, only one diskette (STOCK:) is needed to actually run the stocking rationale program (see Append. 4 for details). The second diskette (STOCK1:) is needed when making program enhancements or minor modifications (Append. 5).

The programs were written to operate on an Apple II+ or IIe computer with at least 48K RAM and a language board in slot #0. An 80 column integrated circuit card in slot #3 attached to a monitor screen is optional, but highly desirable. The user enters data interactively using the computer keyboard. Files are created or updated within the programs and stored on the program diskette. Output is available through the monitor screen or through a printer. Hard copy output can be printed on an Epson FX-80 or similar printer interfaced to the computer via a Grappler+ or similar card in slot #1. Switches in the printer hardware should be set to "condensed mode" to allow more than 80 characters to be printed per line.

PROGRAM DESCRIPTION

The program uses a dynamic allocation procedure that requires a catch and stocking data base to be updated on a periodic basis and that the user provide input each time the program is run. The program prompts the user for input in an interactive fashion. Input required to run the program includes a catch objective, species preferences on a percentage basis, the number of lake trout available to be stocked, the weights of coho and chinook salmon for the past two years, and the stocking locations for coho salmon. Using the input, current values in the data files, and internally coded variable values, the program computes a recommended stocking plan by species, size (fingerling/yearling), and management zone (Fig. 1).

The program system consists of eight programs and six data files (Fig. 2, Append. 1, 2, and 3). The programs are all self-run in response to user input, thus no computer programming knowledge is necessary to run the model. The first two programs are starter or driver programs which run automatically. The next four programs relate to management of the data base. The last two programs do the actual computations and provide output. The model flow is cyclic, always returning to the driver program and/or an option menu until the user wants to quit (see Fig. 2).

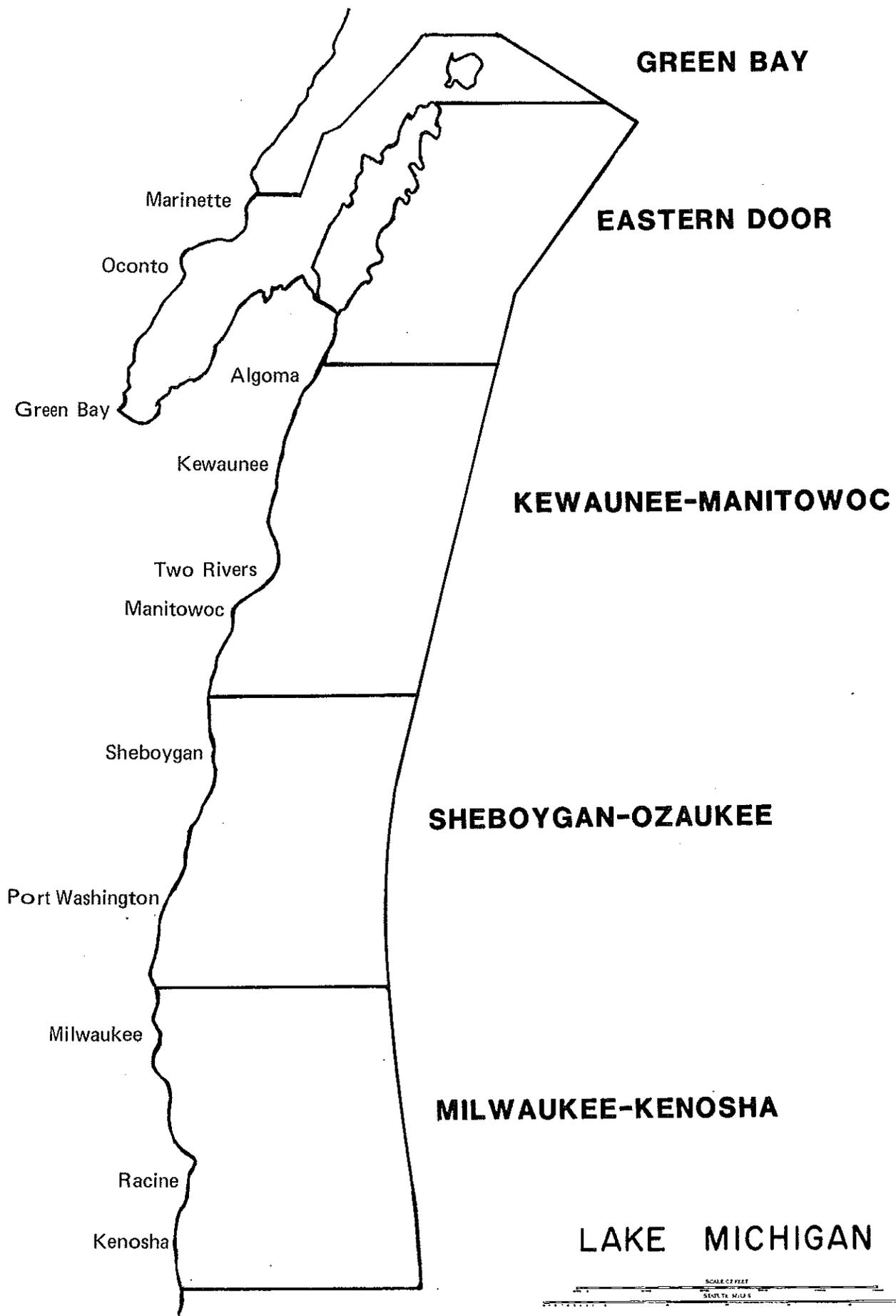


Figure 1. Salmonid management zones in the Wisconsin waters of Lake Michigan.

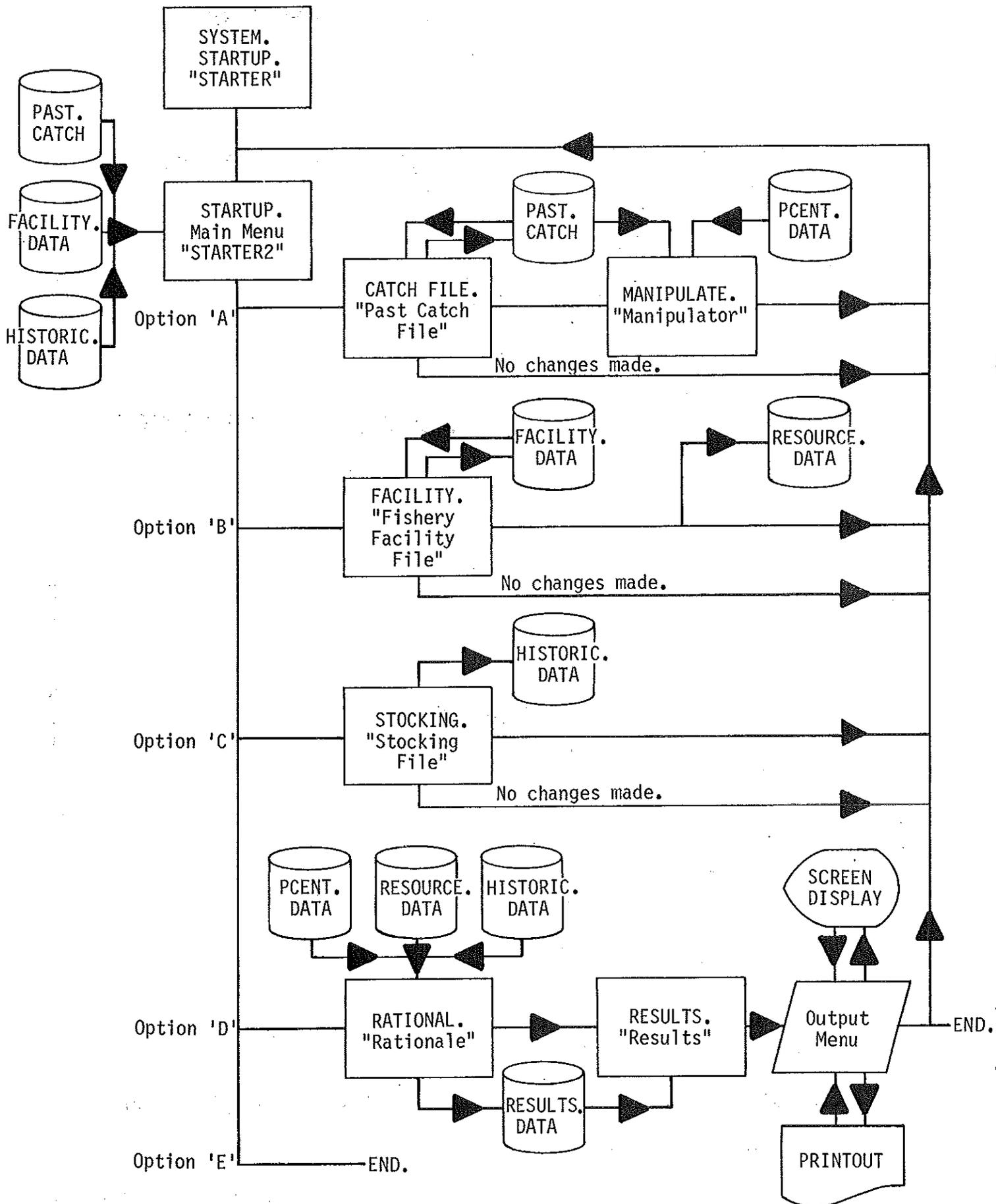


Figure 2. General flow of the salmonid stocking recommendation model.

The first program ("Starter", Fig. 2, Append. 1, 2, and 3) is run automatically when the computer is turned on (referred to as a "cold boot") or when the [RESET] key ([Control][Reset] for Apple IIe) is pressed (referred to as a "warm boot"). The printer must already be turned on in order for this program to execute properly. The program will display an introductory narrative of the model on the monitor screen and will prompt the user for input in order to run a driver program ("Starter2", Fig. 2, Append. 1 and 2). Program Starter2 displays an option menu providing access to programs that allow the user to input or update the data files, run the computational program, or quit using the model (see Example Operation section). Program Starter2 scans three major data files and displays the age of the available data on the monitor screen. This allows the user to immediately see if the data files are current or if they require updating with new information.

The next four programs ("PastCatchFile", "Manipulator", "FisheryFacilityFile", and "StockingFile"; Fig. 2, Append. 1 and 2) access and store information in the data files. These programs are run by choosing one of the three maintenance options given by menu in program Starter2 (see Example Operation section). The programs allow the user to initialize or input the entire data matrix for any of three files (past sport catch, fishery facilities, and past/proposed stocking). Two of the data files, past catch and past/proposed stocking files, can be updated to include a new current set of data without inputting the entire data matrix. The program prompts the user for data in a sequential fashion (e.g., catch by species by area by year by fishery).

The past catch file and the fishery facility file are read into internal files where the percentage of each value by area and/or fishery is computed. These percentage values are stored in separate files and used in program Rationale (Fig. 2, Append. 1, 2, and 5) to compute the stocking priority rank of each area for each species to be stocked. The past/proposed stocking file is used in the computational program ("Rationale") to compute the PAR or Predation Adjustment Ratio (Krueger and Dehring, in prep.).

Data are stored on floppy diskette as characters in text files. In effect, numbers (both real and integer) are stored as their character equivalents in character strings (e.g., the number 2 is stored as the character "2", 123 is stored as "1", "2", "3"). When they are read from the text file for use in the program, they are read as numbers. While the program is running, the data are used as numbers and converted to character strings if they must be stored in a disk file again. For these reasons, the same variable name may be used for a different data type (i.e., integer versus character string) in different programs (Append. 1; VAR lists).

All the data files and compiled program code are stored on diskette STOCK:. An extra copy of this diskette should be made before running the model. The copy should be updated each time a data base file is input or updated.

The last two programs and the remaining data file are accessed through the fourth option in the menu of Starter2. One of these programs, Rationale, contains the computational portion of the model which executes the procedure to develop the recommended stocking plan. The final program, Results, uses data generated by program Rationale to make further computations and arrange the data in a tabular form.

Program Rationale reads three of the data files: past catch by area and fishery file (STOCK:PCENT.DATA), the fishery facility by area file (STOCK:RESOURCE.DATA), and the past/proposed stocking file (STOCK:HISTORIC.DATA), and stores the information in internal memory (Fig. 2). Using the two percentage files along with equations and ratios described by Krueger and Dehring (in prep.), a tentative ranking for each area by fish species is determined. These computations take place in procedure RankSpecies (Append. 1, 2, and 5).

The program now asks the user to input information interactively on the keyboard. The user is asked to provide a catch objective, angler species preferences, the number of lake trout available to be stocked, the weights of coho and chinook salmon for the last two years, and the locations to stock coho salmon. When all the information is entered, the program determines the number of each species to stock in order to meet the specified catch objective in the species ratios given by the user. These values are computed using stock/catch ratios based on historic data (Append. 1; Krueger and Dehring, in prep.).

Coho salmon present a special case and are not distributed according to a species rank by zone. An alternative distribution procedure was developed because of the known migration patterns of coho salmon in Lake Michigan (Patriarche 1980). To accommodate these migration patterns, an equation was developed to calculate the proportion of coho to stock in each zone. The proportions stock the majority of fish in the northernmost management zones (excluding Green Bay) (Krueger and Dehring, in prep.). The remaining fish are allotted in decreasing proportions to management zones south to the state line. Green Bay receives the lowest allotment if it is chosen to be stocked. The equation is as follows:

$$P_i = (200/(n^2 + n)) * (n-(i-1))$$

where

P_i = the proportion of the total number to be stocked in zone i .

i = 1..n where 1 is the northernmost zone and n is either Milwaukee-Kenosha or Green Bay.

n = the total number of management zones to be stocked.

These equations are incorporated into program Rationale in procedure RankCoho (Append. 1, 2, and 6).

The next procedure in program Rationale (CheckForagePressure) is to compute PAR, the Predation Adjustment Ratio, and SFI, the Salmon Forage Indicator. These two variables are used to modify the recommended stocking (if necessary) in order that excessive predation pressure is not exerted on the Lake Michigan forage base.

PAR is an index based on the total predation pressure that existed in the Wisconsin waters of Lake Michigan in 1982, the predation pressure lost due to death of the oldest year classes, and the new predation pressure (recommended stocking). This index is calculated using a "predation factor" that standardizes the average annual forage consumption for each species relative to chinook salmon (brook trout and coho salmon = 0.53, brown and rainbow trout = 0.88; Stewart et al. 1981; Krueger and Dehring, in prep.). The predation factor for each species is multiplied by the total number of that species stocked to establish the year classes present in the Wisconsin waters of Lake Michigan in 1982. The sum of these products (the Total Predation Pressure in 1982: TPP82 = 24,293,610) was then used as the base figure to compare future recommended stockings. Using 1982 as a base figure assumes that the forage pressure that existed in Lake Michigan in 1982 was at an acceptable level (Krueger and Dehring, in prep.).

To compare the PAR, the program calculates the lost predation pressure from data contained in the past/proposed stocking file, and the new predation pressure from the stocking recommendations just developed. These values are used in the equation:

$$PAR = \frac{TPP82 - FLTPP - RPP}{SPP - FLTPP}$$

where

- TPP82 = the Total Predation Pressure in the Wisconsin waters of Lake Michigan in 1982 (24,293,610).
- FLTPP = the Future Lake Trout Predation Pressure (the number of lake trout available to be stocked multiplied by its forage consumption index of 0.88).
- RPP = the Residual Predation Pressure, the total predation pressure expected to be in the system one year prior to the recommended stocking minus the lost (decreased) predation pressure.
- SPP = the Stocked Predation Pressure, or the predation pressure expected from the recommended stocking.

The Salmon Forage Indicator (SFI) is used as a feedback mechanism from forage populations to salmonid numbers, and is based on changes in the average weights of coho and chinook salmon in the past two years (Krueger and Dehring, in prep.). Stocking recommendations are modified based on the direction and magnitude of change in salmon growth. This method of measuring the available forage assumes that the forage base is comprised of evenly distributed populations. The ratio of salmon weights used in the program is calculated as follows:

$$SFI = \frac{((\text{weight of age II+ coho salmon in year } t) + (\text{weight of age I+ chinook salmon in year } t))}{((\text{weight of age II+ coho salmon in year } t-1) + (\text{weight of age I+ chinook salmon in year } t-1))}$$

If the sum of PAR and SFI is greater than 2, then PAR and SFI are each set equal to 1. This step prevents the proposed stocking number from being increased in excess of that required to achieve the specified catch objective.

