Igloo Brick Counter

# IPO Chart

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Radius, Entrance Length | Check validity of values | ‘Enter again’ or OK |
|  | Calculate no. of rows |  |
|  | Calculate no. of bricks per row |  |
|  | Calculate no. of bricks in entrance tunnel |  |
|  | Calculate no. of bricks to remove to allow for entrance hole |  |
|  | Calculate total bricks |  |
| Brick Type | Displays results for a certain brick type | No. of Bricks |

# Pseudo Code

BEGIN Calculate

Get *Radius*

Get *Entrance Length*

Set *Rows* to 0

Set *DomeTotal* to 0

Set *EntranceTotal* to 0

Set *Total* to 0

Set *Length* to {Brick Length}

Set *Height* to {Brick Height}

Set *Depth* to {Brick Depth}

DomeRows (*Radius*, *Rows*)

RowBricks (*Radius, Length, Height, Depth, Rows, DomeTotal*)

Entrance (*Entrance Length, EntranceTotal*)

*Total* = *DomeTotal* + *EntranceTotal*

PRINT *Total*

END Calculate

BEGIN SUBPROGRAM DomeRows (*Radius*, *Rows*)

*Rows* = {Calculation Here – No. of Rows in Dome}

END SUBPROGRAM DomeRows

BEGIN SUBPROGRAM RowBricks (*Radius, Length, Height, Width, Rows, Bricks, DomeTotal*)

Set *Count* to 0

Set *Temp* to 0

WHILE *Count* < *Rows*

*Temp* = {Calculation Here – No. of Bricks per Row - uses variables Length, Height, Width, Rows, Radius}

IF *Temp* needs bricks removed for entrance THEN

*Temp* = *Temp* \* {Calculation to remove bricks – returned as percentage}

ENDIF

*DomeTotal* = *DomeTotal* + *Temp*

Increment *Count*

ENDWHILE

ENDSUBPROGRAM RowBricks

BEGIN SUBPROGRAM Entrance (*Entrance Length, Length, Height, Depth, Rows, EntranceTotal*)

*EntranceTotal* = {Calculation Here – No. of Bricks in Entrance Tunnel - uses variables Length, Height, Width, Entrance Length}

END SUBPROGRAM Entrance

# Test Values

The decided minimum and maximum values for the internal radius of the dome were decided to be 0.5m and 1000m. Meanwhile, the decided minimum and maximum values for the entrance length were decided to be 0.1m and 100m. Whilst it is highly unlikely that a user would wish to enter values at the higher end of the decided range, if a user wishes to do so and it is within the program’s capabilities, it should be allowed. In addition, any errors whilst entering data should be easily rectified as the entered values are displayed again alongside the results and can be checked easily.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Significance |  | INPUT VALUES (m) | | OUTPUT VALUES (bricks) | | | Screenshot |
|  | | **Radius** | **Entrance Length** | **Budget** | **Superior** | **Luxury** |  |
| Minimum Values | | 0.5 | 0.1 | 38 | 46 | 65 | 1. |
| Beyond Minimum Values | | 0.3 | - | Please enter a radius between 0.50 & 1000 | | | 2. |
| Beyond Minimum Values | | - | 0.05 | Please enter a length between 0.10 & 100 | | | 3. |
| Maximum Values | | 1000 | 100 | 62863684 | 71851246 | 104779694 | 4. |
| Beyond Maximum Values | | 1500 | - | Please enter a radius between 0.50 & 1000 | | | Same as 2. |
| Beyond Maximum Values | | - | 150 | Please enter a length between 0.10 & 100 | | | Same as 3. |
| Acceptable Value | | 60 | 5 | 228102 | 261097 | 380558 | 5. |

# Screenshots

## 

## 

## 

## 

## 

# Social and Ethical Considerations

## Ease of Use

For this application, the application was designed to utilise a Graphical User Interface. This decision was made based on the experience of the users of the application, most of which will be likely more suited to using GUIs as can be seen from the most popular applications in use at this time such as Web Browsers and Document Processing software. Having a GUI means that a user is therefore more familiar with the use of a mouse to click on a button rather than having to type a command into a command-line interface. In addition, a GUI can feature a smaller learning curve as the user only needs to operate the program through the existing buttons and controls, which are provided onscreen, instead of remembering commands or keys to press to use the program. This means that if the client wishes to employ others to use this application, less training would be involved and therefore will result in time and cost savings. A mistyped command could also result in error messages which confuse the user whereas with a GUI a user cannot cause the program to execute a command that the developer has not provided an access point for. This allows the developer to control what they want the user to do or be able to do to complete a task resulting in less confusion if the interface is of an easy-to-use design. Furthermore, a GUI is capable of a greater range of prompts allowing for the program to alert the user if, for example in this case, a value is too high, and preventing the user from continuing on until the issue is resolved.

## Intellectual Property

Intellectual property is an important issue because it focuses on who has the rights to reproduce and recreate a particular piece of code. In this application it is required to consider who would own these rights after the project has been completed; whether it be the developers of the project or the client who requested the program to be designed by them. This will depend on whether the program was designed to be released as part of a commercial product or as a one-off. In the commercial case, then the developers may want to maintain ownership of the code as then they are able to distribute it to various customers including the original client. They are then also able to use the code in future projects and are responsible for the maintaining and support of the current application. However, on the other hand, if the project included direct input and utilised many original ideas direct from the client, then the client may have a claim to the application as well and may wish for the program to remain in their control to prevent distribution, especially when concerning their rival companies.

## Acceptable Response Time

As this program contains calculations that involve multiple repetitions of code and the requirement of multiple variables, it may cause a deteriorated performance when executed on low-end system. With this in mind, the application should either increase its speed to be less noticeable, or the more likely outcome of providing a method for users to be able to see the progress of what the application is performing or an indication that input from the user has been received. This indication is important as otherwise users get the impression that the program has stopped responding and will decide to cancel or close the operation. Most users expect an instant response to a command. This issue can be solved by providing a simple message or indication whilst the calculation is being performed to allow the user to at least know that the program is responding and can also act to inform the user that a wait is expected.