**IDENTIFIER DICTIONARY:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Identifier** | | | **Const/**  **Var/**  **Func** | **Data Type** | **Value** | Description |
| GetName() | | | func | string | User input | Function to get the birthday child’s name. |
| GetAge() | | | func | int | User input | Function to get the birthday child’s age. |
| GetGuests() | | | func | int | User input | Function to get the number of guests. |
| GetCakeCost() | | | func | float | User input | Function to get the cost of the cake. |
| GetFileName() | | | func | String | User input | Function to get the name of the output file. |
| PrintReport() | | | func | void | Non-value returning function | Prints report to output file partyReport |
| name | | | var | string | User input | Birthday child’s name. |
| age | | | var | int | User input | Birthday child’s age. |
| guests | | | var | int | User input | Number of guests invited. |
| cakeCost | | | const | float | User input | The cost of the cake. |
| fileName | | | var | string | User input | Name of the output file. |
| partyReport | | | var | ofstream | Output file | File to log reports. |
| children | | | var | int | guests + 1 | All children, including birthday child. |
| numBalloons | | | var | int | guests \* age | Each guest gets 1 balloon for each year the birthday child is old. |
| CalcBalloonsCost() | | | func | float | Returns total  See algorithm for selection statement | Determines the cost of each balloon depending upon the number of balloons. |
| local | costPerBalloon | | var | float | Determined by CalcBalloonsCost() | The cost of each balloon. |
| balloonsCost | | | var | float | CalcBalloonsCost() | Total cost of all balloons. |
| COST\_PER\_GIFT\_BAG | | | const | float | 4.15 | The cost of each gift bag. |
| numBags | | | var | int | guests | Each guest gets 1 gift bag. |
| giftBagsCost | | | var | float | guests \* COST\_PER\_GIFT\_BAG | Total cost of gift bags. |
| NAPKINS\_PER\_PACK | | | const | int | 12 | Number of napkins in each pack. |
| napkinPacks | | | var | int | ceil((children \* 4) / NAPKINS\_PER\_PACK) | Total napkin packs needed. Each child gets 4 napkins. |
| CalcNapkinsCost() | | | func | float | Returns total  See algorithm for selection statement | Determines the cost of napkins depending upon number of napkins. |
| local | numNapkins | | var | int | napkinPacks \* NAPKINS\_PER\_PACK | Total number of napkins. |
| local | costPerPack | | var | float | Determined by value of numNapkins | Cost per napkin pack. |
| napkinsCost | | | var | float | CalcNapkinsCost() | The total cost of napkin packs. |
| PLATES\_PER\_PACK | | | const | int | 8 | Number of plates in each pack. |
| platePacks | | | var | int | ceil((children \* 2) / PLATES\_PER\_PACK) | Total plate packs needed. Each child gets 2 plates. |
| CalcPlatesCost() | | | func | float | Returns total  See algorithm for selection statement | Determines the cost of plates depending upon number of plates. |
| local | | numPlates | var | int | platePacks \* PLATES\_PER\_PACK | Total number of plates. |
| local | | costPerPack | var | Float | Determined by value of numPlates | Cost per plate pack. |
| platesCost | | | var | float | CalcPlatesCost() | The total costs of plate packs. |
| COST\_PER\_PIZZA | | | const | float | 5.35 | The cost per pizza. |
| SLICES\_PER\_PIZZA | | | const | int | 12 | Number of slices per pizza. |
| pizzas | | | var | int | ceil((children \* 3) / SLICES\_PER\_PIZZA) | Total pizzas needed. Each child gets 3 slices of pizza. |
| pizzaCost | | | var | float | pizzas \* COST\_PER\_PIZZA | The total cost of pizza. |
| COST\_PER\_JUICE\_PACK | | | const | float | 4.25 | The cost of each juice box pack. |
| JUICE\_BOXES\_PER\_PACK | | | const | int | 10 | Number of juice boxes in each pack. |
| juiceBoxPacks | | | var | int | ceil((children \* 2) / JUICE\_BOXES\_PER\_PACK) | Total juice box packs needed. Each child gets 2 juice boxes. |
| juiceCost | | | var | float | juiceBoxPacks \* COST\_PER\_JUICE\_PACK | The total cost of juice box packs. |
| COST\_PER\_FULL\_SHEET | | | const | float | 15.75 | Cost per full sheet of cake. |
| COST\_PER\_HALF\_SHEET | | | const | float | 10.00 | Cost per half sheet of cake. |
| halfSheetCost | | | var | fkiat | If halfSheet is true, this is set to COST\_PER\_HALF\_SHEET, otherwise 0 | Total cost for half sheet. |
| fullSheetCost | | | var | float | fullSheets \* COST\_PER\_FULL\_SHEET | Total cost for full sheet(s). |
| halfSheet | | | var | bool | false | Whether or not we need a half sheet. |
| fullSheets | | | var | int | children / 50 | Max amount of full sheet cakes to feed children. |
| remainingChildren | | | var | int | children % 50 | Number of remaining children after buying full sheets. |
| total | | | var | float | balloonsCost + giftBagsCost + napkinsCost + platesCost + pizzaCost + juiceCost + fullSheetCost + halfSheetCost | Total cost for all supplies |