All the recommended stocking levels are multiplied by SFI and PAR to determine the final species stocking recommendations in procedure CalculateStocking (program Rationale). The species stocking recommendations are then multiplied by the tentative ranking for each zone by species calculated earlier in the program (procedures RankSpecies and RankCoho). These values represent the recommended stocking of each species in each zone based on user input, the data files, the PAR, and the SFI.

Finally, in procedure MinRequirements, program Rationale checks to make sure that certain species in specific management zones receive a minimum stocking. This step is required in order to ensure that adequate adult numbers return at maturity to certain stocking sites where WDNR collects spawn for propagation. Fertilized eggs are collected at Strawberry Creek (eastern Door County) for chinook salmon, the Sheboygan River (Sheboygan-Ozaukee counties) for coho salmon, and the Oconto River (Green Bay) for rainbow trout. If the specified zone is not receiving at least 10% of the total lakewide stocking, then the program allocates 10% to that zone and deducts equal proportions from all other zones being stocked to make up the difference.

These values now form the final adjusted stocking recommendations and are stored, along with the user input that achieved them, in file STOCK:RESULTS.DATA. The next program, Results, is automatically run at the end of program Rationale.

Program Results is designed to provide output concerning the stocking rationale program (Append. 1 and 2). Using the data stored during program Rationale, five output tables are created. In addition to the tables, a list of the parameters used in the runstream is also available. These outputs are made available to the user in a menu format and can be printed on the monitor screen or the printer. The user may select to display the tables on the screen, or to have all the output printed (see Example Operation section).

The list of parameters is made available as an output to reference the parameters that were input by the user to produce the stocking recommendations. This is an especially useful label when several scenarios are being run and printed in a short time or to double check user input. The parameter list is always printed at the top of a printed output.

Table 1 follows the parameter list and is a detailed report of the stocking recommended by the program, using the catch objective, species preference, and ecological data given by the user. The stocking recommendation is made for two years into the future to accommodate the hatchery production schedule. Included in Table 1 are the recommended numbers of fingerlings and yearlings of all species by area, total numbers of fingerlings and yearlings by species, the total number of all fish by species, and the grand total of the entire stocking recommendation. The program calculates the numbers of fingerlings and yearlings to stock by using fingerling/yearling stocking ratios calculated from past stocking practices (Krueger and Dehring, in prep.). These values are written into program Results in procedure CalculateStocking (Append. 1 and 2).

Parameters used in this run.

Catch objective : 450000
 Preference for each species (%) :
 Brook : 2
 Brown : 15
 Rainbow : 9
 Chinook : 47
 Coho : 27

Lake trout available for stocking : 1000000
 Weight of chinook salmon in 1985 : 5.40
 Weight of chinook salmon in 1984 : 4.80
 Weight of coho salmon in 1985 : 5.86
 Weight of coho salmon in 1984 : 5.55

Table 1. Recommended stocking levels for 1987 (numbers of fish).

Management Zone	Brook	Brown	Rainbow	Chinook	Coho	Total
Green Bay						
Yearlings	21600	138100	98300	000	000	258000
Fingerlings	7400	137000	112700	305000	000	562000
East Door County						
Yearlings	11900	56200	45200	000	000	113000
Fingerlings	4100	55800	51800	197000	000	309000
Kewaunee to Manitowoc						
Yearlings	43300	180700	137000	000	000	361000
Fingerlings	14700	179300	157000	612000	000	963000
Sheboygan to Ozaukee						
Yearlings	50700	110400	92300	000	285000	538000
Fingerlings	17300	109600	105700	453000	000	686000
Milwaukee to Kenosha						
Yearlings	17200	117500	103000	000	143000	381000
Fingerlings	5800	116500	118000	403000	000	643000
Total Yearlings	144700	602900	475800	000	428000	1651000
Total Fingerlings	49300	598100	545200	1970000	000	3163000
TOTAL	194000	1201000	1021000	1970000	428000	4814000

Table 2. Predicted catch by area from the 1987 stocking.

Management Zone	Brook	Brown	Rainbow	Chinook	Coho	Total
Green Bay	1200	14200	7700	30100	000	53300
East Door Co.	700	5800	3500	19400	000	29500
Kewaunee to Manitowoc	2500	18600	10700	60400	000	92200
Sheboygan to Ozaukee	2900	11400	7200	44700	74400	140600
Milwaukee to Kenosha	1000	12100	8100	39800	37300	98200
TOTAL	8300	62100	37300	194400	111700	413700
OBJECTIVE	9000	67500	40500	211500	121500	450000
DIFFERENCE	-8%	-8%	-8%	-8%	-8%	

Table 3. Predicted catch by fishery from the 1987 stocking.

Fishery	Brook	Brown	Rainbow	Chinook	Coho	Total
Trolling	500	24900	9600	98900	87100	221000
Pier	3100	13700	9900	12100	15200	54000
Shore	2000	14400	7900	8200	2900	35400
Stream	2600	9000	9800	75200	6500	103100
TOTAL	8300	62100	37300	194400	111700	413800

Table 4. Predicted rearing and distribution costs to the state hatchery system for the 1987 recommended stocking.

Species	1981-82 Cost		Stocking Numbers		Total Cost
	Fingerlings	Yearlings	Fingerlings	Yearlings	
Brook Trout	\$ 0.089	\$ 0.248	49000	145000	\$ 40277.1
Brown Trout	\$ 0.102	\$ 0.243	598000	603000	\$ 207511.
Rainbow Trout	\$ 0.099	\$ 0.225	545000	476000	\$ 161028.
Chinook Salmon	\$ 0.018	\$ 0.000	1970000	0000	\$ 35460.0
Coho Salmon	\$ 0.000	\$ 0.225	0000	428000	\$ 96300.0
Total Cost					\$ 540576.

Table 5. Stocking recommendations for lake trout into the Wisconsin waters of Lake Michigan.

Management Zone	Distribution Proportion	Recommended Stocking
Clay Banks	0.21	210000
Mid-Lake Reef	0.50	500000
Kewaunee to Kenosha	0.29	290000
Total		1000000

Table 2 is the output of the predicted catch by species and area from the recommended stocking using past stock/catch ratios (Krueger and Dehring, in prep.). The predicted catch is summed by species and zone and the totals are given on the table. Also in Table 2 is the original catch objective specified by the user and listed by species. This value is compared to the predicted catch from the final stocking recommendations by species and the difference is given in percent.

Table 3 lists the predicted catch from this stocking by species and fishery. The percent of the total catch predicted to be caught of each species by each fishery is derived from the past catch data files (STOCK:PCENT.DATA). These values are summed by species and fishery. The catch totals by species will agree with the species total listed in Table 2.

Table 4 is the output of the predicted rearing and distribution costs associated with the recommended stocking. Table 4 lists the estimated costs (based on 1981-82 costs) per fingerling and yearling of each species, the total number of fingerlings and yearlings of each species recommended to be stocked, the cost per species, and the total cost. The cost values are written into program Results in procedure Table4a (Append. 1 and 2).

Table 5 is the output of the recommended stocking plan for the lake trout available to be stocked (specified by the user) in the Wisconsin waters of Lake Michigan. The management zones and the proportion of fish to be distributed within each zone are taken from Krueger and Dehring (in prep.). These values are written into program Results in procedure Table5 (Append. 1 and 2).

The final two options available on the menu given in program Rationale are to (1) restart the program or (2) quit. Choosing the option of restarting the program will rerun program Starter2, once again giving the user the options of updating any file or running the stocking rationale program. Choosing the option to quit transfers the computer into Apple Pascal mode.

EXAMPLE OPERATION

This section was written to provide a sample execution of the system as it is being run on the computer. We suggest that users read this section through once and then go through it again actually using the computer program, following the steps listed, and entering the same data.

If you have received the diskette STOCK: from the authors, and it does not include the four Apple Pascal system files listed in Table 1, Append. 3, they will need to be written onto the diskette (see page 34 of Apple Pascal Operating System Reference Manual). These files can be transferred by following the steps listed below:

1. Place Apple 1 Pascal diskette in drive 1 (#4).
2. Place STOCK: diskette in drive 2 (#5).
3. Turn computer on.
4. Type F. (for F (iler))

5. Type T. (for T (ransfer))
6. Computer will respond with: Transfer What File?
7. Type #4: file name.extension
(file name.extension from Append. 3)
8. Computer will respond with: To Where?
9. Type #5: file name.extension
10. Repeat steps 5-9 for each file
11. Type Q. for (For Q (uit))

The stocking rationale programs are self-driven, requiring only the insertion of diskette STOCK: into disc drive #1 and turning on the computer and printer. Although the programs are written in Apple Pascal, it is not necessary to load the Apple Pascal system before running the programs. There is sufficient Apple Pascal on diskette STOCK: to run the programs. The program will activate the 80 column board (if one is available), set margins on the printer, and present an introduction to the programs. The introduction will appear as follows:

A program to allocate the annual stocking of salmonids
in the Wisconsin waters of Lake Michigan

This program develops recommendations on the species, life stage, numbers, and geographic distribution of salmonids to be stocked in the Wisconsin waters of Lake Michigan.

Following this display is a menu which offers you various file creation or maintenance options. Also available is an option to run the stocking program if the available data files are in order.

While running the stocking program, you will be requested to provide various information regarding catch objective, species preference, salmon weights, etc. Each time the program prompts you for information, enter the data and press [RETURN].

If you are ready to continue, press [RETURN].

When you press the return key, a second program is run which scans three major data files, prints the files (years), and prints options available on the monitor screen in a menu format. Available options include re-initializing or updating the data files, running the main program, or quitting. The menu appears as follows:

*** MAIN MENU ***

A - Past Catch File

A file containing the past catch by species, area,
and fishery for the years 1981, 1982, and 1983.

B - Fishery Facility File

A file containing data about the available fishery facilities (boat ramps, piers, etc.) by geographic area current through 1983.

C - Past Stocking File

A file containing the past stocking numbers by species from 1977 through 1984, and proposed stocking numbers for 1985 and 1986.

D - Stocking Program

If the data files appear to be in order, you may choose this option.

E - Quit.

Choose one option and press [RETURN].

The years listed in the first three menu options are variable and dependent on data in the files. If option 'A', 'B', or 'C' is chosen, the program will automatically run the corresponding program. Several questions are then asked by the program which will require the user to respond with various answers (see below for example).

Option 'A' is chosen to initialize or update the past catch file by running program PastCatchFiles (see Append. 1). First, the program displays the data file in a tabular form and provides the option of printing the file on the printer. The program then asks the user, "Do you want to initialize the past catch file (y/n)?" If your reply is yes, press "y". Do not press "y" at this time unless you want to input the entire data file. The past catch of brook, brown, and rainbow trout, and coho and chinook salmon by area and fishery for three years (i.e., 1982, 1983, and 1984) is requested in sequential fashion (i.e., Brook Green Bay 1983 Trolling:). After each catch value is typed in, press [RETURN]. Periodically the program will stop prompting for data, and disk drive #1 will be activated. While this happens, the computer is taking catch data stored in memory and writing it to disk in file STOCK:PAST.CATCH. When the memory buffer is cleared, the program will resume requesting catch data input until all the data is keyed in. The program will then display the data on the screen to allow the user to check the keyed input against the original data table for discrepancies. An option is again given to print the data on the printer.

If you had chosen not to re-initialize the past catch file (by keying in "n" in response to the original question), the program asks, "Do you want to update the catch file (y/n)?" This option is used when new catch data is available. If you reply "y" for yes, the program reads the past catch file, discards the oldest year of data and asks you for the most current year's data (in sequential fashion once again). Here again, when all the data are entered, they are displayed for the user to cross check for errors.

If there are no errors to correct, the program runs another program: Manipulator (see Append. 1). Program Manipulator reads the past catch data file, computes an average catch for the past three years by species and area, and determines the percentage of catch by species for each area and fishery. These data are stored on diskette in file STOCK:PCENT.DATA.

The introductory program which displays the main menu is then re-run, allowing you to update the other files, run the main program, or quit.

Option "B" should be chosen if there are any changes in the fishery facility file (i.e., development of new boat launching facilities, stream areas closed or opened to snagging, etc.). Choosing Option "B" causes program FisheryFacilityFile (see Append. 1) to be run. First, the data in the file are printed in tabular form. Next, the user is offered the option of having the data printed on the printer. Then the program asks, "Do you want to update the facility file (y/n)?" If any change must be made in the file, the entire matrix must be entered. If you respond "y" for yes, then the program requests the year for which the data is current, and the size, length, or number of each facility by area. The data are stored on diskette in file STOCK:FACILITY.DATA. After all the data are inputted, the program prints the values on the screen to allow the user to check the keyed input against the original values. If the values are correct, the percentage of each facility type by area is computed, the values are printed on the screen for inspection, and then stored on diskette in STOCK:RESOURCE.DATA. After this step is completed or if you typed "n" for the preceding question, the main menu program is run again.

Option "C" on the main menu is selected if new stocking data or proposed stocking data are available. If option "C" is chosen, program StockingFile is run (see Append. 1). The first question you are asked is: "Do you want to re-initialize the historical stocking file (y/n)?" Answering "y" for yes will require entering the entire matrix. This should only be done to correct errors in previous data. Answering "n" will prompt another question: "Do you want to update the historical file to include new stocking and/or proposed stocking data? (y/n)". Answering "y" for yes will prompt the next question of whether you want to input new actual stocking (for the past year) and planned stocking data (current and next year), or merely change the planned stocking data. Both options are available.

After entering the stocking and/or planned stocking values, the data are stored on diskette in STOCK:HISTORIC.DATA. The program then lists the data contained in the file in tabular form and offers the option for a printout to be made. Again, the program reruns the main menu, returning the user to the beginning of the program.

The last two options on the main menu are "D" and "E": Option "D" runs the main stocking rationale program, and Option "E" allows the user to quit the program. Choosing option "E" puts you out of the stocking rationale programs and into Apple Pascal.

If the user chooses Option "D", the program runs program Rationale, the main computing program of the stocking rationale program system (see Append. 1 and 6). The program reads the past catch percentage values for zones and fisheries (STOCK:PCENT.DATA) and the percentage values of facility by zones (STOCK:RESOURCE.DATA). By using predetermined weighting values for past catch and fishery facilities, a tentative ranking is arrived at for each zone by species (see procedure RankSpecies; Append. 1, 2, and 5).

The tentative ranking values are stored internally. The past stocking and proposed stocking values are read from diskette file STOCK:HISTORIC.DATA and stored internally also.

The program now requests data from the user. The first input requested is the catch objective exclusive of lake trout and exclusive of the moored boat angler catch. If the value provided is less than 50,000, the catch objective defaults to 50,000. If the value given by the user is more than 1,000,000, the catch objective value defaults to 1,000,000. These default values help prevent the program from "bombing" or causing a fatal internal error. If the program does bomb, a message will appear explaining the problem and telling you what to do to start over (i.e., press the spacebar).

Additional information requested by the program includes the percentage of the catch objective for each species, the current year and month, the number of lake trout available to be stocked, the weight of coho and chinook salmon in the past two years, whether or not you want to stock coho salmon in Green Bay, and how far north in Lake Michigan you want to stock coho salmon. The actual questions as they appear on the monitor screen with sample responses are given below:

What is the total catch objective for sportfish in the Wisconsin waters of Lake Michigan exclusive of lake trout and moored boat angler catch?: 450000 [RETURN]

List the percent of the objective preferred for each species:
They must sum to 100!

Brook : 2 [RETURN]
Brown : 15 [RETURN]
Rainbow: 9 [RETURN]
Chinook: 47 [RETURN]
Coho : 27 [RETURN]

What is the current year?: 1985 [RETURN]

What is the current month? (Jan = 1, Dec = 12): 12 [RETURN]

How many lake trout are available to be stocked in the Wisconsin waters of Lake Michigan in 1987?: 1000000 [RETURN]

Fish weights must be greater than 0.0.

What was the weight of age I+ chinook salmon in 1985: 5.4 [RETURN]

What was the weight of age I+ chinook salmon in 1984: 4.8 [RETURN]

What was the weight of age II+ coho salmon in 1985: 5.86 [RETURN]

What was the weight of age II+ coho salmon in 1984: 5.55 [RETURN]

Do you want to stock coho salmon in Green Bay? (y/n): n [RETURN]

What is the northernmost zone on Lake Michigan where coho are to be stocked?

