**Soft Ireland**

**Procedural Programming Assignment**

**KTQ1852**

**Philip Woulfe**

Contents

[Contents 1](#_Toc452471834)

[Task 1 – Report on the main advantages and disadvantages of the procedural programming paradigm 3](#_Toc452471835)

[What is the Procedural Programming paradigm? 3](#_Toc452471836)

[Features shared with other programming paradigms 3](#_Toc452471837)

[Features of C 3](#_Toc452471838)

[Differences from OOP 4](#_Toc452471839)

[Modularity 4](#_Toc452471840)

[What are its main advantages? 4](#_Toc452471841)

[What are its main disadvantages? 4](#_Toc452471842)

[Task 2 – Report on the design specification of the payroll application 5](#_Toc452471843)

[Design specification 5](#_Toc452471844)

[0001 – Employees – Add and remove 5](#_Toc452471845)

[0002 – Employees – Department 5](#_Toc452471846)

[0003 – Employees – Rate of pay 5](#_Toc452471847)

[0004 – Employees – Wages 6](#_Toc452471848)

[0005 – Employees – I/O 6](#_Toc452471849)

[Data Flow Diagrams 8](#_Toc452471850)

[Custom Data Types 9](#_Toc452471851)

[Procedures 10](#_Toc452471852)

[Main Procedures 10](#_Toc452471853)

[UI procedures 10](#_Toc452471854)

[Sample Employee File 12](#_Toc452471855)

[Sample Payroll File 12](#_Toc452471856)

[UI 13](#_Toc452471857)

[Main Menu 13](#_Toc452471858)

[View Employees Menu 14](#_Toc452471859)

[Add Employee Menu 15](#_Toc452471860)

[Change Employee Employment Status Menu 19](#_Toc452471861)

[Change Employee Department Menu 22](#_Toc452471862)

[Change Employee Pay Rate Menu 25](#_Toc452471863)

[Calculate Payroll Menu 28](#_Toc452471864)

[Task 3 – Develop application 31](#_Toc452471865)

[Task 4 – Report on testing 32](#_Toc452471866)

[Documentation 32](#_Toc452471867)

[Test of the application 33](#_Toc452471868)

[Analysis of actual test results against expected results to identify discrepancies 33](#_Toc452471869)

[0001 – Employees – Add and remove 33](#_Toc452471870)

[0002 – Employees – Department 34](#_Toc452471871)

[0003 – Employees – Rate of pay 34](#_Toc452471872)

[0004 – Employees – Wages 35](#_Toc452471873)

[0005 – Employees – I/O 36](#_Toc452471874)

[0006 – Employees – Main UI Navigation 37](#_Toc452471875)

[Bugs 38](#_Toc452471876)

[Bug 0001 38](#_Toc452471877)

[Bug 0002 40](#_Toc452471878)

[Bug 0003 42](#_Toc452471879)

[Recommendation for improvements 44](#_Toc452471880)

[Improve wage processing UI 44](#_Toc452471881)

[Create individual payslips for employees 44](#_Toc452471882)

# Task 1 – Report on the main advantages and disadvantages of the procedural programming paradigm

## What is the Procedural Programming paradigm?

Procedural programming (PP) is a programming paradigm derived from the structural programming paradigm. The core concept of procedural programming is to break down the program into procedures (comparable with Object Oriented (OOP) methods) to make the program more modular. Modularity is desirable as it allows a team of developers to collaborate more easily on a large project. The modularity allows different developers to work on separate self-contained procedures without disrupting the rest of the program.

### Features shared with other programming paradigms

#### Variables

Variables are used to store information. These can be primitive variables such as:

* natural numbers
* characters
* floating point numbers
* booleans

#### Structs

Structs are custom data-types, and example of this would be a *string*, under the surface this is just a char array.

#### Procedures

Procedures are a way of manipulating data. This can be procedures that take input and providing output or it can be structures like *if* statements or *loops*. These procedures are the key feature of procedural programing. Before procedures, goto statements were used to reuse other parts of code. This made code to read and understand what it was doing, as it could access any other part of the program. It could also have unexpected consequences as lines of code you didn’t want rerun could be executed.

### Features of C

One of the more difficult concepts in C is the notion of pointers. As C is strictly pass by value, i.e. a copy of the variable is passed to and from a procedure, in order to manipulate a specific variable and not a copy of that variable, one must use pointers.

In the following code, we declare an integer variable num and store the value of 20 in it. We then declare a pointer to an integer num\_pointer. We can then store the address of num in the pointer variable. This means that num and num\_pointer are now pointing to the same variable.

int num = 20; // Declare variable

int \*num\_pointer; // Declare pointer variable

num\_pointer = &num; // store address of num in pointer variable

Doing this allows us to pass these pointers into and out of procedures. This means the procedure and the procedure calling it can manipulate the same variables and not just copies of that variable.

### Differences from OOP

Where OOP bundles both procedures and data into objects, PP considers procedures and data as two separate concepts. Libraries are used to define data-types (structs) and procedures which can then be reused as needed. This is similar to OOP’s use of objects, except the data types only ever contain information, not programming logic.

### Modularity

Inputs – arguments

Procedures can accept arguments as inputs. The argument types have to be defined in a prototype and they must match the argument types in the actual procedure definition. A procedure can also accept an argument of type void i.e. no input.

It is important to note that all arguments in C are passed by value; pass by reference is simulated by passing the address of a variable (a pointer) and dereferencing that address within the function to read or write the actual variable.

Outputs – return values

Procedures can also return values from a procedure. The return type must be defined in the prototype and must match the return value in the actual procedure. A procedure can have a return type of type void i.e. no output.

It is important to note that all return values in C are passed by value; pass by reference is simulated by passing the address of a variable (a pointer) and dereferencing that address within the function to read or write the actual variable.

## What are its main advantages?

* Ubiquity – they are so prevalent that they are the language with the most resources available for learning.
* They are good for general purpose programming.
* They are more portable than low level languages which have to be written for specific CPUs

## What are its main disadvantages?

* They are less efficient than custom build low-level programs. This is only really an issue in a system where there are limited resources.
* Since there are so many different languages, programmers generally have to specialise in a particular language.

# Task 2 – Report on the design specification of the payroll application

## Design specification

We are building a payroll application for our client. It is to be constructed from the C procedural programming language. The specifics are details in the user stories listed below. The application will read the list of employees from a .csv (Comma Separated Values) file. The payroll reports will be saved in a .txt. Examples are provided below.

