Trini Style Band Management PROGRAM

For professional use only.

This is a new program to help manage a Carnival band. Should be used by end users.



2013

PCC Computing

1/7/2013

**TABLE OF CONTENTS**

**Page Number**

* Identification of Problem 3
* Reasons for choosing this programming language 4
* Hardware Requirements 4
* Software Requirements 5
* Screenshots and Instructions 6
* Algorithm 10
* Trace Table (Testing Program) 14
* Pascal Program Listing 16

PROBLEM

Carnival has long been a traditional cultural festival in several Caribbean territories. Every year, thousand of costumed masqueraders parade through the streets, dancing to the pulsating sounds of steelband, calypso and soca. Most of these revellers register to participate in the band of their choice months before the activities begin. Organizers of these bands must utilize management and organizations skills in order to stage a successful event.

This program helps the band leaders to organize and manage their band without having to manually write all the information .The program can utilized to help in members payment and where members should be i.e. different sections All information required from a member can be input and saved for later reference in order to aid the band leader in the organization part of his job.

The leader would need a program to help them to input member’s name, payment, section, and other personal information and to help him display the information for viewing and updating.

Reasons for Programming Language Chosen

* The Pascal programming language was the first programming language I was exposed to in high school and the one I am most proficient at.
* It makes use of English-Like statements as well as contain its own built in data types such as Integer, Real, Character and Boolean. This characteristic of the program makes it easy to debug, develop and write.
* The Pascal language provides a defined set of data structures which include Arrays, Records, Files and Sets.
* It supports Structured programming through the use of subprograms called procedures and functions. Structured Programming is a style of computer programming in which a program consists of simple subroutines.
* Because the language is simple and expressive in nature, it allows for effective teaching of computer programming techniques.
* The program also has useful debugging utilities.
* It is not machine dependant, which means that it can run on a variety of different types of computers, giving you more flexibility in data transfer and storage.
* Technical assistance for the program can be easily obtained.

Hardware Requirements

* Intel Pentium 2 Processor with speed of 2.1GHz. – A processer of high speed is needed to process data as quickly as possible.
* 1 GB RAM –The computer system will require Random Access Memory to hold whatever data that will currently be used by the program. The greater the amount of RAM, the easier it is for the processor to access information more quickly.
* 80 GB Hard disk– The hard disk is used to store all information and data from the program when they are not being processed so they can be accessed when next the computer is turned back on.
* Monitor – A monitor would be used to observe the various input and output screens of the program in use.
* CD-ROM Drive – This will be used to keep copies of storage files in a remote location for further use and backup in case of an emergency.
* Mouse: The pointing device used for moving the cursor on the monitor screen to access windows and execute programs. With this device, one can access the program, execute files and make adjustments to suite the user’s needs.
* Keyboard: Another input device like the mouse, used for entering data into the computer. This can be used for creating the program and can work together with the mouse for applications and executions within the program.
* Printer – A printer will be required to print any documentation that will be needed for hard copies.
* UPS System-This is used to ensure that data will be saved in the event of a power failure.
* USB ports – A USB ports is used to access data from external information storage devices such as flash drives.
* Modem- A modem is used for connecting to the internet to download updates on software and other uses for the business.

Software Requirements –

* Pascal Compiler – The Pascal compiler is needed to write and run programs. In this case Dev-Pascal ver 1.9.2 was used
* Word Processor –This will be needed to type and make adjustments to documentation for the system.
* Operating System – Needed to manage the computer resources and for any other program to be executed. The Operating System used is Windows 7 Ultimate (The program is backward compatible).
* Antivirus Software – Antivirus software is needed to protect the computer from viruses and other threats. The Antivirus software used was Avast Internet Security.
* Disk utility – Disk utility is needed to intercept files, decompress them, and then pass them to the operating system.
* Games- Games can be used to keep staff members busy when there are currently no customers in the building though it is not encouraged in order to maximize productivity.

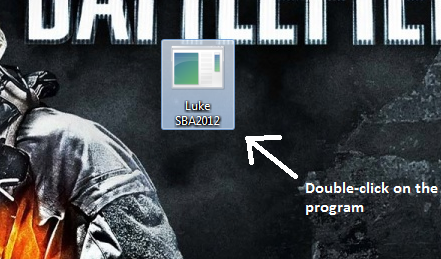
Screenshots and Instructions

ATTENTION PCC STUDENTS:  
Please note not all screenshots have been inserted….