- A) East Door County
- B) Kewaunee-Manitowoc
- C) Sheboygan-Ozaukee
- D) Milwaukee-Kenosha

Choose one: C [RETURN]

Using the entered data, the tentative rankings, and the historic stocking file, stocking recommendations are arrived at and stored on diskette in file STOCK:RESULTS.DATA. Program Results is then run automatically. This program reads the data from diskette file STOCK:RESULTS.DATA and stores the values internally. This process takes about 20 seconds.

Summary tables of the stocking recommendations are also constructed with program Results. The tables can be viewed on the monitor screen or printed in hard copy form. The available output options are given in a menu format as follows:

OUTPUT CHOICES

- A - Parameters used in this run.
- B - Table 1. Summarization of recommended stocking levels.
- C - Table 2. Predicted catch levels by area.
- D - Table 3. Predicted catch levels by fishery.
- E - Table 4. Projected cost of stocking.
- F - Table 5. Lake trout stocking recommendations.
- G - Print all of the above.
- H - Restart the program.
- I - Quit.

Select one letter and press [RETURN]: G [RETURN]

For our demonstration, we chose Option "G" and printed all of the tables. The printout, with the details we have selected, appears on the following pages.

ACKNOWLEDGMENTS

The authors express their appreciation to James T. Addis and Lee T. Kernen for their support and encouragement during the development of this project. Special thanks go to Nancy D. Tammi, Cornell University, for her helpful advice about the Apple Pascal operating system.

LITERATURE CITED

- Apple Computer, Inc.
1980. Apple Pascal Version 1.1 operating system reference manual (for Apple II). Apple Computer, Inc. 298 pp.
- Patriarche, M. H.
1980. Movement and harvest of coho salmon in Lake Michigan, 1978-79. Mich. Dep. Nat. Resour., Fish Res. Rep. No. 1889. 52 pp.
- Schultz, P. T.
1979. Lake Michigan trout and salmon stocking rationale. Wis. Dep. Nat. Resour., Anad. Fish Act. Comp. Rep., Unpubl. Mimeo. 51 pp.
- Stewart, D. J., J. F. Kitchell, and L. B. Crowder
1981. Forage fishes and their salmonid predators in Lake Michigan. Trans. Am. Fish. Soc. 110:751-763.

Donna Mears, Copy Editor

1940

1941

1942

1943

1944

1945

1946

APPENDIX 1.
Program listings.

```

PROGRAM Starter;

USES CHAINSTUFF;

VAR
  Reply : Char;
  Printer : Interactive;

PROCEDURE RollEm;

BEGIN
  REWRITE (Printer, 'PRINTER: ');
  WRITELN (Printer, CHR(27),CHR(108),CHR(15));
  CLOSE (Printer)
END;

PROCEDURE FirstFrame;

BEGIN
  WRITELN;
  WRITELN ('    A program to allocate the annual stocking of salmonids');
  WRITELN ('    in the Wisconsin waters of Lake Michigan');
  WRITELN; WRITELN; WRITELN;
  WRITELN ('This program develops recommendations on the species, life');
  WRITELN ('stage, numbers, and geographic distribution of salmonids to');
  WRITELN ('be stocked in the Wisconsin waters of Lake Michigan. ');
  WRITELN;
  WRITELN ('Following this display is a menu which offers you various file');
  WRITELN ('creation or maintenance options. Also available is an option');
  WRITELN ('to run the stocking program if the available data files are in');
  WRITELN ('order. '); WRITELN;
  WRITELN ('While running the stocking program, you will be requested to');
  WRITELN ('provide various information regarding catch objective, species');
  WRITELN ('preference, salmon weights, etc. Each time the program prompts');
  WRITELN ('you for information, enter the data and press <RETURN>. ');
  WRITELN; WRITELN;
  WRITE ('If you are ready to continue, press <RETURN> : ');
  READLN (Reply); WRITELN; WRITELN
END;

BEGIN
  RollEm;
  FirstFrame;
  SETCHAIN ('STOCK:STARTUP')
END.

```

```

PROGRAM Starter2;

USES CHAINSTUFF;

VAR
  DataFile : Text;
  CYr1, CYr2, CYr3, FYr, SYr1, SYr2, SYr3 : Integer;
  Answer, Reply : Char;

PROCEDURE GetCatchData;

BEGIN
  RESET (DataFile, 'STOCK:PAST.CATCH');
  READ (DataFile, CYr1, CYr2, CYr3);
  CLOSE (DataFile)
END;

PROCEDURE GetFacilityData;

BEGIN
  RESET (DataFile, 'STOCK:RESOURCE.DATA');
  READ (DataFile, FYr);
  CLOSE (DataFile)
END;

PROCEDURE GetStockData;

BEGIN
  RESET (DataFile, 'STOCK:HISTORIC.DATA');
  READ (DataFile, SYr1, SYr2, SYr3);
  CLOSE (DataFile)
END;

PROCEDURE MenuHeader;

BEGIN
  WRITELN ('                               *** MAIN MENU ***');
  WRITELN; WRITELN;
  WRITELN ('      A - Past Catch File');
  WRITELN ('          A file containing the past catch by species, area,');
  WRITE ('          and fishery for the years ',CYr1:4,', ',CYr2:4);
  WRITELN (' ',CYr3:4,', ');
  WRITELN;
END;

PROCEDURE MainMenu;

BEGIN
  MenuHeader;
  WRITELN ('      B - Fishery Facility File');
  WRITELN ('          A file containing data about the available fishery');
  WRITELN ('          facilities (boat ramps, piers, etc.) by geographic');
  WRITELN ('          area current through ',FYr:4,', ');
  WRITELN;
  WRITELN ('      C - Past Stocking File');
  WRITELN ('          A file containing the past stocking numbers by ');
  WRITE ('          species from ',(SYr1-7):4,' through ',SYr1:4,', and ');
  WRITELN ('proposed');

```

```

WRITELN ('          stocking numbers for',SYr2:5,' and',SYr3:5,'.');
WRITELN;
WRITELN ('      D - Stocking Program');
WRITELN ('          If the data files appear to be in order, you may');
WRITELN ('          choose this option. '); WRITELN;
WRITELN ('      E - Quit'); WRITELN;
WRITE ('Choose one option and press <RETURN> : ');
READLN (Answer);
END;

```

```

PROCEDURE GoFish;

```

```

VAR
Options : SET OF Char;

```

```

BEGIN
Options := ['A'..'E', 'a'..'e'];
IF NOT (Answer IN Options) THEN MainMenu;
CASE Answer OF
'A', 'a' : SETCHAIN ('STOCK:CATCHFILE');
'B', 'b' : SETCHAIN ('STOCK:FACILITY');
'C', 'c' : SETCHAIN ('STOCK:STOCKING');
'D', 'd' : SETCHAIN ('STOCK:RATIONAL')
END;
END;

```

```

BEGIN
GetCatchData;
GetFacilityData;
GetStockData;
MainMenu;
GoFish
END.

```

```

PROGRAM FastCatchFile;

USES CHAINSTUFF;

TYPE

Spp = (Brook, Brown, Rainbow, Chinook, Coho);
Area = (GreenBay, EastDoor, KewMan, ShebOz, MilKen);
Fishery = (Trolling, Pier, Shore, Stream);
YrCatch = (Crnt2, Crnt1, Crnt); (*Crnt = Current Yr, Crnt1 = Current Yr - 1*)
SppData = ARRAY [Brook..Coho, GreenBay..MilKen, Crnt2..Crnt,
                  Trolling..Stream] OF String;

VAR

FishryType : Fishery;
Species : Spp;
Yrs : YrCatch;
Zone : Area;
TempFile : SppData;
CatchFile, NewCatchFile : Text;
FshryLbl, SppLbl, YrLbl, ZoneLbl : String;
CatchNumber, CY, CY2, CY1, CY0, Year : String;
Answer, Reply : Char;
N, Number, Option : Integer;
LongNum : Integer[5];
NewNum : Real;
NCY, NCY2, NCY1, NCY0 : 1980..2000;
Medium : Interactive;
PrintOut : Boolean;

PROCEDURE PrintFile;

PROCEDURE GetInput;

BEGIN
  WRITE ('Do you want a print out of these values? (y/n): ');
  READLN (Answer);
  IF (Answer = 'Y') OR (Answer = 'y') THEN
    BEGIN
      CLOSE (Medium);
      PrintOut := True;
      PrintFile;
      WRITELN (Medium, CHR(27),CHR(108),CHR(15));
      CLOSE (Medium);
    END
  ELSE BEGIN
      GOTOXY (0,23);
      WRITE ('Press <RETURN> to continue: ');
      READLN (Reply);
      CLOSE (Medium);
    END;
END;

PROCEDURE ReadData;

BEGIN
  IF PrintOut THEN
    BEGIN
      RESET (Medium, 'Printer: ');

```

```

    WRITELN (Medium, CHR(27),CHR(108),CHR(0));
END
ELSE RESET (Medium, 'Console: ');
IF NOT PrintOut THEN
BEGIN
    WRITELN (Medium, 'Reading data file...');
    RESET (CatchFile, 'STOCK:PAST.CATCH');
    READ (CatchFile, NCY2, NCY1, NCY0);
    STR (NCY2, CY2);
    STR (NCY1, CY1);
    STR (NCY0, CY0);
    FOR Species:= Brook TO Coho DO
        FOR Zone:= GreenBay TO MilKen DO
            FOR Yrs:= Crnt2 TO Crnt DO
                FOR FishryType:= Trolling TO Stream DO
                    BEGIN
                        READ (CatchFile, LongNum);
                        STR (LongNum, CatchNumber);
                        TempFile (Species, Zone, Yrs, FishryType) := CatchNumber;
                    END;(*FishryType Loop Section*)
                CLOSE (CatchFile);
                WRITE (CHR(12));
            END;
        END;
    END;

BEGIN
    ReadData;
    GOTOXY (0,5);
    WRITELN (Medium);
    WRITE (Medium, 'These are the values currently in the past sport catch');
    WRITELN (Medium, ' file. ');
    WRITELN (Medium);
    FOR Species := Brook TO Coho DO
        BEGIN
            CASE Species OF
                Brook : Spplbl:='Brook Trout';
                Brown : Spplbl:='Brown Trout';
                Rainbow : Spplbl:='Rainbow Trout';
                Chinook : Spplbl:='Chinook Salmon';
                Coho : Spplbl:='Coho Salmon'
            END;
            WRITELN (Medium, Spplbl, ' catch values in the past catch file. ');
            WRITELN (Medium);
            WRITE (Medium, ' :10, 'Green Bay':17, 'Eastern Door':25);
            WRITE (Medium, 'Kewaunee-Manitowoc':27, 'Sheboygan-Ozaukee':24);
            WRITELN (Medium, 'Milwaukee-Kenosha':24);
            WRITE (Medium, 'Fishery ', CY2, CY1:8, CY0:8);
            FOR Zone := EastDoor TO MilKen DO
                BEGIN
                    WRITE (Medium, CY2:8, CY1:8, CY0:8);
                END;
                WRITELN (Medium);
                WRITE (Medium, '-----');
                WRITE (Medium, '-----');
                WRITELN (Medium, '-----');
            FOR FishryType := Trolling TO Stream DO
                BEGIN
                    CASE FishryType OF
                        Trolling : FshryLbl := ' Trolling ';
                        Pier : FshryLbl := ' Pier ';

```

```

        Shore : FshryLbl := ' Shore ' ;
        Stream : FshryLbl := ' Stream '
    END;
    WRITE (Medium, FshryLbl);
    FOR Zone := GreenBay TO MilKen DO
        FOR Yrs := Crnt2 TO Crnt DD
            BEGIN
                WRITE (Medium, TempFile [Species,Zone,Yrs,FishryType];5);
                WRITE (Medium, ' ':3);
            END;
            WRITELN (Medium);
        END;
        WRITELN (Medium);
    END;
    WRITELN (Medium);
    IF NOT PrintOut THEN GetInput;
END;

```

```

PROCEDURE GetNumber (VAR Number:String);

```

```

TYPE

```

```

    SetOfChar = SET OF Char;

```

```

VAR

```

```

    S1 : String [1];
    STemp : String [12];
    OKSet : SET OF Char;

```

```

FUNCTION GetChar (OKSet : SetOfChar) : Char;

```

```

VAR

```

```

    CH : Char;
    Good : Boolean;

```

```

BEGIN

```

```

    REPEAT

```

```

        READ (Keyboard, CH);
        IF EOLN (Keyboard) THEN CH := CHR(13);
        Good := CH IN OKSet;
        IF NOT Good THEN WRITE (CHR(7))
            ELSE IF CH IN ['0'..'9', '.'] THEN WRITE (CH);

```

```

    UNTIL Good;

```

```

    GetChar := CH;

```

```

END;

```

```

BEGIN

```

```

    OKSet := ['0'..'9', '.'];

```

```

    S1 := '';

```

```

    STemp := '';

```

```

    REPEAT

```

```

        IF LENGTH (STemp) = 0 THEN S1[1] := GetChar (OKSet + [CHR(13)])

```

```

        ELSE IF LENGTH (STemp) = 12 THEN S1[1] := GetChar ([CHR(13), CHR(8)])

```

```

            ELSE S1[1] := GetChar (OKSet + [CHR(13), CHR(8)]);

```

```

        IF S1[1] IN OKSet THEN STemp := CONCAT (STemp, S1)

```

```

        ELSE IF S1[1] = CHR(8) THEN

```

```

            BEGIN

```

```

                WRITE (CHR(8)); WRITE (' '); WRITE (CHR(8));

```

```

                DELETE (STemp, LENGTH (STemp), 1);

```

```

            END;

```

```

    UNTIL (S1[1] = CHR(13)) AND (LENGTH (STemp) <> 0);

```

```
Number := STeap;  
END;
```

```
PROCEDURE InitializeFile;
```

```
BEGIN
```

```
Write ('Do you want to initialize the past catch file (y/n)?');
```

```
Readln (Answer);
```

```
WHILE (Answer = 'y') OR (Answer = 'Y') DO
```

```
BEGIN
```

```
Option := 1;
```

```
REWRITE (CatchFile, 'STOCK:PAST.CATCH');
```

```
CatchNumber := '1980 1981 1982';
```

```
WRITELN (CatchFile, CatchNumber);
```

```
FOR Species:=Brook TO Coho DO
```

```
BEGIN
```

```
CASE Species OF
```

```
Brook : Spplbl:='Brook Trout';
```

```
Brown : Spplbl:='Brown Trout';
```

```
Rainbow : Spplbl:='Rainbow Trout';
```

```
Chinook : Spplbl:='Chinook Salmon';
```

```
Coho : Spplbl:='Coho Salmon'
```

```
END;
```

```
FOR Zone:=GreenBay TO MilKen Do
```

```
BEGIN
```

```
CASE Zone OF
```

```
GreenBay : ZoneLbl:= 'Green Bay';
```

```
EastDoor : ZoneLbl:= 'East Door County';
```

```
KewMan : ZoneLbl:= 'Kewaunee to Manitowoc';
```

```
ShebOz : ZoneLbl:= 'Sheboygan to Ozaukee';
```

```
Milken : ZoneLbl:= 'Milwaukee to Kenosha'
```

```
END;
```

```
FOR Yrs:=Crnt2 TO Crnt DO
```

```
BEGIN
```

```
CASE Yrs OF
```

```
Crnt2 : YrLbl:= '1980';
```

```
Crnt1 : YrLbl:= '1981';
```

```
Crnt : YrLbl:= '1982'
```

```
END;
```

```
FOR FishryType:=Trolling to Stream DO
```

```
BEGIN
```

```
CASE FishryType OF
```

```
Trolling : FshryLbl:= 'Trolling';
```

```
Pier : FshryLbl:= 'Pier ';
```

```
Shore : FshryLbl:= 'Shore ';
```

```
Stream : FshryLbl:= 'Stream '
```

```
END;
```

```
WRITE (Spplbl, ' ', ZoneLbl, ' ', FshryLbl, ' ', YrLbl, ' ');
```

```
GetNumber (CatchNumber);
```

```
WRITE (CatchFile, CatchNumber);
```

```
WRITE (CatchFile, ' ');
```

```
END;(*Fishery Loop & Write Section*)
```

```
END;(*Year Loop Section*)
```

```
WRITELN (CatchFile)
```

```
END;(*Zone Loop Section*)
```

```
END;(*Species Loop Section*)
```

```
CLOSE (CatchFile, LOCK);
```

```
WRITE (CHR(12));
```

```

PrintOut := False;
PrintFile;
WRITE ('Do you wish to re-enter the values in the file (y/n)? ');
READLN (Answer);
WRITE (CHR(12));
END;(*WHILE LOOP*)
END;