### 0001 – Employees – Add and remove

As a payroll operative, I would like to be able to add and remove employees from the application so I can keep the payroll files up to date with current employees.

#### Happy Path:

|  |  |
| --- | --- |
| Criteria 1 | Adding an employee allows users to calculate payroll for that employee. |
| Criteria 2 | Removing an employee prevents users from calculating payroll for that employee. |

#### Error Validation:

|  |  |
| --- | --- |
| Criteria 1 | Employee names should have a unique employee number |
| Criteria 2 | If an employee is removed, you should no longer be able to calculate payroll for that employee. |
| Criteria 3 | If an employee is removed, they should no longer appear on the payroll output file. |

#### Comment:

Removing an employee does not remove the employee from the system; it just sets a flag on that employee that causes it not to appear on the lists apart from the change employment status list. The employee will also not be included in the calculation of payroll.

### 0002 – Employees – Department

As a payroll operative, I would like to be able to assign each employee a department so I can know which department each employee works in.

#### Happy Path:

|  |  |
| --- | --- |
| Criteria 1 | Each employee must be assigned a department on creation. |
| Criteria 2 | The department should be updateable in case the employees move departments. |
| Criteria 3 | Department must be either HR, IT, Production or Sales & Marketing. |

#### Error Validation:

|  |  |
| --- | --- |
| Criteria 1 | Selecting a department outside the above should display an error. |

#### Comment:

There are 4 possible departments. Employees may be transferred between these departments. There are no plans to include any more departments at present.

### 0003 – Employees – Rate of pay

As a payroll operative, I would like to be able to assign each employee a rate of pay so I can calculate their wages.

#### Happy Path:

|  |  |
| --- | --- |
| Criteria 1 | There should be three bands of pay. |
| Criteria 2 | An employee can change the rate of pay between bands. |
| Criteria 3 | Department must be either HR, IT, Production or Sales & Marketing. |

#### Error Validation:

|  |  |
| --- | --- |
| Criteria 1 | Selecting an incorrect level of pay should display an error. |

#### Comment:

There are 3 possible pay bands. Employees may be transferred between pay bands. There are no plans to include any more pay rates at present.

### 0004 – Employees – Wages

As a payroll operative, I would like to be able to calculate wages for each employee so I can issue payments weekly.

#### Happy Path:

|  |  |
| --- | --- |
| Criteria 1 | I should be able to enter weekly hours worked for each current employee |
| Criteria 2 | Gross wages should be calculated as hours worked \* hourly rate |
| Criteria 3 | The smallest allowable unit of work should be no less than ¼ of an hour |

#### Error Validation:

|  |  |
| --- | --- |
| Criteria 1 | I should not be able to enter hours for non-employees |

#### Comment:

Hours and minutes must be input individually for each employee. Former employees should not appear when processing employee wages.

### 0005 – Employees – I/O

As a payroll operative, I would like to be able to issue a weekly statement of wages for the purposes of record keeping.

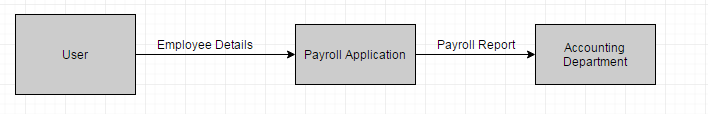
#### Happy Path:

|  |  |
| --- | --- |
| Criteria 1 | Application should read employee list from a file. |
| Criteria 2 | Application should save updated employee list to a file. |
| Criteria 3 | Once the wages are calculated, the result should be output to a file. |
| Criteria 4 | The name of the output file should include the date of the current week. |
| Criteria 5 | If more than one file is produced in a week, the newer file should overwrite the older file. |

#### Error Validation:

|  |  |
| --- | --- |
| Criteria 1 | Should generate error if can’t open employee list |
| Criteria 2 | Should create employee list if can’t find employee list |

#### Comment:



## Data Flow Diagrams

Figure 1 - Level 0 DFD

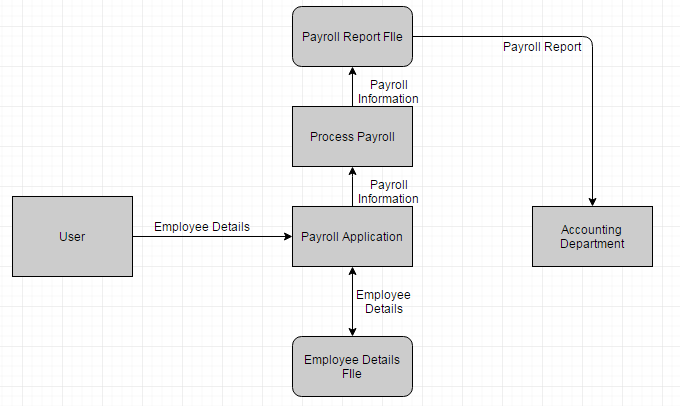


Figure 2 - Level 1 DFD

## Custom Data Types

typedef enum {

HR = 1,

IT = 2,

PRODUCTION = 3,

SALES\_AND\_MARKETING = 4

} Department\_t;

typedef enum {

ASSOCIATE = 1,

SENIOR = 2,

MANAGER = 3,

} Rate\_t;

typedef struct {

char name[50];

Department\_t dept;

Rate\_t rate;

bool currentEmployee;

int hoursWorked;

int minutesWorked;

float hourlyRate;

float weeksWages;

} Employee\_t;

## **Procedures**

### **Main Procedures**

/\* Processes wages for employees \*/

void processWages(Employee\_t empArr[], int arrayLength);

/\* Calculate pay for the week per employee \*/

Employee\_t calculateWeekly(Employee\_t employee);

Employee procedures

/\* Create a new employee\*/

int createNewEmployee(Employee\_t employees[], int arrayLength);

/\* Changes employee's department \*/

void changeDept(Employee\_t employees[], int arrayLength);

/\* Change employee's pay rate \*/

void changeRate(Employee\_t employees[], int arrayLength);

/\* Terminate or rehire employees \*/

void changeEmployeeStatus(Employee\_t employees[], int arrayLength);

### **UI procedures**

/\* Display Header \*/

void displayHeader(void);

/\* Clear command screen \*/

void clearScreen(void);

/\* Load text with character delay \*/

void printText(char \*text);