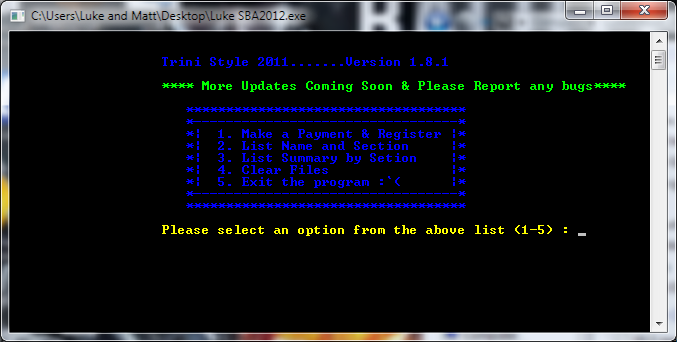
(Only a few have been included to give you an idea of what is required)

Starting the program :

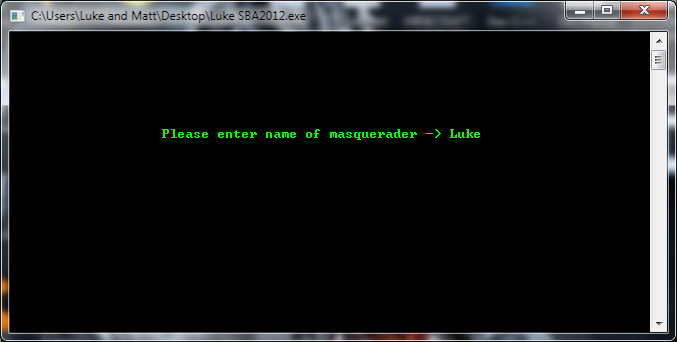
1. The Program should be placed on Desktop.
2. Double-click on the program as shown below.

T

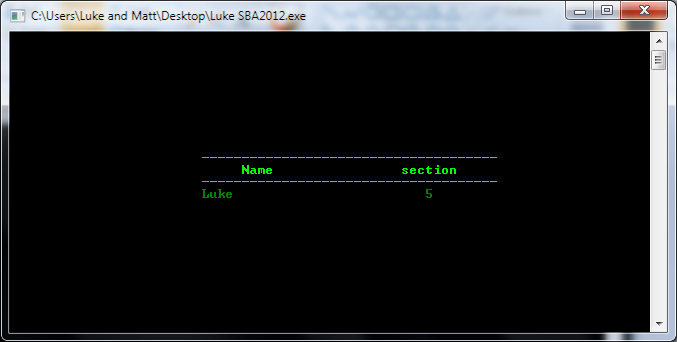
1. The Program should open as shown below :



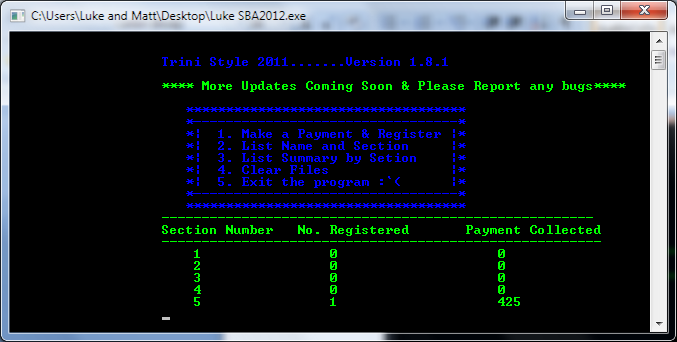
1. Select an Option From the list.
2. Option 1 Allows the User to Register a Masquerader and His/her Payment, then the program Assigns them to a Section by the payment received and the data is stored.

Example of Selection 1:

1. Option 2 Displays the Masqueraders Registered By Name and Section.

Example of Selection 2:

1. Option 3 Displays The Total Number of Persons in Each Section as well as it calculates the total payments received per section.

Example of Selection 3 : 

1. Option 4 Clears All Data In the Record and allows a user to start over from scratch if needed.
2. Option 5 Exits the program.

Algorithm

The Following Algorithm was used to make the program.

