**IDENTIFIER DICTIONARY:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Identifier** | | | **Const/**  **Var/**  **Func** | **Data Type** | **Value** | Description |
| COST\_PER\_GIFT\_BAG | | | const | float | 4.15 | The cost of each gift bag. |
| NAPKINS\_PER\_PACK | | | const | int | 12 | Number of napkins in each pack. |
| PLATES\_PER\_PACK | | | const | int | 8 | Number of plates in each pack. |
| SLICES\_PER\_PIZZA | | | const | int | 12 | Number of slices per pizza. |
| COST\_PER\_PIZZA | | | const | float | 5.35 | The cost per pizza. |
| JUICE\_BOXES\_PER\_PACK | | | const | int | 10 | Number of juice boxes in each pack. |
| COST\_PER\_JUICE\_PACK | | | const | float | 4.25 | The cost of each juice box pack. |
| COST\_PER\_HALF\_SHEET | | | const | float | 10.00 | Cost per half sheet of cake. |
| COST\_PER\_FULL\_SHEET | | | const | float | 15.75 | Cost per full sheet of cake. |
| response | | | var | char | User input | The user’s response to the question as to whether to run the program again. |
| currentPartyCost | | | var | float | Calculated by GenerateReport() | The total cost of the current party. |
| numParties | | | var | int | 0, increased by GenerateReport() | Number of party reports run. |
| partiesTotalCost | | | var | float | += currentPartyCost | Total of all parties run through the program. |
| fileName | | | var | string | User input | Name of the output file. |
| WelcomeMessage() | | | func | void | None | A welcome message to the user. |
| local | artFileName | | var | string | “art/” + argument + “.txt” | The file to read the welcome art from. |
| local | artFile | | var | ifstream | artFileName | Stream to import the art. |
| GetFileName() | | | func | String | User input | Function to get the name of the output file. |
| GenerateReport() | | | func | void | None | Generates the report for a party. |
| local | name | | var | string | User input | Birthday child’s name. |
| local | age | | var | int | User input | Birthday child’s age. |
| local | guests | | var | int | User input | Number of guests invited. |
| local | numBalloons | | var | int | guests \* age | Each guest gets 1 balloon for each year the birthday child is old. |
| local | balloonsCost | | var | float | Determined by numBallons | Total cost of all balloons. |
| local | numBags | | var | int | guests | Each guest gets 1 gift bag. |
| local | giftBagsCost | | var | float | guests \* COST\_PER\_GIFT\_BAG | Total cost of gift bags. |
| local | napkinPacks | | var | int | ceil((children \* 4) / NAPKINS\_PER\_PACK) | Total napkin packs needed. Each child gets 4 napkins. |
| local | napkinsCost | | var | float | napkinPacks \* costPerPack | The total cost of napkin packs. |
| local | platePacks | | var | int | ceil((children \* 2) / PLATES\_PER\_PACK) | Total plate packs needed. Each child gets 2 plates. |
| local | platesCost | | var | float | platePacks \* costPerPack | The total costs of plate packs. |
| local | pizzas | | var | int | ceil((children \* 3) / SLICES\_PER\_PIZZA) | Total pizzas needed. Each child gets 3 slices of pizza. |
| local | pizzaCost | | var | float | pizzas \* COST\_PER\_PIZZA | The total cost of pizza. |
| local | juiceBoxPacks | | var | int | ceil((children \* 2) / JUICE\_BOXES\_PER\_PACK) | Total juice box packs needed. Each child gets 2 juice boxes. |
| local | juiceCost | | var | float | juiceBoxPacks \* COST\_PER\_JUICE\_PACK | The total cost of juice box packs. |
| local | halfSheet | | var | bool | false | Whether or not we need a half sheet. |
| local | fullSheets | | var | int | children / 50 | Max amount of full sheet cakes to feed children. |
| local | halfSheetCost | | var | fkiat | If true:  COST\_PER\_HALF\_SHEET  Else: 0 | Total cost for half sheet. |
| local | fullSheetCost | | var | float | fullSheets \* COST\_PER\_FULL\_SHEET | Total cost for full sheets. |
| local | children | | var | int | guests + 1 | All children, including birthday child. |
| GetName() | | | func | string | User input | Gets the birthday child’s name. |
| IsNumber() | | | func | bool | true / false | Checks whether the input has a number in it. |
| GetAge() | | | func | int | User input | Gets the birthday child’s age. |
| GetGuests() | | | func | int | User input | Gets the number of guests. |
| CalcBalloons() | | | func | void | None | Calculates the number and cost of ballons. |
| local | costPerBalloon | | var | float | Determined by numBalloons | The cost of each balloon. |
| CalcBags() | | | func | void | None | Calculates the number and cost of gift bags. |
| CalcNapkins () | | | func | void | None | Calculates the number of napkin packs needed and cost 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. |
| CalcPlates() | | | func | void | None | Calculates the number of plate packs needed and cost 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. |
| CalcPizzas() | | | func | void | None | Calculates the number of pizzas needed and their total cost. |
| CalcJuice() | | | func | void | None | Calculates the number of juice box packs needed and their total cost. |
| CalcCake() | | | func | void | None | Calculates number of half and full sheet cakes needed and their total cost. |
| local | | remainingChildren | var | int | children % 50 | Number of remaining children after buying full sheets cakes. |
| PrintReport() | | | func | void | None | Appends results of GenerateReport() to output file fileName. |
| local | | partyReport | var | ofstream | Output file | File to log reports. |
| PrintSummary() | | | func | void | None | Appends summary of parties to the output file fileName. |
| local | | partyAverage | var | float | partiesTotalCost / numParties | The average cost of each party. |
| local | | partyReport | var | ofstream | Output file | File to log summary. |
| ExitMessage() | | | func | void | None | An exit message to the user. |