/\* Display main menu \*/

void displayMenu(void);

/\* Display Employees \*/

void displayEmployees(Employee\_t employees[], int arrayLength);

/\* Display new Employee \*/

void displayNewEmployee(Employee\_t employee, int arrayLength);

/\* Display Employee Status \*/

void displayStatusEmployee(Employee\_t employee[], int arrayLength);

/\* Display this weeks payroll \*/

void displayPayroll(Employee\_t employee[], int arrayLength);

IO Procedures

/\* Load Employees \*/

int loadEmployees(Employee\_t employees[], int arrayLength);

/\* Get fields from csv \*/

const char\* getfield(char\* line, int num);

/\* Save current employees to file \*/

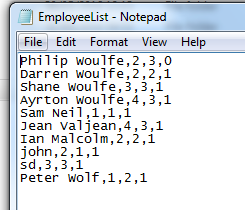
void saveEmployees(Employee\_t employees\_p[], int arrayLength);

/\* get week end date \*/

char \* getWEDate();

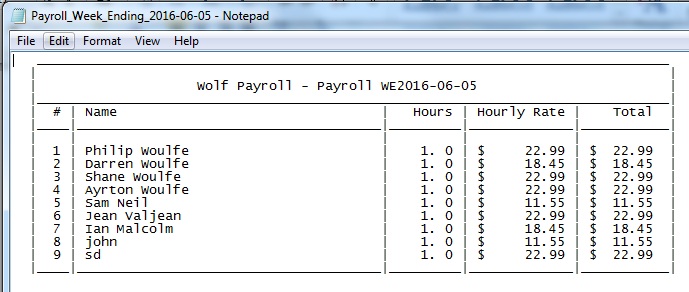
IO

### Sample Employee File



Note: File is a .csv

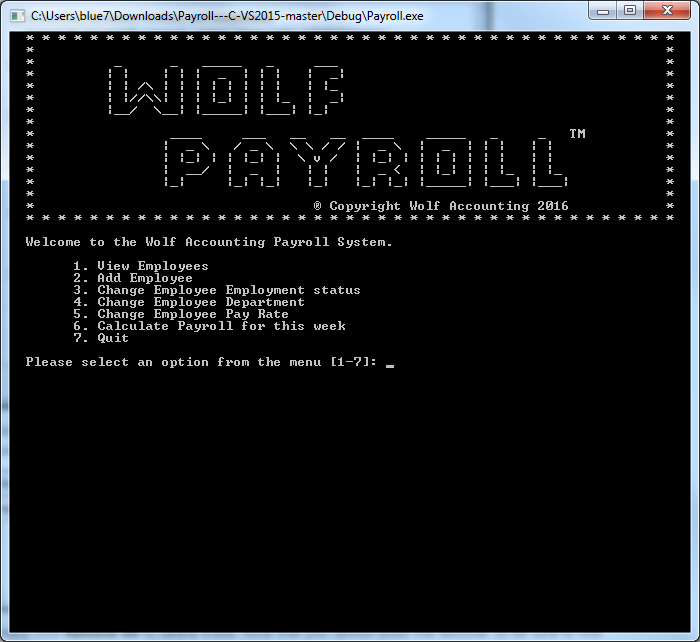
### Sample Payroll File



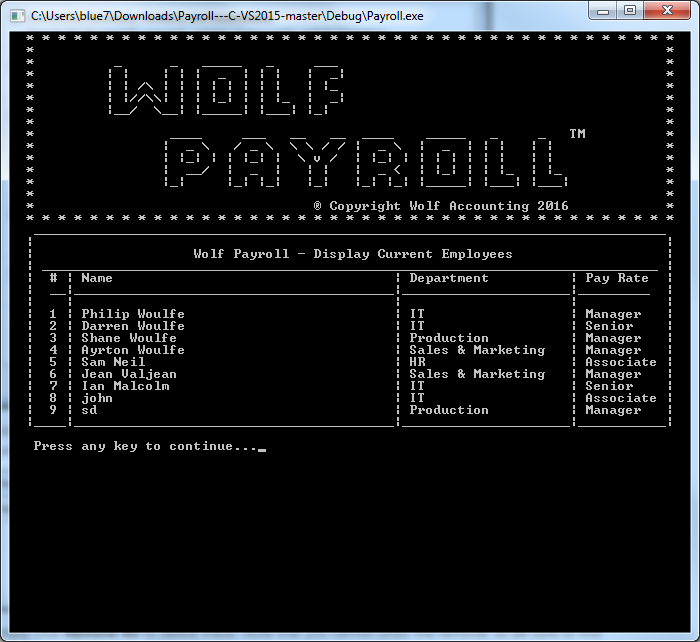
