Contents

[Algorithm: 2](#_Toc19050145)

[Generalized-character-class characteristics: 3](#_Toc19050146)

[Generalized-tool-class characteristics: 5](#_Toc19050147)

[Program characteristics: 6](#_Toc19050148)

# Algorithm:

1. Greet user
2. Ask user to enter character name and select character class.
3. Repeat 3. for another user.
4. Ask if would like to add more players.
5. If yes, do 3.
6. If no, break;
7. Until one player is left:
8. Ask p1 what would like to do:
9. Choose tools unique to class, then attack
10. Health pack then attack?
11. Defence boost then attack?
12. Attack.
13. Execute what the player chooses.
14. Record order of deaths for leaderboard at end of game.

Code for a. and b.:

userChoose(Character\*ownedByUser, Tool\*tool1owned,Tool\*tool2owned){

if (tool1owned==NULL){

cout<<””;

}

Else{

cout<<tool1owned->number<<”. ”;

cout<<tool1owned->function<<endl;

}

if(tool2owned==NULL){

cout<<””;

}

Else{

cout<<tool2owned->number<<”. ”;

cout<<tool2owned->function<<endl;

}

}

1. Display leaderboard.
2. (Ask if want to play again?).
3. Game over.

# Generalized-character-class characteristics:

1. Note that the base class will undergo polymorphism.
2. Health, attack and defence (private).
3. Has perceived health (public).
4. Percieved health = health + defense.
5. New health = perceived health – enemy attack. //utilized by tools/enemies.
6. Has pointers to addresses of its tools (private).
7. Has a pointer to itself (public to allow easy interaction).
8. Has a pointer to its owner to allow for the tool to set the pointer itself as NULL.
9. Ability to create own tools, link itself to those tools upon creation and link those tools to eachother.

Code:

aCharacterType(){//begin constructor

createTools(Character\*self){

//create tools

ClassTools\*one;

ClassTools\*two;

//link self to tools.

this->pointerToTool1=one;

this->pointerToTool2=two;

//link tools to eachother

//link tool1 prev to NULL. (set as default for all pointers.)

//link tool1 next to tool2.

//link tool2 prev to tool1.

//link tool2 next to NULL.

}

}//end constructor

1. Ability to use tools on any character.

Code:

useTool(Character\*user, Character\*target){

if(this->selectedToolPointer!=NULL){

(this->selectedToolPointer)->toolUse(target);

(this->selectedToolPointer)=&(selectedToolPointer->pointerToNextTool);

user->pointerToThisTool=NULL;

}

//add feature to selection function to only allow user to choose tools that have not //been used by checking for in each of the user’s character’s owned tool pointers for //NULL.

}

1. Ability to take damage on behalf of any character except the one that it is attacking or its only enemy.
2. Ability to give its tools to any character.

Code:

giveTool(Character\*target){

target->extraToolPointer=this->selectedToolPointer;

this->selectedToolPointer=NULL;//ensure that donor cannot use this tool by //employing an if loop to check if selected tool to be used is NULL before use.

}

# Generalized-tool-class characteristics:

1. Note that the base class will undergo polymorphism.
2. Has pointer pointing to the tool previous in number.

Code:

Tool\* prevTool=NULL;

1. Has a pointer pointing to the tool next in number.

Code:

Tool\* nexTool=NULL;

1. Has string describing its function.

Code:

string function;

1. Has an integer that indicates its number in relation to the tools owned by its owner. /\*when tools are used and transferred between characters, implement number and pointer transfer via linked list.\*/

Code for number:

Int number;

1. Has a characteristic modifier (amount) and target value to be modified (ex: health [perceived], attack etc), that is unique to each morph of the base class.

Int modifier;

1. //Has a Boolean variable that is changed to false once used/transferred by owner object.
2. 7. Is replaced by checking for NULL pointer to used/transferred tool.
3. Code for when used:

useTool(Character\*target){

target->affectedTargetCharacteristic=(target->percievedTargetCharacteristic) – (amount deducted by that morph of tool);

//note that if there is no perceived characteristic, then the realAndCurrentCharacteristic is //used

//link prev tool’s next pointer to address of this used tool’s next address.

}

# Program characteristics:

1. Saves all written output for display at end of program (with timestamps) and to be saved in a text file called “log.txt”.
2. Clearly separates each action visually (via ----------).
3. Clearly separates each round visually (via ===========).