Министерство образования и науки Российской Федерации

МГТУ им .Н.Э Баумана

Факультет «Робототехника и комплексная автоматизация»

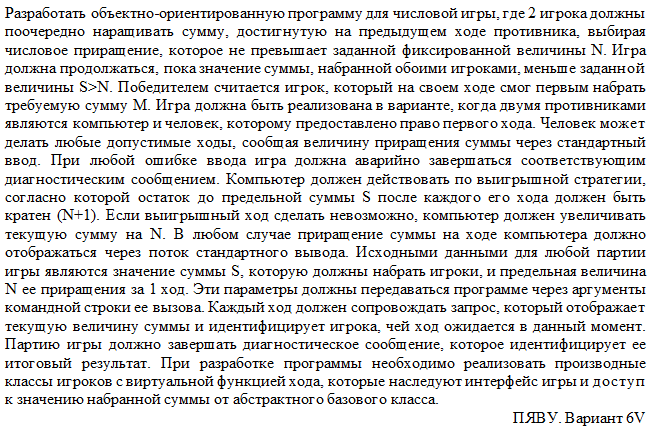
Кафедра «РК6»

**Отчет по лабораторной работе**

По курсу «Объектно-ориентированное программирование»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Выполнил: |  |  | Студент | Журавлев Н.В. |
|  |  |  | Группа | РК6-12Б |
|  |  |  |  |  |
| Проверил: |  |  |  | Волосатова Т.М. |
|  |  |  | Дата | 21.04.2020 |
|  |  |  | Подпись | \_\_\_\_\_\_\_\_\_\_\_ |

**РК**



**Описание алгоритма**

1) Описание класс Gambler и с методами move и query и виртуальным деструктором и класс кучи с полем размером максимального приращения кучи и размером кучи и методами её увеличения и получением размера максимального приращения.

2) Наследование от него 2 классов игрока и компьютера.

3) Цикл while в котором поочерёдно выбирается move для компьютера и игрока(первый ход игроку), пока сумма не будет равна предельной сумме кучи.

4) Для игрока вводится размер добавления в кучу, проверка на соответствие правилам, затем увеличение кучи на это число.

5) Для компьютера рассчитывается по стратегии, согласно которой остаток до предельной суммы после каждого его хода должен быть кратен максимальному числу приращений +1. Если выигрышный ход сделать невозможно, компьютер должен увеличивать текущую сумму на кол-во максимальный приращений.

6)

**Текст программы**

#include <cstdlib>

#include <iostream>

using namespace std;

class Heap {

private:

int size;

int lim;

public:

Heap(int s, int l) : size(s), lim(l) {};

int get\_lim() {return lim;}

int get(int n=0) {size += n;return (size);}

};

class Gambler {

protected:

Heap\* heap;

const int limit;

const char\* name;

public:

Gambler(Heap& h, int l) : limit(l) {heap = &h;}

int query();

virtual int move()=0;

virtual ~Gambler(){};

};

int Gambler::query() {

cout << "Heap = "<< heap->get()<< name << " > ";

return heap->get();

}

class Man : public Gambler {

public:

Man (Heap& h, int l, const char\* n) : Gambler(h, l){name = n;}

virtual int move();

};

class Pen : public Gambler {

public:

Pen (Heap& h, int l, const char\* n) : Gambler(h, l){name = n;}

virtual int move();

};

int Man::move() {

int g;

cin>>g;

if((g < 1) || (g > limit))

g = -1;

return heap->get(g);

}

int Pen::move() {

int h = heap->get\_lim() - heap->get();

int rest = 0;

while (((((h - rest) % (limit + 1)) != 0 ) && rest < limit) || rest == 0 ) {

rest++;

if ((h - rest) == 0) {

h = heap->get(rest);

cout << rest << endl;

return h;

}

}

h = heap->get(rest);

cout << rest << endl;

return h;

}

int main(int argc, char \* argv[]) {

if (argc != 3) {

printf("Неверное количество аргументов\n");

return 1;

}

if (atoi(argv[1]) == 0 || atoi(argv[2]) == 0 || atoi(argv[1]) <= atoi(argv[2])) {

printf("Неверный ввод\n");

return 1;

}

int i = 0;

int l = atoi(argv[1]);

Heap h(0, l);

Gambler \*g[] = {new Man(h,atoi(argv[2])," Man"), new Pen(h, atoi (argv[2]), " Pen")};

while(g[i]->query() != l) {

int a = h.get();

g[i]->move();

if(a > h.get() || h.get() > l) {

printf("Неверный ход\n");

return 0;

}

if (++i > 1)

i = 0;

}

if(i == 0) {

cout << "Looser\nHeap = "<< h.get() << " Pen > Winner\n";

} else {

cout << "Looser\nHeap = "<< h.get() << " Man > Winner\n";

}

delete g[0];

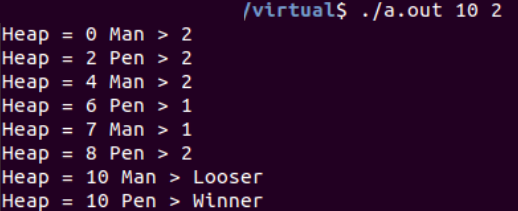
delete g[1];

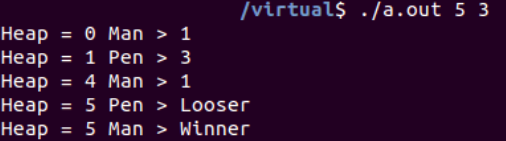
return 0;

}

**Тесты**

**Ввод Вывод**

10 2

5 3

3 5



**Результат**

Чья-то победа

**Списки литературы**

1)Волосатова Т. М., Родионов С. В. «Объектно-ориентированное программирование на С++» 2)http://bigor.bmstu.ru/?cnt/?doc=VU/base.cou