```

```
PROCEDURE ReadOldFile;
```

```

BEGIN
  RESET (CatchFile, 'STOCK:PAST.CATCH');
  READ (CatchFile, NCY2, NCY1, NCY0);
  STR (NCY2, CY2);
  STR (NCY1, CY1);
  STR (NCY0, CY0);
  WRITE ('The present catch file contains data from ');
  WRITELN(CY2, ', ', CY1, ', and ', CY0);
  FOR Species:= Brook TO Coho DO
    FOR Zone:= GreenBay TO Milken DO
      FOR Yrs:= Crnt2 TO Crnt DO
        FOR FishryType:= Trolling TO Stream DO
          BEGIN
            READ (CatchFile, Number);
            STR (Number, CatchNumber);
            TempFile (Species, Zone, Yrs, FishryType) := CatchNumber;
          END;(*FishryType Loop Section*)
        CLOSE (CatchFile);
      END;
    END;
  END;

```

```
PROCEDURE CopyFile;
```

```

BEGIN
  REWRITE (NewCatchFile, 'STOCK:PAST.CATCH');
  WRITELN (NewCatchFile, CY1, ', ', CY0, ', ', CY);
  FOR Species := Brook to Coho DO
    FOR Zone := GreenBay to Milken DO
      BEGIN
        FOR Yrs := Crnt2 to Crnt DO
          FOR FishryType := Trolling TO Stream DO
            BEGIN
              CatchNumber:=TempFile (Species, Zone, Yrs, FishryType);
              WRITE (NewCatchFile, CatchNumber);
              WRITE (NewCatchFile, ' ');
            END;
          WRITELN (NewCatchFile)
        END;
      CLOSE (NewCatchFile, LOCK)
    END;
  END;

```

```
PROCEDURE UpdateFile;
```

```

BEGIN
  WRITE ('Do you want to update the catch file (y/n)? ');
  READLN (Answer);
  WHILE (Answer = 'y') OR (Answer = 'Y') DO
    BEGIN
      Option := 1;
      ReadOldFile;
      WRITE ('What is the current year? ');

```

```

READLN (Year);
IF NCY = (NCY0 + 1) THEN
BEGIN
STR (NCY, CY);
FOR Species := Brook TO Coho DO
BEGIN
CASE Species OF
Brook : SppLbl := ' Brook ';
Brown : SppLbl := ' Brown ';
Rainbow: SppLbl := ' Rainbow ';
Chinook: SppLbl := ' Chinook ';
Coho : SppLbl := ' Coho '
END;
FOR Zone := GreenBay TO MilKen DO
BEGIN
CASE Zone OF
GreenBay : ZoneLbl := ' Green Bay ';
EastDoor : ZoneLbl := ' East Door Co. ';
KewMan : ZoneLbl := ' Kewaunee to Manitowoc ';
ShebOz : ZoneLbl := ' Sheboygan to Ozaukee ';
MilKen : ZoneLbl := ' Milwaukee to Kenosha '
END;
FOR FishryType := Trolling TO Stream DO
BEGIN
CASE FishryType OF
Trolling : FshryLbl := ' Trolling ';
Pier : FshryLbl := ' Pier ';
Shore : FshryLbl := ' Shore ';
Stream : FshryLbl := ' Stream '
END;
TempFile [Species, Zone, Crnt2, FishryType] :=
TempFile [Species, Zone, Crnt1, FishryType];
TempFile [Species, Zone, Crnt1, FishryType] :=
TempFile [Species, Zone, Crnt, FishryType];
WRITE (SppLbl, ZoneLbl, CY, FshryLbl, ' ');
GetNumber (CatchNumber);
TempFile [Species, Zone, Crnt, FishryType] := CatchNumber;
END;
END;
END;
ELSE CY:=CY0;
CopyFile;
Answer := 'N'
END;
END;

BEGIN
Option := 0;
WRITE (CHR(12)); GOTOXY (0,5);
PrintOut := False;
PrintFile;
WRITE (CHR(12));
InitializeFile;
WRITE (CHR(12));
UpdateFile;
WRITE (CHR(12));
IF (Option = 0) THEN SETCHAIN ('STOCK:STARTUP')
ELSE SETCHAIN ('STOCK:MANIPULATE')
END.

```

PROGRAM Manipulator;

USES CHAINSTUFF;

TYPE

Spp = (Brook, Brown, Rainbow, Chinook, Coho);
Area = (GreenBay, EastDoor, KewMan, ShebOz, MilKen);
Fishery = (Trolling, Pier, Shore, Stream);
YrCatch = (Crnt2, Crnt1, Crnt);
AreaFile = ARRAY [Brook..Coho, GreenBay..MilKen] OF Real;
FshryFile= ARRAY [Brook..Coho, Trolling..Stream] OF Real;

VAR

FishryType : Fishery;
Species : Spp;
Zone : Area;
Yrs : YrCatch;
CatchFile, PercentFile : Text;
Numbers, PCatch : String;
AreaRank : AreaFile;
FshryRank : FshryFile;
PrCatch, N : Integer;
CatchNumber, AreaTotal, PerCatch, TotCatch, Total : Real;
TrollCatch, PierCatch, ShorCatch, StrmCatch : Real;
Average, TotAvg : Real;

PROCEDURE PrintInfo;

BEGIN

REWRITE (PercentFile, 'STOCK:PCENT.DATA');

FOR Species := Brook to Coho DO

 BEGIN

 FOR Zone := GreenBay TO MilKen DO

 BEGIN

 PerCatch := AreaRank [Species, Zone];

 PrCatch := TRUNC (PerCatch * 10000);

 STR (PrCatch, PCatch);

 WRITE (PercentFile, PCatch, ' ');

 END;

 FOR FishryType := Trolling TO Stream DO

 BEGIN

 PerCatch := FshryRank [Species, FishryType];

 PrCatch := TRUNC (PerCatch * 10000);

 STR (PrCatch, PCatch);

 WRITE (PercentFile, PCatch, ' ');

 END;

 WRITELN (PercentFile)

 END;

 CLOSE (PercentFile, LOCK)

END;

BEGIN

 RESET (CatchFile, 'STOCK:PAST.CATCH');

 READLN (CatchFile, Numbers);

 FOR Species:= Brook TO Coho DO

 BEGIN

```

TrollCatch := 0;
PierCatch := 0;
ShorCatch := 0;
StrmCatch := 0;
CatchNumber:= 0;
TotCatch := 0;
TotAvg := 0;
N := 0;
FOR Zone:= GreenBay TO MilKen DO
  BEGIN
  AreaTotal := 0;
  Average := 0;
  FOR Yrs:= Crnt2 TO Crnt DO
    BEGIN
    Total := 0;
    N := N + 1;
    FOR FishryType:= Trolling TO Stream DO
      BEGIN
      READ (CatchFile, CatchNumber);
      Total := Total + CatchNumber;
      CASE FishryType OF
        Trolling : TrollCatch := TrollCatch + CatchNumber;
        Pier : PierCatch := PierCatch + CatchNumber;
        Shore : ShorCatch := ShorCatch + CatchNumber;
        Stream : StrmCatch := StrmCatch + CatchNumber
      END;
      TotCatch := TotCatch + CatchNumber
    END;(*FishryType Loop Section*)
    AreaTotal := AreaTotal + Total;
  END;(*Species FOR*)
  Average := AreaTotal / 3;
  AreaRank (Species, Zone) := Average;
  TotAvg := TotAvg + Average;
END;
FOR Zone := GreenBay TO MilKen DO
  BEGIN
  AreaRank (Species, Zone) := AreaRank (Species, Zone) / TotAvg
  END;
FOR FishryType:= Trolling TO Stream DO
  BEGIN
  CASE FishryType OF
    Trolling : FshryRank (Species, Trolling) := TrollCatch / TotCatch;
    Pier : FshryRank (Species, Pier) := PierCatch / TotCatch;
    Shore : FshryRank (Species, Shore) := ShorCatch / TotCatch;
    Stream : FshryRank (Species, Stream) := StrmCatch / TotCatch
  END;
  END;
END;
CLOSE (CatchFile);
PrintInfo;
SETCHAIN ('STOCK:STARTUP')
END.(*Program*)

```

```
PROGRAM FisheryFacilityFile;
```

```
USES CHAINSTUFF;
```

```
TYPE
```

```
Area = (GreenBay, EastDoor, KenMan, ShebOz, MilKen);  
Facility = (Ramps, Moored, Piers, ShrLine, StrmLength, CloseSnag, OpenSnag);  
FacRes = (Boats, FishingPiers, ShoreLine, AnglingStream, SnaggingStream);  
FacData = ARRAY [GreenBay..MilKen, Ramps..OpenSnag] OF Real;  
FacMeasures = ARRAY [GreenBay..MilKen, Boats..SnaggingStream] OF Real;  
SetOfChar = SET OF Char;
```

```
VAR
```

```
Zone : Area;  
Fac : Facility;  
Res : FacRes;  
Data : FacData;  
ResData : FacMeasures;  
FacLbl, ZoneLbl : String;  
FacNumber, StrNumber, ResNumber, Year : String;  
FacFile, ResFile : Text;  
Answer, Reply : Char;  
Number : Integer;  
TotRamps, TotMoored, TotPiers, TotShore, TotStream, TotNoSnag, TotSnag : Real;  
FaxNumber : Real;  
Medium : Interactive;  
Printout : Boolean;
```

```
PROCEDURE PrintFacFile;
```

```
PROCEDURE GetInput;
```

```
BEGIN
```

```
  WRITE ('Do you want a print out of these values? (y/n): ');
```

```
  READLN (Answer);
```

```
  IF (Answer = 'Y') OR (Answer = 'y') THEN
```

```
    BEGIN
```

```
      CLOSE (Medium);
```

```
      PrintOut := True;
```

```
      PrintFacFile;
```

```
      CLOSE (Medium);
```

```
    END
```

```
    ELSE BEGIN
```

```
      GOTOXY (0,23);
```

```
      WRITE ('Press <RETURN> to continue: ');
```

```
      READLN (Reply);
```

```
      CLOSE (Medium);
```

```
    END;
```

```
END;
```

```
BEGIN
```

```
  IF PrintOut THEN RESET (Medium, 'Printer: ');
```

```
  ELSE RESET (Medium, 'Console: ');
```

```
  IF NOT PrintOut THEN
```

```
    BEGIN
```

```
      WRITELN (Medium, 'Reading data file...');
```

```
      RESET (FacFile, 'STOCK:FACILITY.DATA');
```

```
      FOR Zone := GreenBay TO MilKen DO
```

```

FOR Fax := Ramps TO OpenSnag DO
  BEGIN
    READ (FacFile, FaxNumber);
    Data [Zone, Fax] := FaxNumber;
  END;
CLOSE (FacFile);
WRITE (CHR(12));
END;
GOTOXY (0,5);
WRITE (Medium, 'These are the values currently in the fishery facility');
WRITELN (Medium, ' file. ');
WRITELN (Medium);
WRITELN (Medium, ' :69, 'Miles of Stream');
WRITE (Medium, 'Management Number', 'Number of':14, 'Feet of':11);
WRITELN (Medium, 'Miles of':12, 'Closed to':23, 'Open to':11);
WRITE (Medium, 'Zone':7, 'of Ramps':13, 'Moored Boats':14, 'Piers':9);
WRITELN (Medium, 'Shoreline':14, 'Total':10, 'Snagging':11, 'Snagging':13);
WRITE (Medium, '-----');
WRITELN (Medium, '-----');
FOR Zone := GreenBay TO MilKen DO
  BEGIN
    CASE Zone OF
      GreenBay : ZoneLbl:= 'Green Bay';
      EastDoor : ZoneLbl:= 'East Door';
      KewMan   : ZoneLbl:= 'Kwne-Mntc';
      ShebOz  : ZoneLbl:= 'Sheb-Ozke';
      MilKen  : ZoneLbl:= 'Milke-Ken'
    END;
    WRITE (Medium, ZoneLbl:9, ' ', Data [Zone, Ramps] :6:2, ' ');
  FOR Fax := Moored to OpenSnag DO
    BEGIN
      IF Data [Zone, Fax] > 9999 THEN
        WRITE (Medium, Data [Zone, Fax]:6:1, ' :5)
      ELSE
        WRITE (Medium, Data [Zone, Fax]:6:2, ' :5);
      END;
      WRITELN (Medium);
    END;
    WRITELN (Medium);
    IF NOT PrintOut THEN GetInput;
  END;
PROCEDURE WriteResFile;
BEGIN
  REWRITE (ResFile, 'STOCK:RESOURCE.DATA');
  WRITELN (ResFile, Year);
  FOR Zone := GreenBay TO MilKen DO
    BEGIN
      FOR Res := Boats TO SnaggingStream DO
        BEGIN
          CASE Res OF
            Boats      : ResData [Zone, Res] := (Data [Zone, Ramps] * 0.483)
              + (Data [Zone, Moored] * 0.517);
            FishingPiers : ResData [Zone, Res] := Data [Zone, Piers];
            Shoreline   : ResData [Zone, Res] := Data [Zone, ShrLine];
            AnglingStream : ResData [Zone, Res] := (Data [Zone, StraLength]
              + Data [Zone, CloseSnag]) * 0.5;
            SnaggingStream : ResData [Zone, Res] := Data [Zone, OpenSnag];
          END;
        END;
      END;
    END;
  END;

```

```

        Number := TRUNC (ResData [Zone, Res] * 10000);
        STR (Number, StrNumber);
        WRITE (ResFile, StrNumber, ' ');
    END;
    WRITELN (ResFile)
END;
CLOSE (ResFile, LOCK);
END;

PROCEDURE ComputeResults;

BEGIN
    RESET (FacFile, 'STOCK:FACILITY.DATA');
    FOR Zone := GreenBay TO Milken DO
        FOR Fax := Ramps TO OpenSnag DO
            BEGIN
                READ (FacFile, FaxNumber);
                Data [Zone, Fax] := FaxNumber;
            END;
        CLOSE (FacFile);
        TotRamps := 0; TotMoored := 0; TotPiers := 0; TotShore := 0;
        TotStream:= 0; TotNoSnag := 0; TotSnag := 0;
        BEGIN
            FOR Zone := GreenBay TO Milken DO
                FOR Fax := Ramps to OpenSnag DO
                    BEGIN
                        CASE Fax OF
                            Ramps   : TotRamps := TotRamps + Data [Zone, Fax];
                            Moored   : TotMoored:= TotMoored + Data [Zone, Fax];
                            Piers    : TotPiers := TotPiers + Data [Zone, Fax];
                            ShrLine  : TotShore := TotShore + Data [Zone, Fax];
                            StrmLngh : TotStream:= TotStream + Data [Zone, Fax];
                            CloseSnag: TotNoSnag:= TotNoSnag + Data [Zone, Fax];
                            OpenSnag : TotSnag := TotSnag + Data [Zone, Fax]
                        END;
                    END;
                END;
            FOR Zone := GreenBay TO Milken DO
                FOR Fax := Ramps to OpenSnag DO
                    BEGIN
                        CASE Fax OF
                            Ramps   : Data [Zone, Fax] := Data [Zone, Fax] / TotRamps;
                            Moored   : Data [Zone, Fax] := Data [Zone, Fax] / TotMoored;
                            Piers    : Data [Zone, Fax] := Data [Zone, Fax] / TotPiers;
                            ShrLine  : Data [Zone, Fax] := Data [Zone, Fax] / TotShore;
                            StrmLngh : Data [Zone, Fax] := Data [Zone, Fax] / TotStream;
                            CloseSnag: Data [Zone, Fax] := Data [Zone, Fax] / TotNoSnag;
                            OpenSnag : Data [Zone, Fax] := Data [Zone, Fax] / TotSnag
                        END;
                    END;
                END;
            WriteResFile;
        END;

PROCEDURE GetNumber (VAR Number:String);

TYPE
    SetOfChar = SET OF Char;

VAR
    S1 : String [1];