**A screenshot of a computer

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 and outputs the report to a file determined by the user, breaking down number of items and cost. It can calculate multiple party reports and will print a summary containing the total number of parties calculated, the total cost of all parties, and the average cost for each party at the bottom of the file. It will also output the summary to the console.

**Function: void main()**

Algorithm:

* Declare variables
* Welcome the user
  + Call WelcomeMessage()
* Input
  + Determine the name of the output file
    - fileName = GetFileName()
  + Do, while (reponse == ‘Y’)
    - Call GenerateReport()
    - partiesTotalCost += currentPartyCost
    - reponse = ContinuePrompt()
* Output
  + Call PrintSummary()
  + Call ExitMessage()

**Function: string GetFileName()**Narrative: Determines the value for fileNamePre-condition: nonePost-condition: returns fileNameAlgorithm:

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

**Function: bool IsNumber(string str)**

Narrative: Checks whether str contains a number, and returns true if it does.

Pre-condition: str

Post-condition: GetName() will continually prompt the user while IsNumber() returns true

Algorithm:

* For each character in the string:
  + Check if the character is a digit
    - If it is, return true
* Return false

**Function: string GetName()**

Narrative: Determines the value for name

Pre-condition: none

Post-condition: returns name

Algorithm:

* Input
  + Do, while IsNumber(name):
    - Output “What is the birthday childs name?”
    - Store in name
    - if (name contains a number):
      * Output “No digits allowed in name!”
* Output
  + Return name

**Function: int GetAge()**

Narrative: Determines the value for age

Pre-condition: none

Post-condition: returns age

Algorithm:

* Input
  + Do, while (age < 1 || age >= 120):
    - Output “How old will the birthday child be? ”
    - Store in age
    - while (cin.fail()):
      * Output “INPUT ERROR, Please enter an integer between 1-119: “
      * Output “How old will the birthday child be? “
      * Store in age
    - if (age < 1 || age >= 120):
      * Output “RANGE ERROR, Please enter a value between 1-119: “
* Output
  + Return age

**Function: int GetGuests()**

Narrative: Determines the value for guests

Pre-condition: none

Post-condition: returns guests

Algorithm:

* Input
  + Do, while (guests < 1 || guests >= 500):
    - Output “How many invited guests? ”
    - Store in guests
    - while (cin.fail()):
      * Output “INPUT ERROR, Please enter an integer between 1-499: “
      * Output “How old will the birthday child be? “
      * Store in age
    - if (guests < 1 || guests >= 500):
      * Output “RANGE ERROR, Please enter a value between 1-499: “
* Output
  + Return guests

**Function: bool ContinuePrompt(char response)**

Narrative: Asks the user if they would like to process another party, and returns their reponse

Pre-condition: response

Post-condition: returns char response to main()

Algorithm:

* Input
  + Do, while (response != ‘Y’ && response != ‘N’):
    - Output “Would you like to process another party (Y/N)? “
    - Store in response
    - Set response = toupper(response)
    - if (response != ‘Y’ && response != ‘N’):
      * Output “ERROR, Please enter Y or N! “
* Output
  + Return response

**Function: void CalcBalloons(int age, int guests, int& numBalloons, float& balloonsCost)**

Narrative: Determines the values for numBalloons and balloonsCost

Pre-condition: age, guests, numBalloons, balloonsCost

Post-condition: Updates numBalloons, balloonsCost

Algorithm:

* Calculations
  + SetnumBalloons = guests \* age
  + 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 balloonsCost = numBalloons \* costPerBalloon

Function: void CalcBags(int guests, int& numBags, float& giftBagsCost)

Narrative: Determines the values for numBags and giftBagsCost

Pre-condition: guests, numBags, giftBagsCost, COST\_PER\_GIFT\_BAG

Post-condition: Updates numBags and giftBagsCost

Algorithm:

* Calculations
  + Set numBags = guests
  + Set giftBagsCost = numBags \* COST\_PER\_GIFT\_BAG

**Function: void CalcNapkins(int children, int& napkinPacks, float& napkinsCost)**

Narrative: Determines the values for napkinPacks and napkinsCost

Pre-condition: children, napkinPacks, napkinsCost, NAPKINS\_PER\_PACK

Post-condition: Updates napkinPacks and napkinsCost

Algorithm:

* Calculations
  + Set napkinPacks = ceil((children \* 4) / NAPKINS\_PER\_PACK)
  + 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 napkins
    - Set napkinsCost = napkinPacks \* costPerPack

**Function: void CalcPlates(int children, int& platePacks, int& platesCost)**

Narrative: Determines the values for platePacks and platesCost

Pre-condition: children, platePacks, platesCost, PLATES\_PER\_PACK

Post-condition: Updates platePacks and platesCost

Algorithm:

* Calculations
  + Set platePacks = ceil((children \* 2)) / PLATES\_PER\_PACK)
  + 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 platesCost = platePacks \* costPerPack

**Function: void CalcPizzas(int children, int& pizzas, int& pizzaCost)**

Narrative: Determines the values for pizzas and pizzaCost

Pre-condition: children, pizzas, pizzaCost, SLICES\_PER\_PIZZA, COST\_PER\_PIZZA

Post-condition: Updates pizzas and pizzaCost

Algorithm:

* Calculations
  + Set pizzas = ceil((children \* 3)) / SLICES\_PER\_PIZZA)
  + Set pizzaCost = pizzas \* COST\_PER\_PIZZA

**Function: void CalcJuice(int children, int& juiceBoxPacks, int& juiceCost)**

Narrative: Determines the values for juiceBoxPacks and juiceCost

Pre-condition: children, juiceBoxPacks, juiceCost, JUICE\_BOXES\_PER\_PACK, COST\_PER\_JUICE\_PACK

Post-condition: Updates juiceBoxPacks and juiceCost

Algorithm:

* Calculations
  + Set juiceBoxPacks = ceil((children \* 2) / JUICES\_BOXES\_PER\_PACK)
  + Set juiceCost = juiceBoxPacks \* COST\_PER\_JUICE\_PACK