Note: File is a .txt

## UI

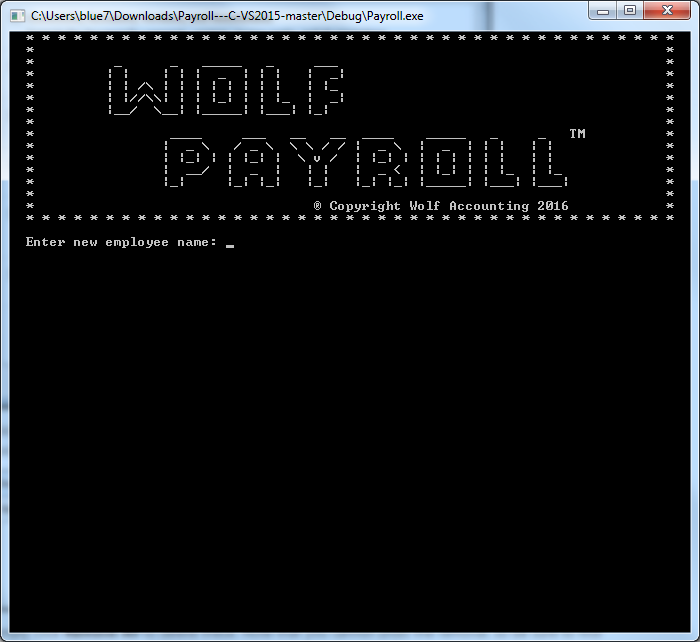
### Main Menu

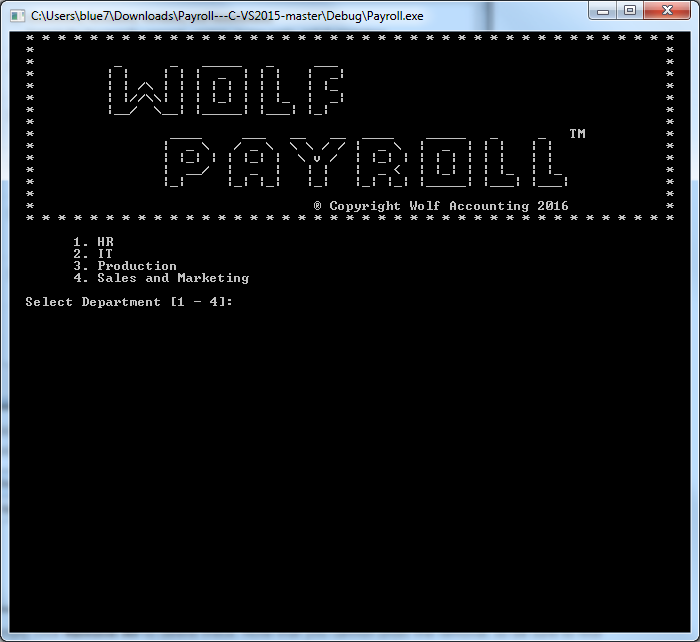


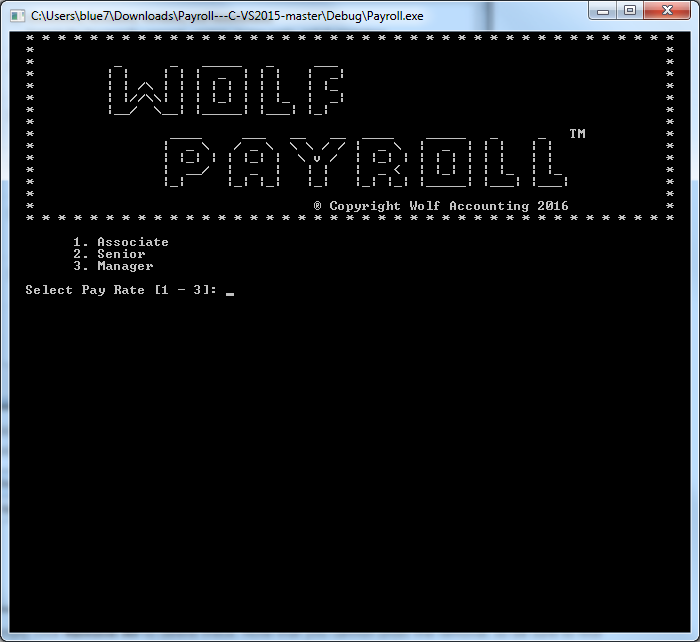
### View Employees Menu

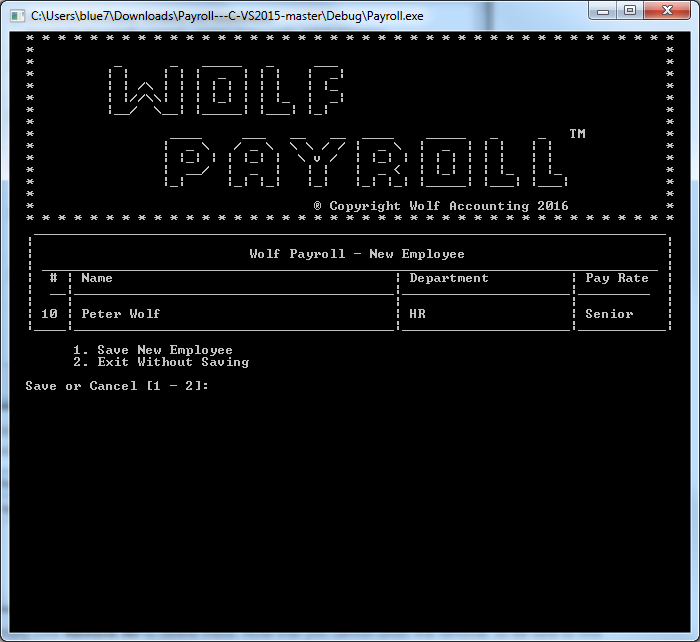


### Add Employee Menu

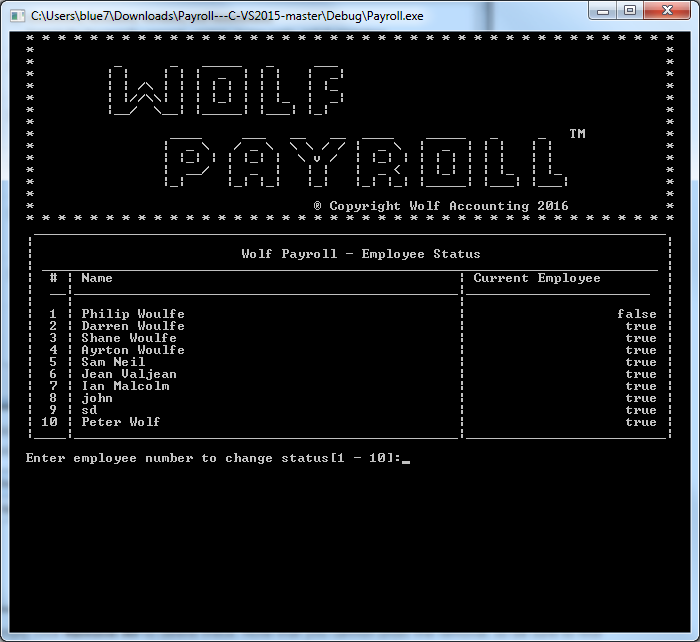


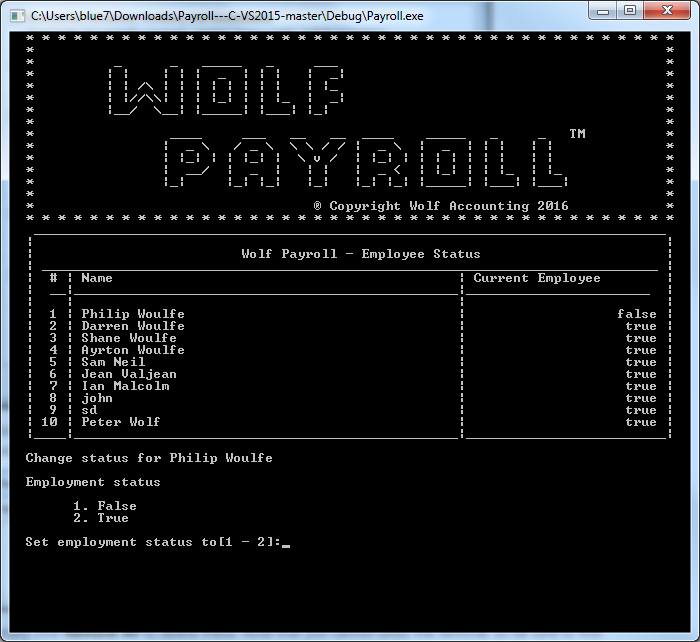


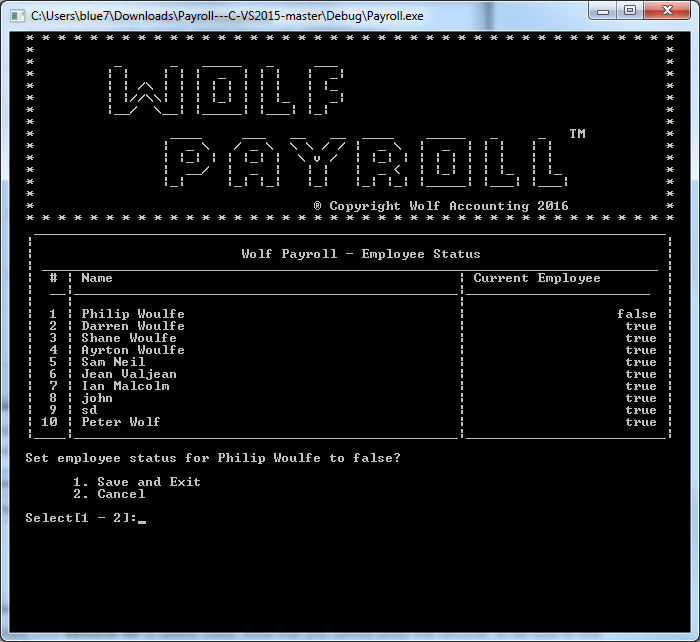




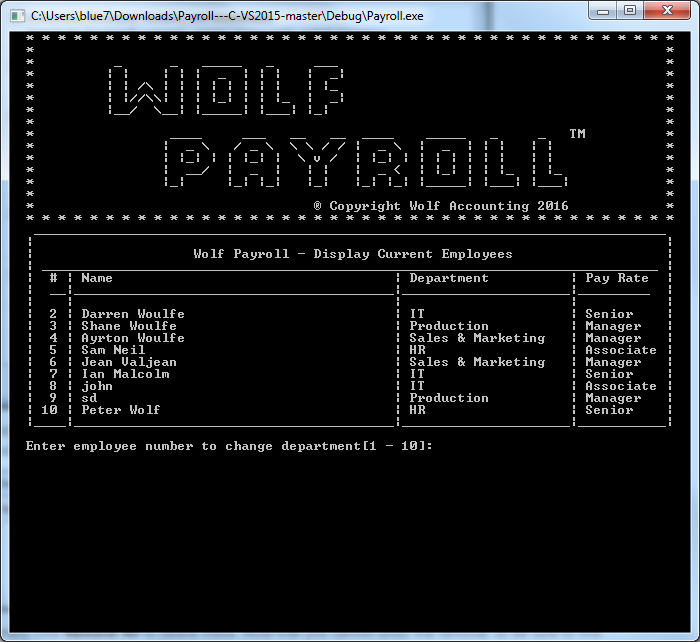
### Change Employee Employment Status Menu