```

```

STemp : String (12);
OKSet : SET OF Char;

FUNCTION GetChar (OKSet : SetOfChar) : Char;

VAR
  CH : Char;
  Good : Boolean;

BEGIN
  REPEAT
    READ (Keyboard, CH);
    IF EOLN (Keyboard) THEN CH := CHR(13);
    Good := CH IN OKSet;
    IF NOT Good THEN WRITE (CHR(7))
      ELSE IF CH IN ('0'..'9', '.') THEN WRITE (CH);
    UNTIL Good;
    GetChar := CH;
  END;

BEGIN
  OKSet := ['0'..'9', '.'];
  S1 := '';
  STemp := '';
  REPEAT
    IF LENGTH (STemp) = 0 THEN S1[1] := GetChar (OKSet + [CHR(13)])
      ELSE IF LENGTH (STemp) = 12 THEN S1[1] := GetChar ([CHR(13), CHR(8)])
        ELSE S1[1] := GetChar (OKSet + [CHR(13), CHR(8)]);
    IF S1[1] IN OKSet THEN STemp := CONCAT (STemp, S1)
      ELSE IF S1[1] = CHR(8) THEN
        BEGIN
          WRITE (CHR(8)); WRITE (' '); WRITE (CHR(8));
          DELETE (STemp, LENGTH (STemp), 1);
        END;
    UNTIL (S1[1] = CHR(13)) AND (LENGTH (STemp) <> 0);
    Number := STemp;
  END;

PROCEDURE InitializeFile;

BEGIN
  Write ('Do you want to update the Facility file (y/n)?');
  Readln (Answer);
  WHILE (Answer = 'y') OR (Answer = 'Y') DO
    BEGIN
      WRITE ('What is the year for which this data is current? : ');
      GetNumber (Year);
      REWRITE (FacFile, 'STOCK:FACILITY.DATA');
      FOR Zone := GreenBay TO MilKen DO
        BEGIN
          CASE Zone OF
            GreenBay : ZoneLbl:= 'Green Bay';
            EastDoor : ZoneLbl:= 'East Door County';
            KewMan : ZoneLbl:= 'Kewaunee to Manitowoc';
            ShebOz : ZoneLbl:= 'Sheboygan to Ozaukee';
            MilKen : ZoneLbl:= 'Milwaukee to Kenosha';
          END;
          FOR Fax := Ramps to OpenSnag DO
            BEGIN
              CASE Fax OF

```

```

Raaps : FacLbl:= ' Number of Raaps ';
Moored : FacLbl:= ' Number of Moored Boats ';
Piers : FacLbl:= ' Feet of Piers ';
ShrLine : FacLbl:= ' Miles of Shoreline ';
StrmLngh: FacLbl:= ' Miles of Stream ';
CloseSnag: FacLbl:= ' Streamlength Closed to Snagging ';
OpenSnag : FacLbl:= ' Streamlength Open to Snagging '
END;
WRITE (ZoneLbl, FacLbl, ' :');
GetNumber (FacNumber);
WRITE (FacFile, FacNumber);
WRITE (FacFile, ' ');
WRITELN
END;
WRITELN (FacFile)
END; (*Zone Loop Section*)
CLOSE (FacFile, LOCK);
WRITE (CHR(12));
PrintOut := False;
PrintFacFile;
WRITE ('Do you wish to reenter the values currently in the file (y/n): ');
READLN (Answer);
WRITE (CHR(12));
ComputeResults
END;(*WHILE LOOP*)
END;

BEGIN
WRITE (CHR(12)); GOTOXY (0,5);
PrintOut := False;
PrintFacFile;
WRITE (CHR(12));
InitializeFile;
WRITE (CHR(12));
SETCHAIN ('STOCK:STARTUP')
END.

```

```

PROGRAM StockingFile;

USES CHAINSTUFF;

TYPE

Spp = (Brook, Brown, Rainbow, Chinook, Coho, Lake);
HistoricalFile = ARRAY [Brook..Lake, 1973..2000] OF Real;

VAR

Species : Spp;
Hist : HistoricalFile;
HistFile : Text;
HistNumber, SppLbl, Year, Year1, Year2 : String;
INumber, Years, Yr, Yr0, Yr1, Yr2 : Integer;
Number, Yer : Real;
Ansl, Answer, Reply : Char;
Medium : Interactive;
PrintOut : Boolean;

PROCEDURE PrintFile;

BEGIN
  IF PrintOut THEN RESET (Medium, 'Printer: ');
  ELSE RESET (Medium, 'Console: ');
  IF NOT PrintOut THEN
    BEGIN
      WRITELN (Medium, 'Reading data file...');
      RESET (HistFile, 'STOCK:HISTORIC.DATA');
      READ (HistFile, Yr0, Yr1, Yr2);
      Yr := Yr2;
      FOR Species := Brook TO Lake DO
        FOR Years := Yr DOWNT0 (Yr - 9) DO
          BEGIN
            READ (HistFile, Number);
            Hist (Species, Years) := Number;
          END;
        CLOSE (HistFile);
        WRITE (CHR(12));
      END;
      GOTOXY (0,5);
      WRITELN (Medium, 'These are the values currently in the stocking file: ');
      WRITELN (Medium);
      WRITE (Medium, ' :10, 'Brook':10, 'Brown':10, 'Rainbow':10, 'Chinook':10);
      WRITELN (Medium, 'Coho':10, 'Lake':10);
      FOR Years := Yr DOWNT0 Yr - 9 DO
        BEGIN
          WRITE (Medium, Years:7, ' ');
          FOR Species := Brook TO Lake DO
            BEGIN
              WRITE (Medium, ROUND ((Hist [Species, Years])/1000):7, '000');
            END;
          WRITELN (Medium);
        END;
      WRITELN (Medium);
      IF NOT PrintOut THEN
        BEGIN
          WRITE ('Do you want a printout of these values ? (y/n): ');
          READLN (Answer);
        END;
    END;
  END;

```

```

    IF (Answer = 'Y') OR (Answer = 'y') THEN
        BEGIN
            CLOSE (Medium);
            PrintOut := True;
            PrintFile;
            CLOSE (Medium);
        END;
    END;
ELSE BEGIN
    GOTDXY (0,23);
    WRITE ('Press <RETURN> to continue: ');
    READLN (Reply);
    CLOSE (Medium)
END;
END;

PROCEDURE GetNumber2 (VAR Number : Real);

TYPE
    SetOfChar = SET OF Char;

VAR
    S : ARRAY [0..15] OF Char;
    Num, SubNum : Real;
    I, N, Place : Integer;
    OKSet : SET OF Char;
    CH : Char;
    Decimal, Good : Boolean;

FUNCTION GetChar (OKSet : SetOfChar) : Char;

BEGIN
    REPEAT
        READ (Keyboard, CH);
        IF EOLN (Keyboard) THEN CH := CHR(13);
        Good := CH IN OKSet;
        IF NOT Good THEN WRITE (CHR(7))
            ELSE IF CH IN ['0'..'9', '.'] THEN WRITE (CH);
    UNTIL Good;
    GetChar := CH;
END;

BEGIN
    Number := 0.0; Num := 0.0; SubNum := 0.0; Place := -1;
    OKSet := ['0'..'9', '.']; Decimal := False;
    FOR I := 1 TO 15 DO S[I] := ' ';
    I := 1;
    REPEAT
        IF I > 1 THEN S[I] := GetChar (OKSet + ((CHR(13), CHR(8))))
            ELSE S[I] := GetChar (OKSet);
        IF S[I] = CHR(8) THEN
            BEGIN
                WRITE (CHR(8)); WRITE (' '); WRITE (CHR(8));
                I := I - 1;
                S[I] := ' ';
            END
        ELSE I := I + 1;
    UNTIL (S[I-1] = CHR(13)) AND (I > 1);
    FOR N := 1 TO I DO
        IF S[N] IN ['0'..'9', '.'] THEN

```

```

BEGIN
  IF S(N) IN ('0'..'9') THEN Num := ORD (S(N)) - 48;
  IF S(N) = '.' THEN Decimal := True;
  IF Decimal THEN
    BEGIN
      IF S(N) = '.' THEN Num := 0;
      Place := Place + 1;
      SubNum := SubNum + (Num / PWRDFTEN(Place));
    END
  ELSE Number := (Number * 10) + Num;
END;
Number := Number + SubNum;
END;

PROCEDURE InitializeFile;

BEGIN
  WRITE (CHR(12)); GOTOXY (0,5);
  Write ('Do you want to re-initialize the Historical Stocking file (y/n)?');
  Readln (Answer);
  WHILE (Answer = 'y') OR (Answer = 'Y') DO
    BEGIN
      WRITE ('What is the year of the most recent stocking data? ');
      GetNumber2 (Yer);
      Yr := TRUNC (Yer);
      REWRITE (HistFile, 'STOCK:HISTORIC.DATA');
      STR (Yr, Year);
      WRITELN (HistFile, Year);
      WRITELN ('Enter the numbers of fish stocked: ');
      WRITELN;
      FOR Species := Brook TO Lake DO
        BEGIN
          CASE Species OF
            Brook   : SppLbl := 'Brook Trout ';
            Brown   : SppLbl := 'Brown Trout';
            Rainbow : SppLbl := 'Rainbow Trout';
            Chinook : SppLbl := 'Chinook Salmon';
            Coho    : SppLbl := 'Coho Salmon';
            Lake    : SppLbl := 'Lake Trout';
          END;
          FOR Years := Yr DOWNTD (Yr - 9) DO
            BEGIN
              WRITE (SppLbl, ' ', Years, ' : ');
              GetNumber2 (Number);
              INumber := ROUND (Number / 1000);
              STR (INumber, HistNumber);
              WRITE (HistFile, HistNumber, '000 ');
            END;
            WRITELN (HistFile);
          END;
        CLOSE (HistFile, LOCK);
        Answer := 'N';
        WRITE (CHR(12));
        PrintOut := False; PrintFile;
      END; (*WHILE LOOP*)
    END;
  END;

```

```

BEGIN

WRITE ('Brook trout - ',(Yr+1),': ');
GetNumber2 (Number);
Hist [Brook, (Yr+1)] := Number;
Hist [Brook, (Yr-1)] := 0;
WRITE ('Brown trout - ',(Yr+1),': ');
GetNumber2 (Number);
Hist [Brown, (Yr+1)] := Number;
Hist [Brown, (Yr-2)] := 0;
WRITE ('Rainbow trout - ',(Yr+1),': ');
GetNumber2 (Number);
Hist [Rainbow, (Yr+1)] := Number;
Hist [Rainbow, (Yr-2)] := 0;
WRITE ('Chinook salmon - ',(Yr+1),': ');
GetNumber2 (Number);
Hist [Chinook, (Yr+1)] := Number;
Hist [Chinook, (Yr-3)] := 0;
WRITE ('Coho salmon - ',(Yr+1),': ');
GetNumber2 (Number);
Hist [Coho, (Yr+1)] := Number;
Hist [Coho, (Yr-1)] := 0;
WRITE ('Lake trout - ',(Yr+1),': ');
GetNumber2 (Number);
Hist [Lake, (Yr+1)] := Number;
Hist [Lake, (Yr-9)] := 0;
END;

PROCEDURE CopyFile;

BEGIN
  REWRITE (HistFile, 'STOCK:HISTORIC.DATA');
  Yr0 := Yr - 2;
  Yr1 := Yr - 1;
  Yr2 := Yr + 0;
  STR (Yr0, Year );
  STR (Yr1, Year1);
  STR (Yr2, Year2);
  WRITE (HistFile, Year, ' ');
  WRITE (HistFile, Year1, ' ');
  WRITELN (HistFile, Year2);
  FOR Species := Brook TO Lake DO
    BEGIN
      FOR Years := Yr DOWNTO (Yr - 9) DO
        BEGIN
          Number := Hist (Species, Years);
          INumber := ROUND (Number / 1000);
          STR (INumber, HistNumber);
          WRITE (HistFile, HistNumber, '000 ');
        END;
        WRITELN (HistFile);
      END;
      CLOSE (HistFile, LOCK);
    END;
  END;

BEGIN
  InitializeFile;
  RESET (HistFile, 'STOCK:HISTORIC.DATA');

```

```

READLN (HistFile, Yr);
WRITELN ('The most recent year of stocking data in the file is ',Yr);
WRITELN ('Do you want to update the historical file to ');
WRITE ('include new stocking and/or proposed stocking data? (y/n) :');
READLN (Answer);
IF (Answer = 'N') OR (Answer = 'n') THEN CLOSE (HistFile)
ELSE
  BEGIN
    FOR Species := Brook TO Lake DO
      FOR Years := Yr DOWNTO (Yr - 9) DO
        BEGIN
          READ (HistFile, Number);
          Hist [Species, Years] := Number;
        END;
      CLOSE (HistFile);
      WRITE ('Do you want to enter stocking data for ',Yr + 1,' ? (y/n): ');
      READLN (Ansi);
      IF (Ansi = 'y') OR (Ansi = 'Y') THEN
        BEGIN
          WRITELN ('Enter the numbers of fish stocked: ');
          GetData;
          Yr := Yr + 1;
        END;
      FOR Yr := (Yr) TO (Yr + 1) DO
        BEGIN
          WRITELN ('Enter the PROPOSED numbers of fish to be stocked: ');
          GetData;
        END;
      CopyFile;
    END;
  WRITE (CHR(12));
  PrintOut := False; PrintFile;
  WRITE (CHR(12));
  SETCHAIN ('STOCK:STARTUP')
END. (*Program*)

```

PROGRAM Rationale;

USES CHAINSTUFF;

TYPE

Spp = (Brook, Brown, Rainbow, Chinook, Coho, Lake);
Area = (GreenBay, EastDoor, KewMan, ShebOz, Milken);
Fishery = (Trolling, Pier, Shore, Stream);
FacRes = (Boats, FishingPiers, ShoreLine, AnglingStream, SnaggingStream);
SppObj = ARRAY [Brook..Lake] OF Real;
Ranking = ARRAY [Brook..Coho, GreenBay..Milken] OF Real;
FshryFile = ARRAY [Brook..Coho, Trolling..Stream] OF Real;
FacMeasures = ARRAY [GreenBay..Milken, Boats..SnaggingStream] OF Real;
HistoricalFile = ARRAY [Brook..Lake, 1973..2000] OF Real;

VAR

Species : Spp;
Zone : Area;
FishryType : Fishery;
Res : FacRes;
CatchObj, NwCtcObj, StockFgl, StockObj, StockYlg : SppObj;
AreaRank, Catch, FglS, Stock, TentativeRanking, Ylgs : Ranking;
FishryRank : FshryFile;
ResData : FacMeasures;
Hist : HistoricalFile;
FacFile, HistFile, PercentFile, ResFile, ResultFile : Text;
AreaCatch, LakeTroutAvail, Number, Objective, ResNumber, SppCatch : Real;
BrookObj, BrownObj, RainbowObj, ChinookObj, CohoObj : Real;
ChinWt1, ChinWt2, CohoWt1, CohoWt2, PAR, SFI : Real;
Latest, Month, Yaer, Year, Year1, Year2, Year3, YearR, Years, Yr : Integer;
Value, ZoneLbl : String;
CohoBB, NorthZone : Char;
Printer : Interactive;

PROCEDURE GetResFile;

BEGIN

RESET (ResFile, 'STOCK:RESOURCE.DATA');
READ (ResFile, YearR);
FOR Zone := GreenBay TO Milken DO
 FOR Res := Boats TO SnaggingStream DO
 BEGIN
 READ (ResFile, ResNumber);
 ResNumber := (ResNumber / 10000);
 ResData [Zone, Res] := ResNumber;
 END;
CLOSE (ResFile);
END;

PROCEDURE GetPercentFile;

BEGIN

RESET (PercentFile, 'STOCK:PCENT.DATA');
BEGIN
FOR Species := Brook TO Chinook DO
 BEGIN
 FOR Zone := GreenBay TO Milken DO
 BEGIN
 READ (PercentFile, AreaCatch);

```

AreaCatch := (AreaCatch / 10000);
AreaRank [Species, Zone] := AreaCatch;
END;
FOR FishryType := Trolling TO Stream DO
BEGIN
  READ (PercentFile, SppCatch);
  SppCatch := (SppCatch / 10000);
  FishryRank [Species, FishryType] := SppCatch;
END;
END;
CLOSE (PercentFile);
END;

PROCEDURE GetHistoricFile;

VAR
  HistNumber : Real;

BEGIN
  RESET (HistFile, 'STOCK:HISTORIC.DATA');
  READ (HistFile, Year1, Year2, Year3);
  Latest := Year3;
  FOR Species := Brook TO Lake DO
    FOR Years := (Latest) DOWNTO (Latest - 9) DO
      BEGIN
        READ (HistFile, HistNumber);
        Hist [Species, Years] := HistNumber;
      END;
  CLOSE (ResFile);
END;

PROCEDURE RankSpecies;

VAR
  Rank : Real;

BEGIN
  FOR Species := Brook TO Chinook DO
    FOR Zone := GreenBay TO MilKen DO
      BEGIN
        Rank := 0;
        Rank := (AreaRank [Species, Zone] * 0.4);
        Rank := Rank + (ResData [Zone, Boats] * FishryRank [Species, Trolling]
          * 0.6);
        Rank := Rank + (ResData [Zone, FishingPiers] * FishryRank
          [Species, Pier] * 0.6);
        Rank := Rank + (ResData [Zone, ShoreLine] * FishryRank [Species, Shore]
          * 0.6);
        CASE Species OF
          Brook, Rainbow : Rank := Rank + (ResData [Zone, AnglingStream] *
            FishryRank [Species, Stream] * 0.6);
          Brown, Chinook : Rank := Rank + (ResData [Zone, SnaggingStream] *
            FishryRank [Species, Stream] * 0.6);
        END; (*CASE*)
        TentativeRank [Species, Zone] := Rank;
      END;
  END;
END;

PROCEDURE GetNumber2 (VAR Number : Real);