**Structure Chart (Show interface information):**

**A screenshot of a computer screen

Description automatically generated** Narrative: This program calculates the cost of throwing a child’s birthday party as determined by the age of the birthday child and the number of guests invited plus the cost of the birthday cake and outputs the report breaking down number of items and cost to a file name determined by the user.

**Function: void main()**

Algorithm:

* Declare variables
* Declare constants
* Welcome the user
  + Output “Welcome to the Birthday Party Cost Calculator”
* Input
  + Determine the name of the birthday child
    - name = GetName()
  + Determine the age of the birthday child
    - age = GetAge()
  + Determine the number of guests to be invited
    - guests = GetGuests()
  + Determine the name of the output file
    - fileName = GetFileName()
* Calculations
  + Set children = guests + 1
  + Calculate number and cost of balloons
    - SetnumBalloons = guests \* age
    - Set balloonCost = CalcBalloonsCost(numBalloons)
  + Calculate number and cost of gift bags
    - Set numBags = guests
    - Set giftBagsCost = numBags \* COST\_PER\_GIFT\_BAG
  + Calculate number and cost of napkin packs
    - Set napkinPacks = ceil((children \* 4) / NAPKINS\_PER\_PACK)
    - Set napkinsCost = CalcNapkinsCost(napkinPacks, NAPKINS\_PER\_PACK)
  + Calculate number and cost of plate packs
    - Set platePacks = ceil((children \* 2)) / PLATES\_PER\_PACK)
    - Set platesCost = CalcPlatesCost(platePacks, PLATES\_PER\_PACK)
  + Calculate number and cost of pizzas
    - Set pizzas = ceil((children \* 3)) / SLICES\_PER\_PIZZA)
    - Set pizzaCost = pizzas \* COST\_PER\_PIZZA
  + Calculate number and cost of juice packs
    - Set juiceBoxPacks = ceil((children \* 2) / JUICES\_BOXES\_PER\_PACK)
    - Set juiceCost = juiceBoxPacks \* COST\_PER\_JUICE\_PACK
  + Calculate the number of full and half sheet cakes and their respective costs
    - Set halfSheet = false
    - Set fullSheets = guests / 50
    - Set remainingChildren = children % 50
    - if (there are more than 25 remaining):
      * Set fullSheets += 1
    - else:
      * Set halfSheet = true (A half sheet will feed 25)
      * Set halfSheetCost = COST\_PER\_HALF\_SHEET
    - Set fullSheetCost = fullSheets \* COST\_PER\_FULL\_SHEET
  + Calculate the total cost
    - Set total = balloonsCost + giftBagsCost + napkinsCost + platesCost + pizzaCost + juiceCost + halfSheetCost + fullSheetCost
* Output
  + Call PrintReport()

**Function: string GetName()**

Narrative: Determines the value for name

Pre-condition: none

Post-condition: returns name

Algorithm:

* Input
  + Output “What is the birthday childs name?”
  + Store in name
* Output
  + Return name

**Function: int GetAge()**

Narrative: Determines the value for age

Pre-condition: none

Post-condition: returns age

Algorithm:

* Input
  + Output “How old will the birthday child be?”
  + Store in age
* Output
  + Return age

**Function: int GetGuests()**

Narrative: Determines the value for guests

Pre-condition: none

Post-condition: returns guests

Algorithm:

* Get user input
  + Output “How many invited guests?”
  + Store in guests
* Output
  + Return guests

**Function: float GetCakeCost()**

Narrative: Determines the value for cakeCost **(Not Used in Current Program)**

Pre-condition: none

Post-condition: returns cakeCost

Algorithm:

* Get user input
  + Output “How much is the birthday cake? $”
  + Store in cakeCost
* Output
  + Return cakeCost

**Function: string GetFileName()**

Narrative: Determines the value for fileName

Pre-condition: none

Post-condition: returns fileName

Algorithm:

* Get user input
  + Output “What would you like to name the output file?”
  + Store in fileName
* Output
  + Return fileName

**Function: float CalcBalloonsCost(int numBalloons)**

Narrative: Determines the value for balloonsCost

Pre-condition: numBalloons

Post-condition: returns total to balloonsCost

Algorithm:

* Calculations
  + Determine the cost per balloon
    - if (numBalloons > 100):
      * Set costPerBalloon = .25
    - else if (numBalloons >= 50):
      * Set costPerBallon = .35
    - else:
      * Set costPerBalloon = .45
  + Calculate the total cost of the ballons
    - Set total = numBalloons \* costPerBalloon
* Output
  + Return total

**Function: float CalcNapkinsCost(int napkinPacks, int NAPKINS\_PER\_PACK)**

Narrative: Determines the value for napkinsCost

Pre-condition: napkinPacks, NAPKINS\_PER\_PACK

Post-condition: returns total to napkinsCost

Algorithm:

* Calculations
  + Calculate to total number of napkins
    - Set numNapkins = napkinPacks \* NAPKINS\_PER\_PACK
  + Determine the cost per napkin pack
    - if (numNapkins >= 300):
      * Set costPerPack = 1.50
    - else if (numNapkins > 100):
      * Set costPerPack = 2.50
    - else:
      * Set costPerPack = 3.00
  + Calculate the total cost of the ballons
    - Set total = napkinPacks \* costPerPack
* Output
  + Return total

**Function: float CalcPlatesCost(int** **platePacks, int** **PLATES\_PER\_PACK)**

Narrative: Determines the value for napkinsCost

Pre-condition: platePacks, PLATES\_PER\_PACK

Post-condition: returns total to platesCost

Algorithm:

* Calculations
  + Calculate to total number of plates
    - Set numPlates = platePacks \* PLATES\_PER\_PACK
  + Determine the cost per plate pack
    - if (numPlates > 50):
      * Set costPerPack = 3.00
    - else:
      * Set costPerPack = 3.10
  + Calculate the total cost of the ballons
    - Set total = platePacks \* costPerPack
* Output
  + Return total

**Function: void PrintReport()**

Narrative: prints the report to output file, name determined by the user, in the same directory as the program.

Pre-condition*:* fullSheets, halfSheet, fileName, name, numBalloons, balloonsCost, numBags, giftBagsCost, napkinPacks, napkinsCost, platePacks, platesCost, pizzas, pizzaCost, juiceBoxPacks, juiceCost, halfSheetCost, fullSheetCost, total

Post-condition: prints the report the output file determined by user

Algorithm:

* Open file stream
  + Declare ofstream variable partyReport
  + Open fileName using partyReport in append mode.
* Output
  + Print the report to output file (output in the following format, numbers to two decimals):

Report for name’s birthday party:

Balloons: numBalloons $ balloonsCost

Gift bags: numBags $ giftBagsCost

Napkin packs: napkinPacks $ napkinsCost

Plate packs: platePacks $ platesCost

Pizzas: pizzas $ pizzaCost

Juicebox Packs: juiceBoxPacks $ juiceCost

★ Cake - sheet(s): fullSheets $ fullSheetCost

★ Cake – half-sheet: 1 $ halfSheetCost

Total cost: $ total

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Note: ★ = if (fullSheets) / ★ = if (halfSheet)