**Function: void CalcCakes(int children, bool& halfSheet, int& fullSheets, float& halfSheetCost, float& fullSheetCost)**

Narrative: Determines the values for halfSheet, fullSheets, halfSheetCost and fullSheetCost

Pre-condition: children, halfSheet, fullSheets, halfSheetCost, fullSheetCost, COST\_PER\_HALF\_SHEET, COST\_PER\_FULL\_SHEET

Post-condition: Updates halfSheet, fullSheets, halfSheetCost and fullSheetCost

Algorithm:

* Calculations
  + Set halfSheet = false
  + Set fullSheets = guests / 50
  + Set remainingChildren = children % 50
  + if (there are more than 25 remaining):
    - Set fullSheets += 1
  + else if (there are more than none remaining):
    - Set halfSheet = true (A half sheet will feed 25)
    - Set halfSheetCost = COST\_PER\_HALF\_SHEET
  + Set fullSheetCost = fullSheets \* COST\_PER\_FULL\_SHEET

Function: void WelcomeMessage(string artName)

Narrative: Welcomes the user to the program with a custom art ASCII.

Pre-condition: artName

Post-condition: Prints the artName to the console.

Algorithm:

* Set artFileName = “art/” + artName + “.txt”
* Open artFileName with ifstream artFile
* Output artFile to the console.
* Close artFile

**Function: void GenerateReport()**

**Narrative:**

**Pre-condition:**

**Post-condition:**

Algorithm:

* Declare variables
* 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()
* Calculations
  + Set children = guests + 1
  + Call CalcBalloons()
  + Call CalcBags()
  + Call CalcNapkins()
  + Call CalcPlates()
  + Call CalcPizzas()
  + Call CalcJuice()
  + Call CalcCake()
  + Set currentPartyCost = balloonsCost + giftBagsCost + napkinsCost + platesCost + pizzaCost + juiceCost + halfSheetCost + fullSheetCost
* Output
  + Call PrintReport()
  + Set numParties += 1

**Function: void PrintReport()**

Narrative: Prints the party report to output file, name determined by the user.

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

Post-condition: Prints the party 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 birthday child: name

Birthday child’s age: age

Number of invited guests: guests

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: $ currentPartyCost

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

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

* + Close ofstream partyReport

**Function: void PrintSummary(string fileName, int numParties, float partiesTotalCost)**

Narrative: Prints the summary of the reports including number of parties run, total cost of all parties, and average cost of each party to both the file determined by the user and the terminal.

Pre-condition: fileName, numParties, partiesTotalCost

Post-Condition: Prints the summary of the reports including number of parties run, total cost of all parties, and average cost of each party to both the file determined by the user and the terminal.

Algorithm:

* Calculations
  + Set partyAverage = partiesTotalCost / numParties
* Open file stream
  + Declare ofstream variable partyReport
  + Open fileName using partyReport in append mode.
* Output
  + Print the summary to output file (output in the following format, numbers to two decimals):

Number of parties: numParties

Total cost of all parties: $ totalPartiesCost

Average spent on each party: $ partyAverage

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

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

* Close ofstream partyReport

**Function: void ExitMessage(string adj)**

Narrative: Thank the user with a dynamic exist message.

Pre-condition: adj

Post-condition: Prints the exit message to the console.

Algorithm:

* Set hour to the current hour in 24 hour format
* Set day to the current day of the week
* Set intHour to hour
* if (intHour <= 10):
  + timeOfDay = 0
* else if (intHour > 10 && intHour <= 16):
  + timeOfDay = 1
* else
  + timeOfDay = 2
* Output
  + “Thanks for using the Birthday Party Cost Calculator v.3.1”
  + switch (timeOfday):
    - case 0:
      * Output “Have a “ + adj + “ “ + day + “ morning!”
    - case 1:
      * Output “Have a “ + adj + “ “ + day + “ afternoon!”
    - case 2:
      * Output “Have a “ + adj + “ “ + day + “ evening!”