```

```

TYPE
    SetOfChar = SET OF Char;

VAR
    S : ARRAY [0..15] OF Char;
    Num, SubNum : Real;
    I, N, Place : Integer;
    OKSet : SET OF Char;
    CH : Char;
    Decimal, Good : Boolean;

FUNCTION GetChar (OKSet : SetOfChar) : Char;

BEGIN
    REPEAT
        READ (Keyboard, CH);
        IF EOLN (Keyboard) THEN CH := CHR(13);
        Good := CH IN OKSet;
        IF NOT Good THEN WRITE (CHR(7))
            ELSE IF CH IN ['0'..'9', '.'] THEN WRITE (CH);
    UNTIL Good;
    GetChar := CH;
END;

BEGIN
    Number := 0.0; Num := 0.0; SubNum := 0.0; Place := -1;
    OKSet := ['0'..'9', '.']; Decimal := False;
    FOR I := 1 TO 15 DO S[I] := ' ';
    I := 1;
    REPEAT
        IF I > 1 THEN S[I] := GetChar (OKSet + ((CHR(13), CHR(8))))
            ELSE S[I] := GetChar (OKSet);
        IF S[I] = CHR(8) THEN
            BEGIN
                WRITE (CHR(8)); WRITE (' '); WRITE (CHR(8));
                I := I - 1;
                S[I] := ' ';
            END
        ELSE I := I + 1;
    UNTIL (S[I-1] = CHR(13)) AND (I > 1);
    FOR N := 1 TO I DO
        IF S[N] IN ['0'..'9', '.'] THEN
            BEGIN
                IF S[N] IN ['0'..'9'] THEN Num := ORD (S[N]) - 48;
                IF S[N] = '.' THEN Decimal := True;
                IF Decimal THEN
                    BEGIN
                        IF S[N] = '.' THEN Num := 0;
                        Place := Place + 1;
                        SubNum := SubNum + (Num / PWR10(Place));
                    END
                ELSE Number := (Number * 10) + Num;
            END;
        Number := Number + SubNum;
    END;
END;

PROCEDURE GetInput;

```

VAR

```
Tot : Real;
Console : interactive;
```

```
BEGIN
WRITELN (CHR(12));GOTOXY (0,10);
WRITELN ('What is the total catch objective for sport fish in the Wisconsin');
WRITELN ('waters of Lake Michigan exclusive of lake trout and moored ');
WRITE ('boat angler catch?: ');
GetNumber2 (Objective); WRITELN;
IF Objective < 50000.0 THEN Objective := 50000.0;
IF Objective > 1000000.0 THEN Objective := 1000000.0;
WHILE Tot (<) 100 DO
  BEGIN
    WRITELN (CHR (12)); GOTOXY (0, 10);
    WRITELN ('List the percent of the objective preferred for each species:');
    Tot := 0.00;
    WRITELN ('They must sum to 100!'); WRITELN;
    WRITE (' Brook : '); GetNumber2 (CatchObj [Brook]); WRITELN;
    Tot := Tot + CatchObj [Brook];
    StockObj [Brook] := (CatchObj [Brook] / 100) * Objective * 23.40;
    WRITE (' Brown : '); GetNumber2 (CatchObj [Brown]); WRITELN;
    Tot := Tot + CatchObj [Brown];
    StockObj [Brown] := (CatchObj [Brown] / 100) * Objective * 19.34;
    WRITE (' Rainbow : '); GetNumber2 (CatchObj [Rainbow]); WRITELN;
    Tot := Tot + CatchObj [Rainbow];
    StockObj [Rainbow] := (CatchObj [Rainbow] / 100) * Objective * 27.40;
    WRITE (' Chinook : '); GetNumber2 (CatchObj [Chinook]); WRITELN;
    Tot := Tot + CatchObj [Chinook];
    StockObj [Chinook] := (CatchObj [Chinook] / 100) * Objective * 10.13;
    WRITE (' Coho : '); GetNumber2 (CatchObj [Coho]); WRITELN;
    Tot := Tot + CatchObj [Coho];
    StockObj [Coho] := (CatchObj [Coho] / 100) * Objective * 3.83
  END;
WRITELN (CHR(12)); GOTOXY (0,10);
WRITE ('What is the current year?: ');
GetNumber2 (Number); Yr := TRUNC (Number); WRITELN;
WRITE ('What is the current month? (Jan = 1, Dec = 12) : ');
GetNumber2 (Number); Month := TRUNC (Number); WRITELN;
IF (Month < 6) THEN Yr := Yr - 1;
END;
```

```
PROCEDURE GetCohoInfo;
```

```
BEGIN
WRITELN (CHR(12)); GOTOXY (0, 10);
WRITELN ('How many lake trout are available to be stocked in the Wisconsin');
WRITE ('waters of Lake Michigan in ', Yr + 2, '? ');
GetNumber2 (LakeTroutAvail); WRITELN;
StockObj [Lake] := LakeTroutAvail; WRITELN;
REPEAT
  GOTOXY (0,15);
  WRITELN ('Fish weights must be greater than 0.0. ');
  WRITE ('What was the weight of age I+ chinook salmon in ', Yr, ': ');
  GetNumber2 (ChinWt1); WRITELN;
  WRITE ('What was the weight of age I+ chinook salmon in ', Yr - 1, ': ');
  GetNumber2 (ChinWt2); WRITELN;
  WRITE ('What was the weight of age II+ coho salmon in ', Yr, ': ');
  GetNumber2 (CohoWt1); WRITELN;
  WRITE ('What was the weight of age II+ coho salmon in ', Yr - 1, ': ');
  GetNumber2 (CohoWt2); WRITELN;
```

```

UNTIL (ChinWt1 > 0.0) AND (ChinWt2 > 0.0) AND (CohoWt1 > 0.0)
  AND (CohoWt2 > 0.0)
END;

PROCEDURE RankCoho;

VAR
i, n : Integer;
Options : SET OF Char;

BEGIN
Options := ['A'..'D', 'a'..'d'];
WRITELN (CHR(12)); GOTOXY (0, 10);
REPEAT
  WRITE ('Do you want to stock coho salmon in Green Bay? (y/n): ');
  READLN (CohoGB);
  WRITELN;
UNTIL (CohoGB = 'Y') OR (CohoGB = 'y') OR (CohoGB = 'N') OR (CohoGB = 'n');
REPEAT
  WRITELN ('What is the northern most zone on Lake Michigan where coho ');
  WRITELN ('are to be stocked?');
  WRITELN;
  WRITELN (' A) East Door County');
  WRITELN (' B) Kewaunee - Manitowoc');
  WRITELN (' C) Sheboygan - Ozaukee');
  WRITELN (' D) Milwaukee - Kenosha');
  WRITELN;
  WRITE ('Choose one : ');
  READLN (NorthZone);
UNTIL (NorthZone IN Options);
CASE NorthZone OF
  'A', 'a' : n := 5;
  'B', 'b' : n := 4;
  'C', 'c' : n := 3;
  'D', 'd' : n := 2
END;
IF (CohoGB = 'N') OR (CohoGB = 'n') THEN n:=n - 1;
i := n;
IF (CohoGB = 'Y') OR (CohoGB = 'y') THEN
  BEGIN
    TentativeRank [Coho, GreenBay] := (2 / (n*n+n)) * (n-(i-1));
    i := i - 1
  END
ELSE TentativeRank [Coho, GreenBay] := 0.00;
FOR Zone := MilKen DOWNT0 EastDoor DO
  BEGIN
    TentativeRank [Coho, Zone] := (2 / (n*n+n)) * (n-(i-1));
    i := i - 1;
    IF (i < 0) THEN TentativeRank [Coho, Zone] := 0.00;
  END;
END;

PROCEDURE CheckForagePressure;

VAR
LostPressure, NewPressure, TotalPressure : Real;
ChinEquiv : SppObj;

BEGIN
Yaer := Latest + 1;

```

```

LostPressure := 0;
NewPressure := 0;
TotalPressure := 0;
ChinEquiv (Brook) := 0.53; ChinEquiv (Brown) := 0.80;
ChinEquiv (Rainbow) := 0.80; ChinEquiv (Coho) := 0.53;
ChinEquiv (Chinook) := 1.00; ChinEquiv (Lake) := 0.88;
FOR Species := Brook TO Lake DO
  BEGIN
    CASE Species OF
      Brook, Coho : Year := Yaer - 2;
      Brown, Rainbow : Year := Yaer - 3;
      Chinook : Year := Yaer - 4;
      Lake : Year := Yaer - 10
    END; (* CASE *)
    LostPressure := LostPressure + (Hist (Species, Year) * ChinEquiv (Species));
    NewPressure := NewPressure + (StockObj(Species) * ChinEquiv (Species));
  END;
FOR Species := Brook TO Lake DO
  FOR Year := (Latest) DOWNTO (Latest - 9) DO
    BEGIN
      TotalPressure := TotalPressure + (Hist (Species, Year)
        * ChinEquiv (Species));
    END;
  SFI := (ChinWt1 + CohoWt1) / (ChinWt2 + CohoWt2);
  PAR := (24293610.0 - (LakeTroutAvail * 0.88) - (TotalPressure - LostPressure))
    / (NewPressure - (LakeTroutAvail * 0.88));
  IF (SFI + PAR) > 2 THEN
    BEGIN
      PAR := 1;
      SFI := 1
    END;
  END;

PROCEDURE CalculateStocking;

BEGIN
FOR Species := Brook TO Coho DO
  BEGIN
    StockObj (Species) := StockObj (Species) * SFI * PAR;
  END;
FOR Species := Brook TO Coho DO
  FOR Zone := GreenBay TO Milken DO
    BEGIN
      Stock (Species,Zone) := TentativeRank(Species,Zone) * StockObj(Species);
    END;
  END;

PROCEDURE MinRequirements;

VAR
Zone0 : Area;
Deficit, Minimum, Proportion : Real;

BEGIN
FOR Species := Rainbow TO Coho DO
  BEGIN
    Minimum := 0.10 * StockObj (Species);
    CASE Species OF
      Rainbow : Zone0 := GreenBay;
      Chinook : Zone0 := EastDoor;
    
```

```

    Coho    : Zone0 := Sheb0z
END;
IF (Stock (Species, Zone0)) < Minimum THEN
  BEGIN
    Deficit := Minimum - Stock (Species, Zone0);
    Proportion := Deficit / (StockObj (Species) - Stock (Species, Zone0));
    FOR Zone := GreenBay TO MilKen DO
      BEGIN
        Stock (Species, Zone) := Stock (Species, Zone) -
          (Stock (Species, Zone) * Proportion);
      END;
    Stock (Species, Zone0) := Minimum;
  END; (* IF *)
END; (* FOR *)
END;

```

```

PROCEDURE StoreResults;

```

```

VAR
  Number : Real;
  ResultNumber : String;
  SObj, BkObj, BnObj, RnObj, ChObj, CoObj, LkObj, Schin1, Schin2, SCohol,
  SCoho2, SYr : String;
  INumber : Integer;

```

```

BEGIN
  REWRITE (ResultFile, 'STOCK:RESULTS.DATA');
  INumber := ROUND (Objective / 1000);
  STR (INumber, SObj);
  INumber := ROUND (CatchObj [Brook] * 10);
  STR (INumber, BkObj);
  INumber := ROUND (CatchObj [Brown] * 10);
  STR (INumber, BnObj);
  INumber := ROUND (CatchObj [Rainbow] * 10);
  STR (INumber, RnObj);
  INumber := ROUND (CatchObj [Chinook] * 10);
  STR (INumber, ChObj);
  INumber := ROUND (CatchObj [Coho] * 10);
  STR (INumber, CoObj);
  INumber := ROUND (LakeTroutAvail / 1000);
  STR (INumber, LkObj);
  STR (Yr, SYr);
  INumber := ROUND (ChinWt1 * 100);
  STR (INumber, Schin1);
  INumber := ROUND (ChinWt2 * 100);
  STR (INumber, Schin2);
  INumber := ROUND (CohoWt1 * 100);
  STR (INumber, SCohol);
  INumber := ROUND (CohoWt2 * 100);
  STR (INumber, SCoho2);
  WRITE (ResultFile, SObj, ' ', BkObj, ' ', BnObj, ' ', RnObj, ' ', ChObj, ' ');
  WRITE (ResultFile, CoObj, ' ', LkObj, '000 ', SYr, ' ', Schin1, ' ', Schin2, ' ');
  WRITELN (ResultFile, SCohol, ' ', SCoho2);
  FOR Species := Brook TO Coho DO
    BEGIN
      FOR Zone := GreenBay TO MilKen DO
        BEGIN
          Number := Stock (Species, Zone);
          IF Number > 9999 THEN
            BEGIN

```

```

      INumber := ROUND (Number / 1000);
      STR (INumber, ResultNumber);
      WRITE (ResultFile, ResultNumber, '000 ');
    END
  ELSE
    BEGIN
      INumber := ROUND (Number / 100);
      STR (INumber, ResultNumber);
      WRITE (ResultFile, ResultNumber, '00 ');
    END;
  END;
  WRITELN (ResultFile);
END;
CLOSE (ResultFile, LOCK)
END;

BEGIN
  GetResFile;
  GetPercentFile;
  GetHistoricFile;
  RankSpecies;
  GetInput;
  GetCohoInfo;
  RankCoho;
  CheckForagePressure;
  CalculateStocking;
  MinRequirements;
  StoreResults;
  SETCHAIN ('STOCK:RESULTS')
END. (*Program*)

```

PROGRAM Results;

USES CHAINSTUFF;

TYPE

Spp = (Brook, Brown, Rainbow, Chinook, Coho, Lake);
Area = (GreenBay, EastDoor, KewMan, ShebOz, MilKen);
Fishery = (Trolling, Pier, Shore, Stream);
SppObj = ARRAY [Brook..Lake] OF Real;
Ranking = ARRAY [Brook..Coho, GreenBay..MilKen] OF Real;
FishryFile = ARRAY [Brook..Coho, Trolling..Stream] OF Real;

VAR

Species : Spp;
Zone : Area;
FishryType : Fishery;
FCost, YCost : SppObj;
CatchObj, NwCtcObj, StockFgl, StockObj, StockYlg : SppObj;
Catch, Fgl, Stock, Ylgs : Ranking;
FishryRank : FishryFile;
PercentFile, ResultsFile : Text;
LakeTrout, Objective, Number, Total, TotalCost, Value : Real;
ChinWt1, ChinWt2, CohoWt1, CohoWt2 : Real;
PrinterOn, Year, Years : Integer;
FishryLbl, SppLbl, ZoneLbl : String;
Answer, CohoGB, CohoNorth, Response : Char;
Medium : Interactive;

PROCEDURE DrawLine;

BEGIN

WRITE (Medium, '-----');
WRITELN (Medium, '-----');

END;

PROCEDURE GetResultsFile;

VAR

BrookT, BrownT, RainbowT, ChinookS, CohoS : Real;
Message : String;