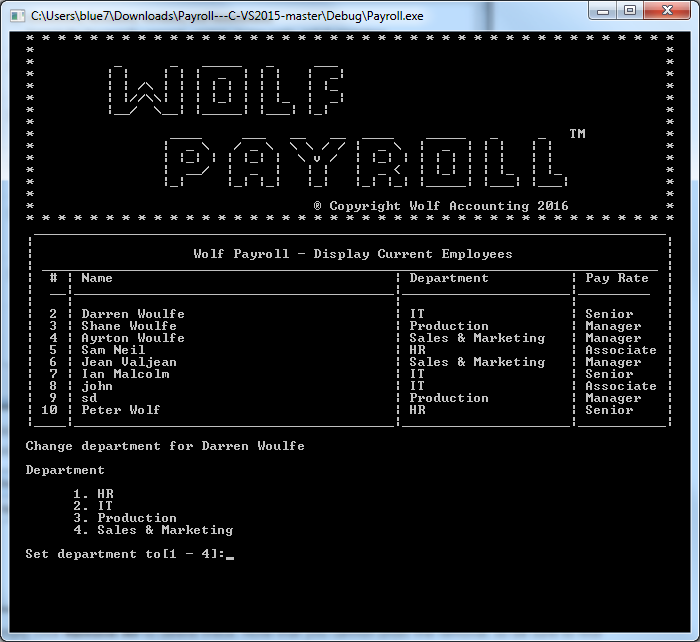


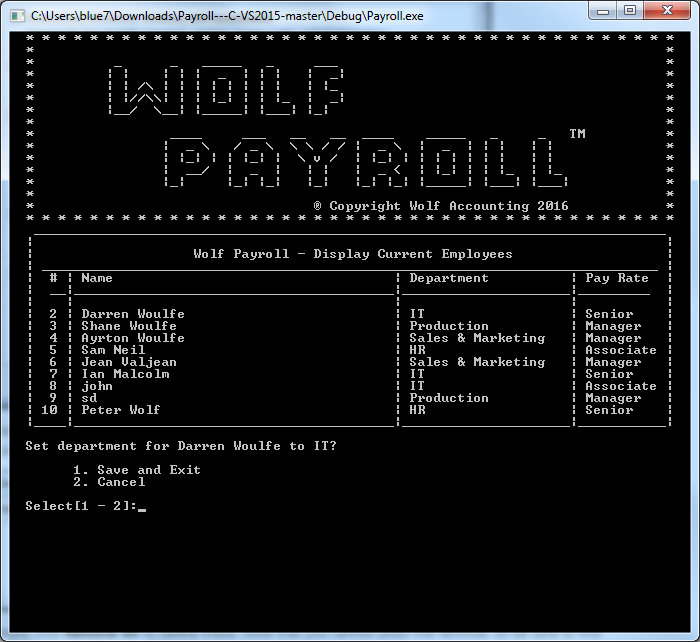




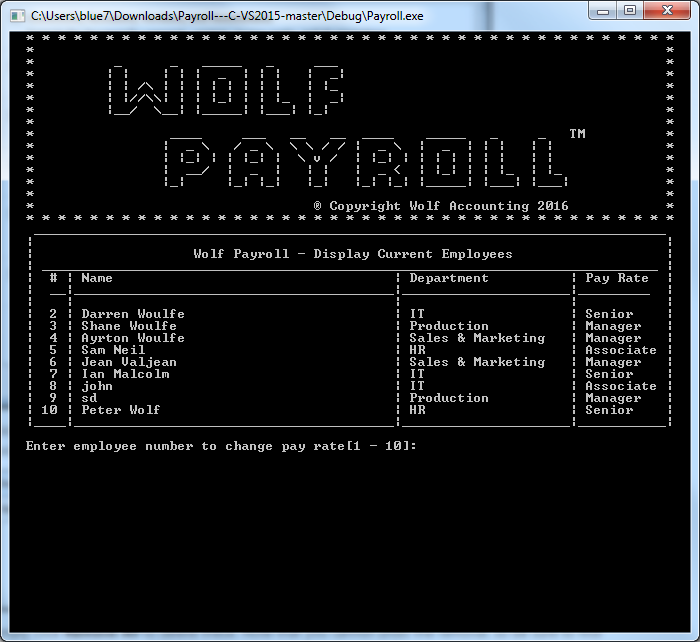
### Change Employee Department Menu

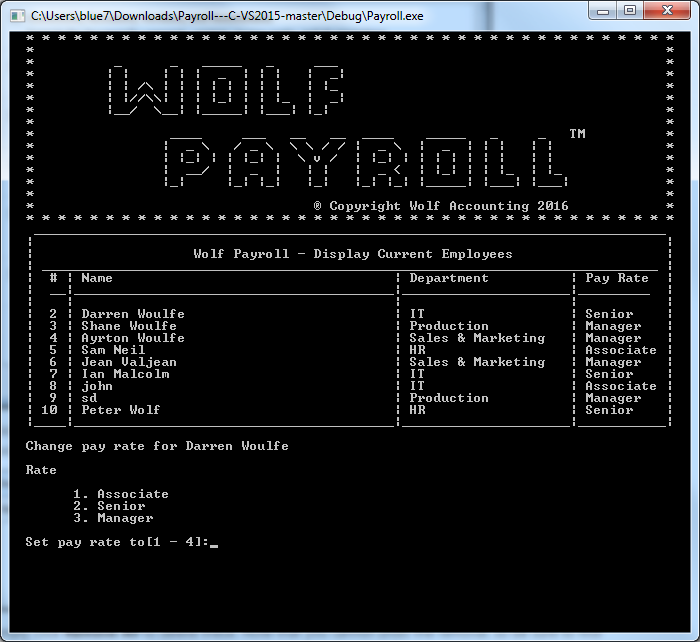


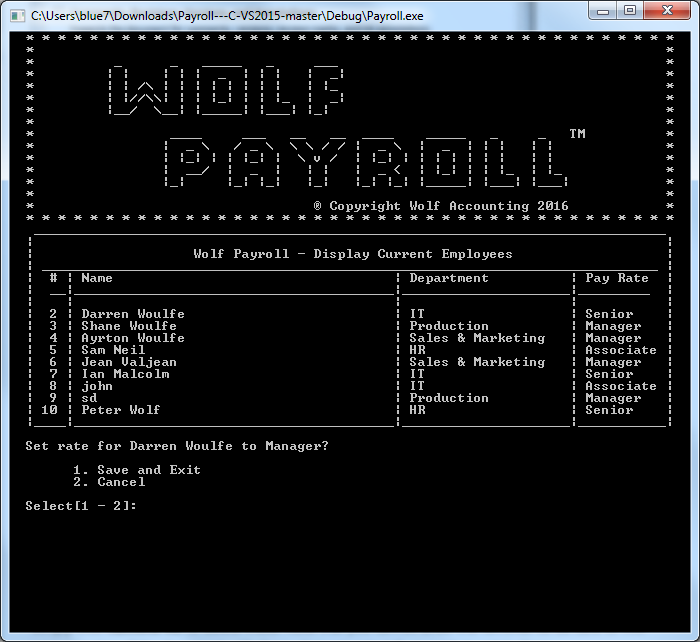




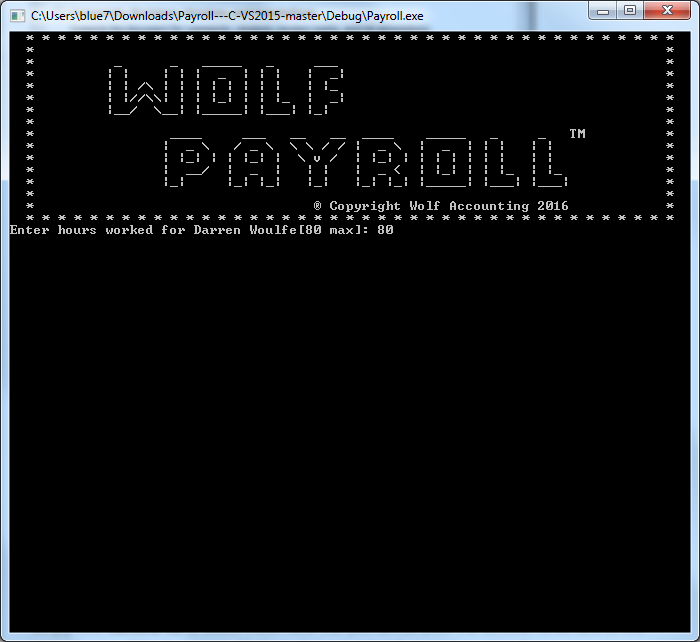
### Change Employee Pay Rate Menu

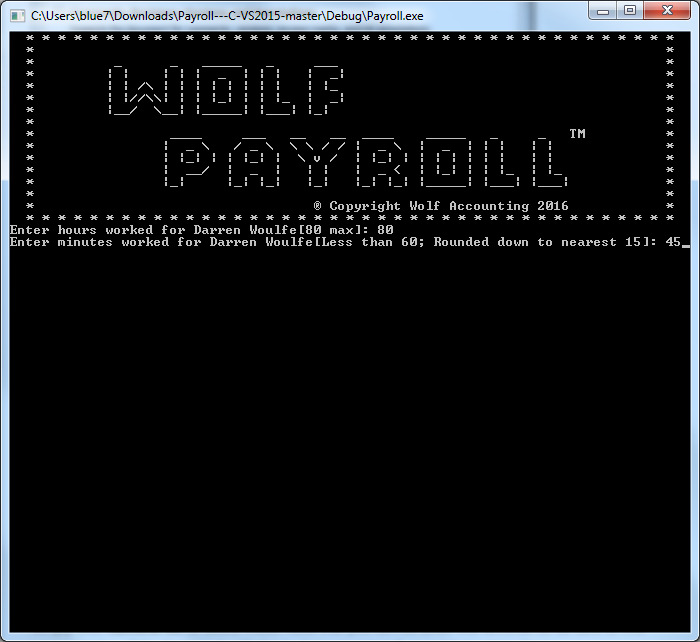


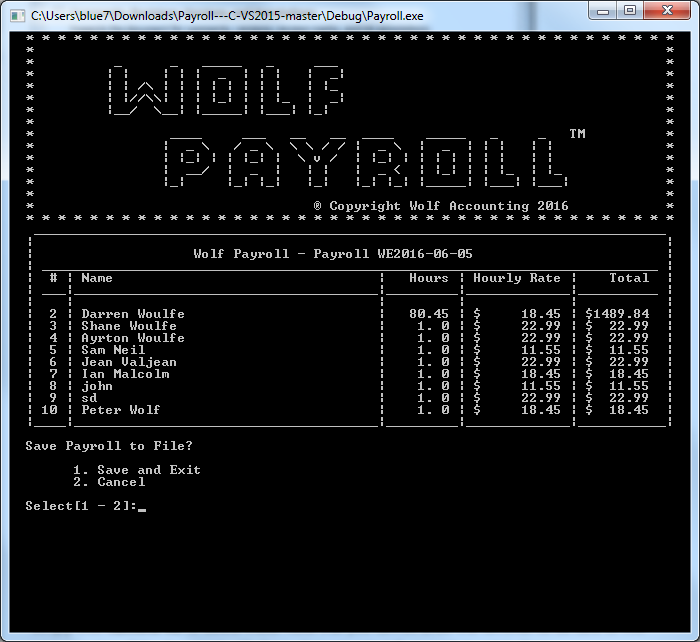




### Calculate Payroll Menu







# Task 3 – Develop application

# Task 4 – Report on testing

## Documentation

Documentation can be accessed at [./Documentation/html/index.html](Documentation/html/index.html)

Note: file should be opened in an internet browser, preferable Google Chrome. If your computer is not set up to open .html files with the default browser automatically, you will need to navigate to the file location (which is the location of this document plus the above) right click and open with… and select a browser.

## Test of the application

## Analysis of actual test results against expected results to identify discrepancies

### 0001 – Employees – Add and remove

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0001 – Employees – Add and remove | | | | | |
| Test Case ID | **Condition** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| 1.1 | Add an employee | Employee appears in the display employees menu | As Expected | Pass |  |
| 1.2 | Remove employee | Employee no longer appears in the display employees menu | As Expected | Pass |  |
| 1.3 | Calculate payroll with removed employee | Payroll not calculated for removed employee | As Expected | Pass |  |
| 1.4 | Calculate payroll for added employee | Payroll can be calculated for new employee | As Expected | Pass |  |
| 1.5 | Calculate payroll with removed employee | Employee should not appear in payroll file | As Expected | Pass |  |
| 1.6 | Save or cancel | Option to save new employee or exit without saving | As Expected | Pass |  |

### 0002 – Employees – Department

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0002 – Employees – Department | | | | | |
| **Test Case ID** | **Condition** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| 1.1 | Possible departments are HR, IT, Production or Sales & Marketing | Trying to assign department outside of above generates error | As Expected | Pass |  |
| 1.2 | Change employee department | Department can be changed for current employees | As Expected | Pass |  |