(\*---------------------------------------\*)

(\* This module is responsible for \*)

(\* initializing some values in the \*)

(\* program (ie setting values to zero \*)

(\*--------------------------------------\*)

module initialise;

start

sec\_1=0;sec\_2=0;sec\_3=0;sec\_4=0;sec\_5=0;

s1\_pay=0;s2\_pay=0;s3\_pay=0;s4\_pay=0;s5\_pay:=0;

end;

(\*---------------------------------------\*)

(\* This module is responsible for \*)

(\* finding the number of persons and total \*)

(\* payment in each various section. \*)

(\*---------------------------------------\*)

module section determined;

start

with one\_rec do

case payment of

160: start increase(sec\_1);section1pay:=section1\_payment + 160;s\_id:= 1; end

220: start increase(sec\_2);section2pay:=section2\_payment + 220;s\_id:= 2; end

280: start increase(sec\_3);section3pay:=section3\_payment + 280;s\_id:= 3; end

350: start increase(sec\_4);section4pay:=section4\_payment + 350;s\_id:= 4; end

425: start increase(sec\_5);section5pay:=section5\_payment + 425;s\_id:= 5; end

stop;

end;

(\*---------------------------------------\*)

(\* This module is responsible for \*)

(\* prompting a name and payment for each \*)

(\* masquerader and adding it to a file \*)

(\* note record files as used to ensure \*)

(\* added data is accessible even when the\*)

(\* program is shut down or restarted. \*)

(\*---------------------------------------\*)

module make\_a\_payment;

start

with one\_rec do

start

write('Please enter name of masquerader -> ');readln(name);

repeat

write('Please enter name of payment -> ');readln(payment);

until (payment = 160) or (payment = 220) or (payment = 280) or

(payment = 350) or (payment = 425);

determine\_section;

end;

assign(mas,'C:\SBA\mas.pas');

rewrite(mas);size:=filesize(mas);

seek(mas,size);

write(mas,one\_rec);

close(mas);

end;

(\*-------------------------------------------\*)

(\* This module is responsible for listing \*)

(\* the name and section of each masquerader \*)

(\*-------------------------------------------\*)

module name\_and\_section;

var i,size:integer;

start

assign(mas,'c:\SBA\mas.pas');

reset(mas);size:=filesize(mas);

gotoxy(20,9);textcolor(blue);

write('-------------------------------------');

gotoxy(20,10);

write(' Name section');

gotoxy(20,11);

write('-------------------------------------');

textcolor(white);

for i:= 1 to size do

start

read(mas,one\_rec);

gotoxy(20,13+ i);

write(one\_rec.name);

gotoxy(53,13+i);

write(one\_rec.s\_id);

end;

close(mas);

readln;

end;

(\*-------------------------------------------\*)

(\* This module is responsible for listing \*)

(\* for the different sections the number of \*)

(\* masqueraders as well as the total payment \*)

(\* made. \*)

(\*-------------------------------------------\*)

module section\_summary;

var size,i:integer;

start

assign(mas,'c:\SBA\mas.pas');

reset(mas);size:=filesize(mas);

initialise;

for i:= 1 to size do

start

read(mas,one\_rec);

determine\_section;

end;

gotoxy(20,11);textcolor(blue);

write('------------------------------------------------------');

gotoxy(20,12);

write('Section number Number Registered amount Payment Collected');

gotoxy(20,13);

write('-------------------------------------------------------');

textcolor(blue);gotoxy(20,14);

write(' 1 ',sec1,' ',s1pay);

write(' 2 ',sec2,' ',s2pay);

write(' 3 ',sec3,' ',s3pay);

write(' 4 ',sec4,' ',s4pay);

write(' 5 ',sec\_5,' ',s5pay);

readln;

stop;

module do\_menu;

var leave:boolean;option:integer;

start

leave:=false;

repeat

gotoxy(20,3);

writeln('Steel Raiders..... version 2.20);

gotoxy(20,5);

writeln(' 1. Make a Payment ');gotoxy(20,6);

writeln(' 2. List Name and Section ');gotoxy(20,7);

writeln(' 3. List Summary by Setion ');gotoxy(20,8);

writeln(' 4. Exit ');gotoxy(20,9);

gotoxy(20,10); textcolor(red);

write('Please select an option (1-4) : ');readln(option);

case option of

1: make\_payment;

2: name\_and\_section;

3: section\_summary;