BEGIN

WRITELN (CHR(12)); GOTOXY (0,10);
WRITELN ('Reading data files...');
RESET (ResultsFile, 'STOCK:RESULTS.DATA');
READ (ResultsFile, Objective, BrookT, BrownT, RainbowT, ChinookS, CohoS,
LakeTrout, Year, ChinWt1, ChinWt2, CohoWt1, CohoWt2);
FOR Species := Brook TO Coho DO
FOR Zone := GreenBay TO MilKen DO
BEGIN
READ (ResultsFile, Number);
Stock [Species, Zone] := Number;
END;
CLOSE (ResultsFile);
Objective := Objective * 1000; CatchObj [Brook] := BrookT / 10;
CatchObj [Brown] := BrownT / 10; CatchObj [Rainbow] := RainbowT / 10;
CatchObj [Chinook] := ChinookS / 10; CatchObj [Coho] := CohoS / 10;
ChinWt1 := ChinWt1 / 100; ChinWt2 := ChinWt2 / 100;
CohoWt1 := CohoWt1 / 100; CohoWt2 := CohoWt2 / 100;

END;

PROCEDURE GetFisheryPercents;

BEGIN

RESET (PercentFile, 'STOCK:PCENT.DATA');

FOR Species := Brook TO Coho DO

BEGIN

FOR Zone := GreenBay TO Milken DO

BEGIN

READ (PercentFile, Number);

END;

FOR FishryType := Trolling TO Stream DO

BEGIN

READ (PercentFile, Number);

Number := (Number / 10000);

FishryRank [Species, FishryType] := Number;

END;

END;

CLOSE (PercentFile)

END;

PROCEDURE CalculateStocking;

VAR

CatchRatio, FglRatio, YlgRatio : SppObj;

BEGIN

CatchRatio [Brook] := 0.0427; CatchRatio [Brown] := 0.0517;

CatchRatio [Rainbow] := 0.0365; CatchRatio [Coho] := 0.2609;

CatchRatio [Chinook] := 0.0987; YlgRatio [Brook] := 0.746;

YlgRatio [Brown] := 0.502; YlgRatio [Rainbow] := 0.466;

YlgRatio [Chinook] := 0.000; YlgRatio [Coho] := 1.000;

FglRatio [Brook] := 0.254; FglRatio [Brown] := 0.498;

FglRatio [Rainbow] := 0.534; FglRatio [Coho] := 0.000;

FglRatio [Chinook] := 1.000;

FOR Species := Brook TO Coho DO

BEGIN

StockObj [Species] := 0.00; NwCtcObj [Species] := 0.00;

StockYlg [Species] := 0.00; StockFgl [Species] := 0.00

END;

FOR Species := Brook TO Coho DO

FOR Zone := GreenBay TO Milken DO

BEGIN

Catch [Species,Zone] := Stock [Species, Zone] * CatchRatio [Species];

Ylgs [Species,Zone] := Stock [Species, Zone] * YlgRatio [Species];

Fgl [Species,Zone] := Stock [Species, Zone] * FglRatio [Species];

StockObj [Species] := StockObj [Species] + Stock [Species, Zone];

NwCtcObj [Species] := NwCtcObj [Species] + Catch [Species, Zone];

StockYlg [Species] := StockYlg [Species] + Ylgs [Species, Zone];

StockFgl [Species] := StockFgl [Species] + Fgl [Species, Zone]

END;

END;

PROCEDURE ParameterList;

BEGIN

WRITELN (Medium, 'Parameters used in this run.');

WRITELN (Medium);

WRITELN (Medium, 'Catch objective : ', ROUND(Objective / 1000), '000');

```

WRITELN (Medium, 'Preference for each species (%) :');
WRITELN (Medium, '    Brook : ', ROUND(CatchObj [Brook]));
WRITELN (Medium, '    Brown : ', ROUND(CatchObj [Brown]));
WRITELN (Medium, '    Rainbow : ', ROUND(CatchObj [Rainbow]));
WRITELN (Medium, '    Chinook : ', ROUND(CatchObj [Chinook]));
WRITELN (Medium, '    Coho : ', ROUND(CatchObj [Coho]));
WRITELN (Medium);
WRITELN (Medium, 'Lake trout available for stocking : ',
    ROUND (LakeTrout/1000), '000');
WRITELN (Medium, 'Weight of chinook salmon in ', Year, ' : ', ChinWt1:6:2);
WRITELN (Medium, 'Weight of chinook salmon in ', Year-1, ' : ', ChinWt2:6:2);
WRITELN (Medium, 'Weight of coho salmon in ', Year, ' : ', CohoWt1:6:2);
WRITELN (Medium, 'Weight of coho salmon in ', Year-1, ' : ', CohoWt2:6:2);
WRITELN (Medium);
IF PrinterOn = 0 THEN
    BEGIN
        WRITELN (Medium); WRITELN (Medium);
        WRITE (Medium, 'Press <RETURN> to continue: ');
        READLN (Answer);
    END;
END;

PROCEDURE Table1a;

BEGIN
    WRITE (Medium, 'Table 1. Recommended stocking levels for ', Year + 2);
    WRITELN (Medium, ' (numbers of fish).'); WRITELN (Medium);
    WRITE (Medium, '-----'); DrawLine;
    WRITE (Medium, 'Management Zone':21, 'Brook':10, 'Brown':10, 'Rainbow':10);
    WRITELN (Medium, 'Chinook':10, 'Coho':10, 'Total':10);
    WRITE (Medium, '-----'); DrawLine;
    FOR Zone := GreenBay TO MilKen DO
        BEGIN
            CASE Zone OF
                GreenBay : ZoneLbl := 'Green Bay          ';
                EastDoor  : ZoneLbl := 'East Door County  ';
                KewMan    : ZoneLbl := 'Kewaunee to Manitowoc';
                ShebOz    : ZoneLbl := 'Sheboygan to Ozaukee';
                MilKen    : ZoneLbl := 'Milwaukee to Kenosha';
            END;
            Total := 0;
            WRITE (Medium, ZoneLbl : 21); WRITELN (Medium);
            WRITE (Medium, 'Yearlings':21);
            FOR Species := Brook TO Coho DO
                BEGIN
                    Total := Total + Ylgs [Species, Zone];
                    WRITE (Medium, (ROUND (Ylgs [Species, Zone] / 100)) : 8, '00');
                END;
            WRITELN (Medium, (ROUND (Total / 1000)):7, '000');
            Total := 0; WRITE (Medium, 'Fingerlings':21);
            FOR Species := Brook TO Coho DO
                BEGIN
                    Total := Total + FglS [Species, Zone];
                    WRITE (Medium, (ROUND (FglS [Species, Zone] / 100)) : 8, '00');
                END;
            WRITELN (Medium, (ROUND (Total / 1000)):7, '000')
        END;
    WRITE (Medium, ':21, '-----');
    WRITELN (Medium, '-----');
END;

```

```

PROCEDURE Table1;

BEGIN
Table1a;
WRITE (Medium, 'Total Yearlings':21);
Total := 0;
FOR Species := Brook TO Coho DO
  BEGIN
    IF StockYlg [Species] < 999999.0 THEN
      BEGIN
        WRITE (Medium, (ROUND (StockYlg [Species] / 100)) : 8, '00');
      END
    ELSE WRITE (Medium, (ROUND (StockYlg [Species] / 1000)) : 7, '000');
    Total := Total + StockYlg [Species];
  END;
WRITELN (Medium, (ROUND (Total / 1000)):7, '000');
WRITE (Medium, 'Total Fingerlings':21); Total := 0;
FOR Species := Brook TO Coho DO
  BEGIN
    IF StockFgl [Species] < 999999.0 THEN
      BEGIN
        WRITE (Medium, (ROUND (StockFgl [Species] / 100)) : 8, '00');
      END
    ELSE WRITE (Medium, (ROUND (StockFgl [Species] / 1000)) : 7, '000');
    Total := Total + StockFgl [Species];
  END;
WRITELN (Medium, (ROUND (Total / 1000)):7, '000'); Total := 0;
WRITE (Medium, 'TOTAL':21);
FOR Species := Brook TO Coho DO
  BEGIN
    WRITE (Medium, (ROUND (StockObj [Species] / 1000)):7, '000');
    Total := Total + StockObj [Species];
  END;
WRITELN (Medium, (ROUND (Total / 1000)):7, '000');
WRITE (Medium, '-----'); DrawLine;
IF PrinterOn = 0 THEN
  BEGIN
    WRITELN (Medium);
    WRITE (Medium, 'Press <RETURN> to continue: ');
    READLN (Answer);
  END;
END;

PROCEDURE Table2a;

VAR
Diff, Total, Value : Real;

BEGIN
WRITELN (Medium); WRITE (Medium, 'Table 2. Predicted catch');
WRITELN (Medium, ' by area from the ', Year + 2, ' stocking. ');
WRITELN (Medium); WRITE (Medium, '-----'); DrawLine;
WRITE (Medium, 'Management Zone':21, 'Brook':10, 'Brown':10, 'Rainbow':10);
WRITELN (Medium, 'Chinook':10, 'Coho':10, 'Total' : 10);
WRITE (Medium, '-----'); DrawLine;
FOR Zone := GreenBay TO Milken DO
  BEGIN
    CASE Zone OF
      GreenBay : ZoneLbl := 'Green Bay';
    
```

```

EastDoor : ZoneLbl := 'East Door Co.';
KewMan   : ZoneLbl := 'Kewaunee to Manitowoc';
ShebOz   : ZoneLbl := 'Sheboygan to Ozaukee';
MilKen   : ZoneLbl := 'Milwaukee to Kenosha'
END;
WRITE (Medium, ZoneLbl : 21); Total := 0;
FOR Species := Brook TO Coho DO
  BEGIN
    WRITE (Medium, (ROUND (Catch [Species, Zone] / 100)) : 8, '00');
    Total := Total + Catch [Species, Zone];
  END;
WRITE (Medium, (ROUND (Total / 100)):8, '00');
END;
WRITE (Medium, ':21, '-----');
WRITE (Medium, '-----');
END;

PROCEDURE Table2;

VAR
  Diff, Total, Value : Real;

BEGIN
  Table2a;
  WRITE (Medium, 'TOTAL':21); Total := 0;
  FOR Species := Brook TO Coho DO
    BEGIN
      WRITE (Medium, (ROUND (NwCtcObj [Species] / 100)) : 8, '00');
      Total := Total + NwCtcObj [Species];
    END;
  WRITE (Medium, (ROUND (Total / 100)):8, '00'); WRITE (Medium, 'OBJECTIVE':21);
  FOR Species := Brook TO Coho DO
    BEGIN
      Value := (CatchObj [Species] / 100) * Objective;
      WRITE (Medium, (ROUND (Value / 100)) : 8, '00');
    END;
  WRITE (Medium, ROUND(Objective / 1000) : 7, '000');
  WRITE (Medium); WRITE (Medium, 'DIFFERENCE':21);
  FOR Species := Brook TO Coho DO
    IF CatchObj [Species] > 0 THEN
      BEGIN
        Diff := (NwCtcObj [Species] - ((CatchObj [Species] / 100) * Objective)) /
          ((CatchObj [Species] / 100) * Objective);
        WRITE (Medium, (ROUND (Diff * 100)) : 9, '%');
      END
    ELSE WRITE (Medium, '0':9, '%');
  WRITE (Medium); WRITE (Medium, '-----'); DrawLine;
  IF PrinterOn = 0 THEN
    BEGIN
      WRITE (Medium, 'Press <RETURN> to continue: ');
      READLN (Answer);
    END;
  END;

PROCEDURE Table3;

VAR
  Diff, Total, Value : Real;

BEGIN

```

```

WRITELN (' A - Parameters used in this run');
WRITELN (' B - Table 1. Summarization of recommended stocking levels. ');
WRITELN (' C - Table 2. Projected catch levels by area. ');
WRITELN (' D - Table 3. Projected catch levels by fishery. ');
WRITELN (' E - Table 4. Projected cost of stocking. ');
WRITELN (' F - Table 5. Lake trout stocking recommendations. ');
WRITELN (' G - Print all of the above. ');
WRITELN (' H - Restart the program. ');
WRITELN (' I - Quit. ');
WRITELN; WRITELN; WRITELN;
WRITE ('Select one letter and then press <RETURN> : ');
READLN (Response);
REWRITE (Medium, 'Console: ');
WRITELN (CHR(12)); GOTOXY (0,5);
CASE Response OF
  'A', 'a' : ParameterList;
  'B', 'b' : Table1;
  'C', 'c' : Table2;
  'D', 'd' : Table3;
  'E', 'e' : Table4;
  'F', 'f' : Table5;
  'G', 'g' : BEGIN
      CLOSE (Medium);
      REWRITE (Medium, 'Printer: ');
      PrinterOn := 1;
      ParameterList;
      Table1;
      Table2;
      WRITELN (Medium, CHR(12));
      Table3;
      Table4;
      Table5;
      WRITELN (Medium, CHR(12));
      PrinterOn := 0;
    END;
END;
CLOSE (Medium);
UNTIL (Response IN ['H'..'I', 'h'..'i']);
IF (Response = 'H') OR (Response = 'h') THEN SETCHAIN ('STOCK:STARTUP')
END;

BEGIN
GetResultsFile;
GetFisheryPercents;
CalculateStocking;
Menu
END. (*Program*)

```

APPENDIX 2.

Program structure by program name and procedure.

Appendix 2. Program structure by program name and procedure.

Program	Procedure	Function
Starter	RollEm	Set printer to indent.
	FirstFrame	Print program description on screen.
Starter2	GetCatchData	Get dates (years) of current past catch data.
	GetFacilityData	Get date (year) of current fishery facility data.
	GetStockData	Get date (year) of current past stocking and proposed stocking data.
	MenuHeader	Print first option of menu on screen.
	MainMenu	Print rest of menu options on screen.
	GoFish	Get option desired and run appropriate program.
PastCatchFile	PrintFile	Prints values in diskette file PAST.CATCH on the screen or on the printer.
	GetNumber	Checks character input and screens for numeric characters.
	InitializeFile	Allows user to input entire past catch file. Prompts for input.
	ReadOldFile	Reads values currently in diskette file PAST.CATCH and stores them in memory.
	CopyFile	Stores file in diskette file PAST.CATCH.
	UpdateFile	Allows user to update diskette file PAST.CATCH to include new catch data.

Appendix 2. Cont.

Program	Procedure	Function
Manipulator	PrintInfo	Stores data on diskette in diskette file PCENT.DATA.
	Manipulator	Reads diskette file PAST.CATCH, stores internally, computes 3 year average by species by area, computes percentage of catch by species by area, and by species by fishery.
FisheryFacilityFile	PrintFacFile	Prints values in diskette file FACILITY.DATA on the screen or on the printer.
	WriteResFile	Stores file on diskette in file RESOURCE.DATA.
	ComputeResults	Reads data from diskette file FACILITY.DATA, computes percentage of available facilities by area.
	GetNumber	Checks character input and screens for numeric characters.
StockingFile	InitializeFile	Allows user to input entire fishery facility file.
	PrintFile	Prints values in diskette file HISTORIC.DATA on the screen or on the printer.
	GetNumber2	Checks character input, screens for numeric characters, and converts them to real numbers.
	InitializeFile	Allows user to input past stocking levels and proposed stocking levels.
	GetData	Prompts user for input regarding number of fish stocked.
	CopyFile	Stores file on diskette in file HISTORIC.DATA.

Appendix 2. Cont.

Program	Procedure	Function
StockingFile	StockingFile	Allows user to update existing HISTORIC.DATA file to include new stocking data.
Rationale	GetResFile	Gets data from diskette file RESOURCE.DATA.
	GetPercentFile	Gets data from diskette file PCENT.DATA.
	GetHistoricFile	Gets data from diskette file HISTORIC.DATA.
	RankSpecies	Ranks areas by species for their stocking allotments using data from diskette files RESOURCE.DATA and PCENT.DATA.
	GetNumber2	Checks character input, screens for numeric characters, and converts them to real numbers.
	GetInput	Prompts user for input: catch objective, species preferences, current year and month. Computes number of fish required to stock to reach catch objective.
	GetCohoInfo	Prompts user for input: lake trout available to be stocked, coho and chinook weights for past two years.
	RankCoho	Prompts user for input: where coho are to be stocked.
	CheckForagePressure	Computes forage pressure in lake at present (from diskette file HISTORIC.DATA), and with proposed stocking. Compares to 1982 forage pressure to compute the Predation Adjustment Ratio (PAR). Uses input salmon weights to compute the Salmon Forage Indicator (SFI).

Appendix 2. Cont.