| 1.3 | Employees are assigned departments on creation | Employee cannot be created without department | As Expected | Pass |  |

### 0003 – Employees – Rate of pay

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0003 – Employees – Rate of pay | | | | | |
| **Test Case ID** | **Condition** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| 1.1 | Three possible rates of pay | Trying to assign pay rate outside of above generates error | As Expected | Pass |  |
| 1.2 | Change employee pay rate | Pay rate can be changed for employees | As Expected | Pass |  |
| 1.3 | Employees are assigned a pay rate on creation | Employee cannot be created without a pay rate | As Expected | Pass |  |

### 0004 – Employees – Wages

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0004 – Employees – Wages | | | | | |
| **Test Case ID** | **Condition** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| 1.1 | Enter weekly hours | I should be able to enter weekly hours worked for each current employee | As Expected | Pass |  |
| 1.2 | Minimum unit of time | The smallest allowable unit of work should be no less than ¼ of an hour | As Expected | Pass |  |
| 1.3 | Calculate pay | Gross wages should be calculated as hours worked \* hourly rate | As Expected | Pass |  |
| 1.4 | Can't process non-current employees | I should not be able to enter hours for non-employees | As Expected | Pass |  |

### 0005 – Employees – I/O

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0005 – Employees – I/O | | | | | |
| **Test Case ID** | **Condition** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| 1.1 | Read employees from file | Application should read employee list from a file. | As Expected | Pass |  |
| 1.2 | Save new employees to file | Application should save updated employee list to a file. | As Expected | Pass |  |
| 1.3 | Save processed wages to file | Once the wages are calculated, the result should be output to a file. | As Expected | Pass |  |
| 1.4 | Output file name | The name of the output file should include the date of the current week. | As Expected | Pass |  |
| 1.5 | Overwrite output file | If more than one file is produced in a week, the newer file should overwrite the older file. | As Expected | Pass |  |
| 1.6 | Employee file error | Should generate error if can’t open employee list | As Expected | Pass |  |
| 1.7 | Create employee file if not found | Should create employee list if can’t find employee list | As Expected | Pass |  |