4:leave:= true;

end;

until (leave = true);

stop

start {......................main program ...................}

initialise;

do\_menu;

Stop {........................main program ...................}

Trace Table for The Program

(Testing)

Assigning Sections

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Name Entered | Payment Entered | Section Assigned |
| 1 | Mark | 280 | 3 |
| 1 | Lisa | 425 | 5 |
| 1 | Montrose | 160 | 1 |
| 1 | Henry | 220 | 2 |
| 1 | Mandy | 350 | 4 |
| 1 | Jimmy | 220 | 2 |
| 1 | Marvin | 425 | 5 |

Outputting Stored Data by Name and Section.

|  |  |  |
| --- | --- | --- |
| Option | Name Display | Section Display |
| 2 | Mark | 3 |
|  | Lisa | 5 |
|  | Montrose | 1 |
|  | Henry | 2 |
|  | Mandy | 4 |
|  | Jimmy | 2 |
|  | Marvin | 5 |

Outputting Section, Number Of Person in Section & Payment Collected Per Section.

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Section Number | Number of Persons in Section | Total Collected |
| 3 | 1 | 1 | 160 |
|  | 2 | 2 | 440 |
|  | 3 | 1 | 280 |
|  | 4 | 1 | 350 |
|  | 5 | 2 | 850 |

Trace Table for When Option 4 is selected and then Option2

|  |  |  |
| --- | --- | --- |
| Option | Name display | Section Display |
| 2 |  |  |

Trace Table When Option 4 is selected and then Option 3.

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Section No. | Number of persons in Section | Total collected |
| 3 |  |  |  |

When Option 5 Is Selected The program Exits.

Pascal Listing for the Program

The Pascal Code used for the making of this program is as follows:

(\*---------------------------------\*)

(\* Name : PCC SAMPLE \*)

(\* Exam Yr : \*)

(\* School : PCC \*)

(\*--------------------------------\*)

program Trini\_Style\_2012;

uses crt;

type mas\_record = record {Declaration of mas\_record as a record type}

id:integer;

name:string;

s\_id:integer;

payment:integer;

end;

mas\_file = file of mas\_record;

var masquerader:mas\_file;

section\_1,section\_2,section\_3,section\_4,section\_5:integer;

sec1\_pay,sec2\_pay,sec3\_pay,sec4\_pay,sec5\_pay:integer;

one\_rec:mas\_record;

(\*-------------------------------------------------\*)

(\* This procedure is responsible for \*)

(\* setting to zero some of the variables \*)

(\* used in the program. \*)

(\*------------------------------------------------\*)

procedure initialise;

begin

section\_1:=0;section\_2:=0;section\_3:=0;section\_4:=0;section\_5:=0;

sec1\_pay:=0;sec2\_pay:=0;sec3\_pay:=0;sec4\_pay:=0;sec5\_pay:=0;

end;

procedure clear\_the\_file;

begin

assign(masquerader,'J:\IT Sba Stuff\Programming\masquerader.pas');

rewrite(masquerader);

close(masquerader);

gotoxy(20,12);write('Please Wait...Contents of file being erased....');

delay(1000);

end;

(\*-----------------------------------------------\*)

(\* This procedure is responsible for \*)

(\* finding the no. of persons and total \*)

(\* payment in each section. \*)

(\*-----------------------------------------------\*)

procedure determine\_section;

begin

with one\_rec do

case payment of

160: begin inc(section\_1);sec1\_pay:=sec1\_pay + 160;s\_id:= 1; end;

220: begin inc(section\_2);sec2\_pay:=sec2\_pay + 220;s\_id:= 2; end;

280: begin inc(section\_3);sec3\_pay:=sec3\_pay + 280;s\_id:= 3; end;

350: begin inc(section\_4);sec4\_pay:=sec4\_pay + 350;s\_id:= 4; end;

425: begin inc(section\_5);sec5\_pay:=sec5\_pay + 425;s\_id:= 5; end;

end;

end;

(\*---------------------------------------\*)

(\* This procedure is responsible for \*)

(\* prompting a name and payment for each \*)

(\* masquerader and adding it to a file \*)

(\* N.B record files as used to ensure \*)

(\* added data is accessible even when the\*)

(\* program is shut down and restarted. \*)