Program	Procedure	Function
Results	CalculateStocking	Computes number of fish to be stocked at each area based on zone ranking, PAR, and SFI.
	MinRequirements	Checks to make sure a minimum stocking of coho, chinook, and rainbow are stocked at sites where brood stock is taken for eggs. Redistributes stocking if the requirements are not met.
	StoreResults	Stores input data and stocking recommendations on diskette in file RESULTS.DATA.
	DrawLine	Draws a line of 60 hyphens.
	GetResultsFile	Gets data from diskette file RESULTS.DATA.
	GetFisheryPercents	Gets catch by fishery data from diskette file PCENT.DATA.
	CalculateStocking	Calculates number of yearlings and fingerlings of each species to stock and predicted catch by species by area.
	PrameterList	Prints parameters used in the run.
	Table1a	Prints detailed stocking recommendations by species by area for fingerlings and yearlings.
	Table1	Prints totals for Table 1.
Table2a	Prints predicted catch by species and area stocking from the recommended. Gives totals by species and area.	

Appendix 2. Cont.

Program	Procedure	Function
Results	Table2	Prints totals for Table 2, catch objectives, and the percentage difference between predicted catch and objective.
	Table3	Prints predicted catch by species by fishery from the recommended stocking. Gives totals by species and area.
	Table4a	Prints heading for Table 4.
	Table4	Prints cost of recommended stocking. Gives total by species and grand total cost.
	Table5	Prints recommended stocking for lake trout by management zone and total.
	Menu	Prints menu on screen and allows user to choose a table to be displayed on the screen, print out all tables on the printer, rerun the program, or quit.

APPENDIX 3.

Contents and size of the programs and files included on the diskettes.

TABLE 1. Contents and size of program code and data files on diskette STOCK:.

File Type	File Name	Size (Sectors)
System programs*	SYSTEM.APPLE	32
	SYSTEM.PASCAL	41
	SYSTEM.MISCINFO	1
	SYSTEM.LIBRARY	34
Stocking rationale program code files	SYSTEM.STARTUP	4
	STARTUP.CODE	5
	CATCHFILE.CODE	11
	MANIPULATE.CODE	4
	FACILITY.CODE	9
	STOCKING.CODE	9
	RATIONAL.CODE	14
RESULTS.CODE	20	
Data files	PAST.CATCH	3
	PCENT.DATA	1
	HISTORIC.DATA	1
	FACILITY.DATA	1
	RESOURCE.DATA	1
	RESULTS.DATA	1

*Apple Pascal files written by Apple Computer, Inc.

TABLE 2. Contents and size of the program text files on diskette STOCK1:.

Program Name	File Name	Size (Sectors)
Starter	SYSTART.TEXT	6
Starter2	STARTUP.TEXT	8
PastCatchFile	CATCHFILE.TEXT	22
Manipulator	MANIPULATE.TEXT	8
FisheryFacilityFile	FACILITY.TEXT	18
StockingFile	STOCKING.TEXT	16
Rationale	RATIONAL.TEXT	28
Results	RESULTS.TEXT	34

TABLE 3. Stocking rationale program cross reference listing.

Program Code File	Program Text File
Diskette: STOCK:	Diskette: STOCK1:
SYSTEM.STARTUP	SYSTART.TEXT
STARTUP.CODE	STARTUP.TEXT
CATCHFILE.CODE	CATCHFILE.TEXT
MANIPULATE.CODE	MANIPULATE.TEXT
FACILITY.CODE	FACILITY.TEXT
STOCKING.CODE	STOCKING.TEXT
RATIONAL.CODE	RATIONAL.TEXT
RESULTS.CODE	RESULTS.TEXT

1912

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

APPENDIX 4.

Procedure for making program changes.

This section assumes you have a working knowledge of the Pascal language and some experience or training with the Apple Pascal software system. In addition to the materials necessary to run the stocking rationale program, an additional disk drive and the Apple Pascal system (version 1.1) software package will be needed to make any changes or modifications in the programs.

First, decide what type of change you want to make and its appropriate location(s) in the program(s). An outline of the programs and procedures is given in Appendix 2. A complete listing of all the programs is given in Appendix 1. Once the exact modifications are decided upon that will effect the desired changes, determine which textfile (on diskette STOCK1:) contains the program (Append. 3).

Boot the Pascal software package with Pascal diskette APPLE1: in disk drive #1 and diskette STOCK1: in disk drive #2 (turn the system power on or press [RESET]). The screen should have a welcome statement with the list of commands across the top of the screen. Press the letter "F" for Filer. A new list of commands should appear across the top of the screen. Press "D" for Date. Enter today's date in the same format as shown on the screen. Once you have the date current, press "G" for Get. The system will ask if you want to erase the old file. If you no longer have a need for the old file, press "y" for yes. The next question is what file do you want loaded? Type in the exact textfile name as it appears in Appendix 3 using STOCK1: as a prefix.

For example, suppose you want to update the program to reflect changes in the hatchery cost of fingerling and yearling fish. Specifically, suppose the cost of raising and distributing brown trout increased by 10%: fingerlings increased from \$0.102 to \$0.110 each and yearlings increased from \$0.243 to \$0.267 each. By looking in Appendix 1 on program Results you find the costs are coded in procedure Table4a. Appendix 3 shows program Results is stored on diskette STOCK1: in file RESULTS.TEXT. Thus to make the change, the text file must be loaded into the Editor using the Filer command Get. The name of the file to be loaded is "STOCK1:RESULTS.TEXT".

The screen should respond "TEXT FILE LOADED". If it does, press "Q" for Quit and return to the command level. Now press "E" for Editor and the system will load the textfile into the Editor worksheet. Locate the area(s) to make the change(s) using the F)ind command, or scroll to the area(s) using the cursor commands CNTL L, CNTL O, left arrow, or right arrow. Make your changes using the eX)change, I)nsert, or D)delete commands (see Apple Pascal Operating System Reference Manual pages 70-133 for details).

To make the changes in our example program, press F for F)ind. The response will be "FIND[1]: ". Type in "/Table4a/" to locate the procedure where the changes are to be made. The cursor should reappear in the middle of the screen over the word "Table4a". Use the cursor control command CNTL L (by pressing the CNTL key and the L key simultaneously) to move the cursor down the screen to the line below 'BEGIN'. The line should appear as follows:

```
FCost [Brook]:= 0.089; FCost [Brown]:= 0.102; FCost [Rainbow]:= 0.099;
```

Use the right arrow to move the cursor over to "FCost [Brown] = 0.102", the cost value of fingerling brown trout. Put the cursor on the '1' in the '0.102'. Press X for eX)change and type '110'. Now press CNTL C to get back to the Editor mode. The line should now appear as follows:

```
FCost [Brook]:= 0.089; FCost [Brown]:= 0.110; FCost [Rainbow]:= 0.099;
```

To make the change in the cost of yearling brown trout, move the cursor down two lines (using CNTL L) and to the left (using the left arrow key) to 'YCost [Brown] = 0.243', the cost value of yearling brown trout. Place the cursor on the '2' in 0243'. Press X for eX)change and type '267' to change the cost value. Press CNTL C to return to the Editor mode.

Once the changes have been made, press "Q" for Q)uit and "U" for Update. Remove diskette STOCK1: from disk drive 2 and in its place insert Apple Pascal diskette APPLE0:. With the system in command mode, press "C" for Compile. The system should compile your edited text version and save a copy of the compiled version on diskette APPLE1:. After this is done, remove diskette APPLE0: from disk drive 2 and replace it with STOCK1:. Enter the Filer mode by pressing "F". Press "T" for Transfer. The system will respond with "Transfer?" or "Transfer what file?". Type in "APPLE1:SYSTEM.WRK.TEXT". The next computer question is "To where?". You should respond with "STOCK1:(PROGRAM NAME).TEXT". For example, if we made the changes discussed earlier in this section in program Results, our response would be: "STOCK1:RESULTS.TEXT".

When this step is successfully completed, remove diskette STOCK1: from disk drive 2 and insert diskette STOCK:. With the system still in Filer mode, press "T" for transfer again. Transfer "APPLE1.SYSTEM.WRK.CODE" to "STOCK:(PROGRAM NAME).CODE", in the same manner. When this is done, press "Q" for Quit to return you to command level. The Apple Pascal diskette APPLE1: can now be removed from disk drive 1 and stocking rationale diskette STOCK: can be transferred from disk drive 2 to disk drive 1. Disk drive 2 can be empty now.

Reboot the system using [CNTL] @, or [RESET] (Open Apple-Control-Reset for Apple IIe), or turn the power off and on. Your updated stocking rationale programs should now be working.

Any changes made to the programs follow the procedure described above. Briefly listed, the steps to be taken are:

- 1) Locate the areas to make the changes (Append. 2).
- 2) Determine the specific changes to make (using Append. 1).
- 3) Cross check to find the appropriate program diskette file to edit (Append. 3).

- 4) Load the program text file into the Pascal Editor workspace.
- 5) Make the program changes.
- 6) Compile the updated program.
- 7) Save the new version of the program to the appropriate diskette.

Different types of changes involve different degrees of complexity in program modification, for example, changing the stock/catch ratio for rainbow trout. To do this requires modifications in two programs. The stock/catch or catch/stock ratio is accessed in program Rationale, procedure GetInput, and in program Results, procedure CalculateStocking (Append. 3).

Changing one program at a time, follow the steps listed above. In Appendix 1, find procedure GetInput in program Rationale. About halfway through the procedure is the equation:

$$\text{StockObj[Rainbow]} := (\text{CatchObj[Rainbow]}/100) * \text{Objective} * 27.40;$$

where

StockObj = the number of fish required to meet the catch objective for that species.

CatchObj = the species preference value, in whole percents, given for that species. The value is divided by 100 to get the fractional part of one (1).

Objective = the total Catch Objective.

27.40 = the stock/catch ratio for rainbow trout.

Follow the procedures described previously to load the program into the Editor, make the change in the equation listed above (altering the stock/catch ratio of 27.40), compile the program, and store the compiled and text version on the appropriate diskettes.

Using the same strategy, locate 'CatchRatio[Rainbow]' in procedure CalculateStocking in program Results. Load the program, make the changes, compile and save the updated versions of the program, in the same manner as above.

As a final example involving very complex program changes is the addition of another species to the stocking rationale program. A change such as this would require changes in six of the eight programs, and an updating of two of the data files. In each of the programs, just below the title, is a section called TYPE (Append. 1). The TYPE section lists user specified arrays and sets. In the stocking rationale program, there are sets for fisheries, management zones, and fish species. To add a species to the stocking

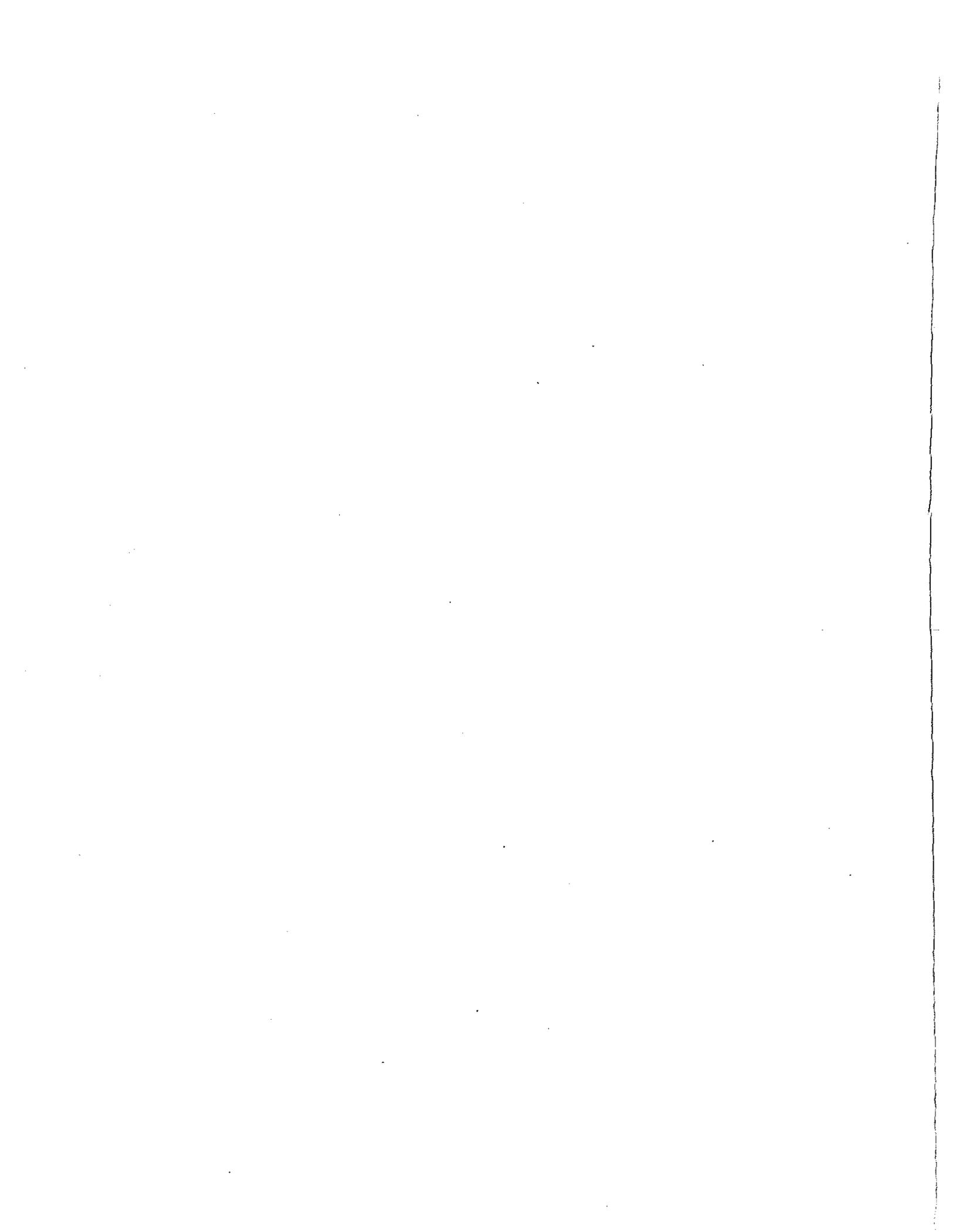
rationale, insert the species name into the list (i.e., between rainbow and chinook) not at the end of the list. This must be done to each species set in all programs in the same manner. In other words, if splake were to be added to the species list between rainbow and chinook, it must be added to all the species lists between rainbow and chinook. The computer makes no allowances for spelling errors either, so it is best to double check each insert. Once the species name is inserted into the set type, the program automatically allows room for it in any array that uses the species list as a parameter (e.g., SppObj = ARRAY [Brook..Lake]).

In addition to changing the species set type, each program must be checked over carefully to find where the species names are used, and allowances made to include the new species. For example, in program PastCatchFile, in procedure InitializeFile, on the tenth line of the program (Append. 1), there is a FOR statement which involves the fish species list. The FOR statement iterates fish species from "Brook" to "Coho". We placed "Splake" on the list between "Rainbow" and "Chinook", so the FOR statement includes "Splake" between "Rainbow" and "Chinook" automatically. In the CASE statement (two lines below the FOR statement) however, each species is listed with its corresponding species label. In the CASE statement, no provision is made for "Splake", so it must be inserted. Placing it between "Rainbow" and "Chinook" would be the obvious choice of location because of its location in the species list.

These types of details must be looked for in all of the programs and changes must be made to include the new species if necessary. Other places where changes would be required include stock/catch and catch/stock ratios, fingerling and yearling ratios, CASE statements that use fish species, species stocking costs, and the species label name on any output table that lists the fish species.

Each program must be read carefully to find where changes must be made. Then each program must be edited to make the changes by going through the steps listed previously. Once all the modifications are made, the past catch file and the historic stocking file must be reinitialized to include stocking and catch data for the new species. With these changes made, the programs can now be run with the new species included.

Extreme care and caution must be exercised when making any and all program modifications. Making a change in one program could produce unforeseen results in another part of the same program or in a different program. Always keep a working backup copy of the diskettes. Update the backup copies only when you are sure the modified programs are running properly.



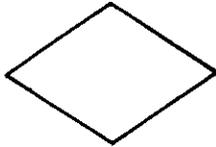
APPENDIX 5.

Flow diagram of main computational program (Rationale).

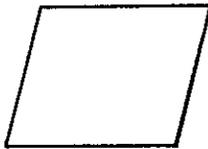
LEGEND



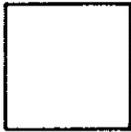
Process



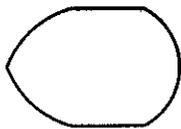
Decision



Input



Storage-Diskette



Monitor Screen



Hardcopy Printout