### 0006 – Employees – Main UI Navigation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0006 – Employees – Main UI Navigation | | | | | |
| **Test Case ID** | **Condition** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| 1.1 | Input outside of accepted range generates error | Error Message | As Expected | Pass |  |
| 1.2 | Input 1 | Displays Employees | As Expected | Pass |  |
| 1.3 | Input 2 | Opens add employee menu | As Expected | Pass |  |
| 1.4 | Input 3 | Opens change employee status menu | As Expected | Pass |  |
| 1.5 | Input 4 | Opens change employee department menu | As Expected | Pass |  |
| 1.6 | Input 5 | Opens change employee pay rate menu | As Expected | Pass |  |
| 1.7 | Input 6 | Opens calculate payroll menu | As Expected | Pass |  |

## Bugs

### Bug 0001

**Bug ID:** 0001

**Bug Name:** Application crash on saving Payroll report  
**Area Path:** Main Menu >> Calculate Payroll (Option 6) >> enter values >> Save to file (option 1)  
**Build Number:** Version Number 1.1  
**Severity:** HIGH   
**Priority:** HIGH   
**Assigned to:** Philip Woulfe  
**Reported By:** Philip Woulfe  
**Reported On:** 30/05/2016  
**Reason:** Defect  
**Status:** Closed  
**Environment:** Windows 7/Visual Studio Community 2015

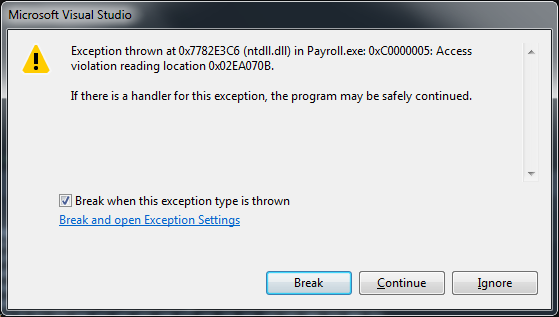
**Description:**  
Application crash on trying to calculate payroll and save to file

**Steps To Reproduce:**  
**1)** Start Application  
**2)** Select option 6 from the main menu  
**3)** Enter values for all employees  
**4)** Select option one from save to file menu  
**5)**Application will crash  
**6)**And also see the attached screenshot of the error page.

**Error Message:**

Exception thrown at 0x7782E3C6 (ntdll.dll) in Payroll.exe: 0xC0000005: Access violation reading location 0x02EA070B.

If there is a handler for this exception, the program may be safely continued.



**Expected result:** On selecting save, a new payroll report should be generated and the program should return to the main menu without crashing

**Update:**

Bug has been confirmed by Philip Woulfe

Bug is being fixed by Philip Woulfe

Bug is fixed, passing into System Test

Bug has been tested by Philip Woulfe

**Build:** 1.4

**Result:** Pass

**Comment:** Program returns to main menu without crashing

Bug has passed System Test and is now closed

### Bug 0002

**Bug Name:** Week end method generator doesn’t account for month limit  
**Bug ID:** 0002  
**Area Path:** Main Menu >> Calculate Payroll (Option 6) >> enter values  
**Build Number:** Version Number 1.2  
**Severity:** Medium  
**Priority:** **Medium**  
**Assigned to:** Philip Woulfe  
**Reported By:** Philip Woulfe  
**Reported On:** 30/05/2016  
**Reason:** Defect  
**Status:** Closed  
**Environment:** Windows 7/Visual Studio Community 2015

**Description:**  
When the week end date is generated, it doesn't account for the number of days in a month i.e. will return 36th May instead of 5th June

**Steps To Reproduce:**  
**1)** Start Application  
**2)** Select option 6 from the main menu  
**3)** Enter values for all employees  
**4)** On the payroll page, the date should be set to the Sunday of the current week, however, at the minute, it doesn’t account for the length of the month i.e. for the week of the 30th of May, if Payroll is run on Monday, the date will be set to 2016-05-36 instead of 2016-06-05

NB this same method is used to set the file name, so check that too

**Error Message:**





**Expected result:** Dates set by the *GetWEDate()* method should be valid dates

**Update:**

Bug has been confirmed by Philip Woulfe

Bug is being fixed by Philip Woulfe

Bug is fixed, passing into System Test

Bug has been tested by Philip Woulfe

Tested on the 31/05/2016

**Build:** 1.4

**Result:** Pass

**Comment:** Method now returns valid dates





Bug has passed System Test and is now closed

### Bug 0003

**Bug Name:** Unable to select employees when employee count is greater than 9  
**Bug ID:** 0003  
**Area Paths:**

Main Menu >> Change Employee Employment Status

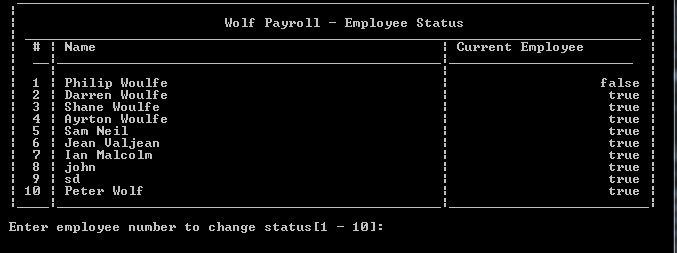
Main Menu >> Change Employee Department

Main Menu >> Change Employee Pay Rate  
**Build Number:** Version Number 1.3  
**Severity:** Medium  
**Priority:** **Medium**  
**Assigned to:** Philip Woulfe  
**Reported By:** Philip Woulfe  
**Reported On:** 30/05/2016  
**Reason:** Defect  
**Status:** Closed  
**Environment:** Windows 7/Visual Studio Community 2015

**Description:**  
The application takes the first char input when selecting from menus, this means if the employee number is greater than 9, any subsequent employees cannot be selected as the application will accept the 1 first and go to employee 1.

**Steps To Reproduce:**  
**1)** Start Application  
**2)** Select any of the menus affected  
**3)** Try to select an employee with an employee number greater than 9

**Error Message:**



**Expected result:** Any employee should be able to be selected

**Update:**

Bug has been confirmed by Philip Woulfe

Bug is being fixed by Philip Woulfe

Bug is fixed, passing into System Test

Bug has been tested by Philip Woulfe

Tested on the 31/05/2016

**Build:** 1.5

**Result:** Pass

**Comment:** All employees are now selectable

Bug has passed System Test and is now closed

## Recommendation for improvements

### Improve wage processing UI

The UI for entering the employee hours worked could be improved; we will look at this in sprint 2.

### Create individual payslips for employees

It would be beneficial to create individual payslips for each employee; we will look at this in sprint 2.