(\*---------------------------------------\*)

procedure make\_payment\_and\_register;

var size:integer;

begin

with one\_rec do

begin

clrscr;gotoxy(20,9); textcolor(lightgreen);

write('Please enter name of masquerader -> ');readln(name);

repeat

gotoxy(20,11);

writeln('Payment Options -$160,$220,$280,$350,$425');

gotoxy(20,13);

write('Please enter payment ($) -> ');

readln(payment);

until (payment = 160) or (payment = 220) or (payment = 280) or

(payment = 350) or (payment = 425);

determine\_section;

end;

assign(masquerader,'J:\IT Sba Stuff\Programming\masquerader.pas');

reset(masquerader);size:=filesize(masquerader);

seek(masquerader,size);

write(masquerader,one\_rec);

close(masquerader);

end;

(\*-------------------------------------------\*)

(\* This procedure is responsible for listing \*)

(\* the name and section of each masquerader \*)

(\*-------------------------------------------\*)

procedure name\_and\_section;

var i,size:integer;

begin

assign(masquerader,'J:\IT Sba Stuff\Programming\masquerader.pas');

reset(masquerader);size:=filesize(masquerader);

clrscr;

gotoxy(25,11);textcolor(lightgreen);

write('-------------------------------------');

gotoxy(25,12);

write(' Name section');

gotoxy(25,13);

write('-------------------------------------');

textcolor(green);

for i:= 1 to size do

begin

read(masquerader,one\_rec);

gotoxy(25,13+ i);

write(one\_rec.name);

gotoxy(53,13+i);

write(one\_rec.s\_id);

end;

close(masquerader);

readln;

end;

(\*----------------------------------------------------------\*)

(\* This procedure is responsible for listing \*)

(\* for the different sections the number of \*)

(\* masqueraders as well as the total payment \*)

(\* made. \*)

(\*----------------------------------------------------------\*)

procedure section\_summary;

var size,i:integer;

begin

assign(masquerader,'J:\IT Sba Stuff\Programming\masquerader.pas');

reset(masquerader);size:=filesize(masquerader);

initialise;

for i:= 1 to size do

begin

read(masquerader,one\_rec);

determine\_section;

end;

gotoxy(20,16);textcolor(lightgreen);

write('------------------------------------------------------');

gotoxy(20,17);

write('Section Number No. Registered Payment Collected');

gotoxy(20,18);

write('-------------------------------------------------------');

textcolor(lightgreen);gotoxy(20,19); textcolor(lightgreen);

write(' 1 ',section\_1,' ',sec1\_pay);gotoxy(20,20);

write(' 2 ',section\_2,' ',sec2\_pay);gotoxy(20,21);

write(' 3 ',section\_3,' ',sec3\_pay);gotoxy(20,22);

write(' 4 ',section\_4,' ',sec4\_pay);gotoxy(20,23);

write(' 5 ',section\_5,' ',sec5\_pay);gotoxy(20,24);

readln;

end;

procedure do\_menu;

var leave:boolean;option:integer;

begin

textcolor(lightblue);

leave:=false;

repeat clrscr;

gotoxy(20,3); textcolor(lightblue);

writeln('Trini Style 2011.......Version 1.8.1');

gotoxy(20,5); textcolor(lightgreen);

writeln ('\*\*\*\* More Updates Coming Soon & Please Report any bugs\*\*\*\*'); gotoxy(20,7); textcolor(lightblue);

writeln(' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');gotoxy(20,8);

writeln(' \*---------------------------------\*');gotoxy(20,9);

writeln(' \*| 1. Make a Payment & Register |\*');gotoxy(20,10);

writeln(' \*| 2. List Name and Section |\*');gotoxy(20,11);

writeln(' \*| 3. List Summary by Setion |\*');gotoxy(20,12);

writeln(' \*| 4. Clear Files |\*');gotoxy(20,13);

writeln(' \*| 5. Exit the program :`( |\*');gotoxy(20,14);

writeln(' \*---------------------------------\*');gotoxy(20,15);

writeln(' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

gotoxy(20,17); textcolor(yellow);

write('Please select an option from the above list (1-5) : ');

readln(option);

case option of

1: make\_payment\_and\_register;

2: name\_and\_section;

3: section\_summary;

4: clear\_the\_file;

5: leave:= true;

end;

until (leave = true);

end;

begin {......................main program ...................}

clrscr;

initialise;

do\_menu;

end. {........................main program ...................}