# Алгоритм

Main

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | предикат | действие | номер перехода |
| 1 |  | string s; string sroot; string child; string findway, return\_ans;  cin>>sroot; //sroot = "root"; // Base\* app\_root = new Base(sroot); string sname; int number1, number2; cin>>sname; string cond; cond = "endtree"; | 2 |
| 2 | (sname.compare(cond) != cond.compare(cond)) | cin>>child; cin>>number1; cin>>number2; | 3 |
|  |  |  | 7 |
| 3 | (number1 == 4) | new FourthDerivative(app\_root->FindName(sname), child, number2, sname); | 5 |
|  |  |  | 4 |
| 4 | (number1 == 2) | new SecondDerivative(app\_root->FindName(sname), child, number2, sname); | 5 |
|  |  | new SixthDerivative(app\_root->FindName(sname), child, number2, sname); | 5 |
| 5 |  | cin >> sname; | 6 |
| 6 | (sname.compare(cond) == cond.compare(cond)) | break; | 7 |
|  |  |  | 2 |
| 7 |  | string child\_s[99]; string child\_to[99]; string obj\_to; int max\_child = 0; string message; int way = 1; int j = 0; app\_root->PrintChildren(app\_root); cout << endl << "Set connects"; cin >> way; | 8 |
| 8 | (!(way == 0)) | cin >> child\_s[j]; cin >> child\_to[j]; cout<<endl<<way<<" " <<child\_s[j]<<" "<<child\_to[j]; cin >> way; j++; max\_child = j; | 8 |
|  |  |  | 9 |
| 9 |  | string child\_s[99]; string child\_to[99]; string obj\_to; int max\_child = 0; string message; int way = 1; int j = 0; app\_root->PrintChildren(app\_root); cout << endl << "Set connects"; cin >> way; | 10 |
| 10 | ((obj\_to != "endsignals")) | cin >> message;  j = 0; | 11 |
|  |  |  | Ø |
| 11 | (j <= max\_child) |  | 12 |
|  |  |  | 13 |
| 12 | ( (app\_root->FindAll(obj\_to)!="0")&&(app\_root->FindAll(child\_to[j])!="0")&&(obj\_to == child\_s[j])) | cout << endl << "Signal to " << child\_to[j] << " Text: " << child\_s[j] << " -> " << message;  j++; | 12 |
|  |  | j++; | 12 |
| 13 |  | cin >> obj\_to; | 11 |

string Base::FindAll(const string& name)

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | предикат | действие | № перехода |
| 1 | (\_name == name) | return name; | Ø |
|  |  |  | 2 |
| 2 |  | int i = 0 | 3 |
| 3 | i <  \_children.size() |  | 5 |
|  |  |  | 6 |
| 4 |  | i++ | 3 |
| 5 | (i->FindWay(way) != " Object not found") | return i->FindWay(way); | Ø |
|  |  |  | 4 |
| 6 |  | return "0"; | Ø |

# Методы

Использую потоки ввода и вывод "cout", "cin".

Объекты класса base.

Описание base:

vector<Base\*> \_children;

string \_name;

string \_way;

bool \_isReady = false;

методы:

добавление дочернего объекта void AddChild(Base\* child);

вывод дочернего объекта void PrintChildren(Base\* rootName, bool condition = false, int tab=0);

вывод табуляции void Tabulation(int tab);

нахождение объекта по имени Base\* FindName(const string& way);

нахождение объекта по имени для вывода string FindAll(const string& way);

установка готовности объектов void GetReady();

объекты классов class FourthDerivative: public Base, class ThirdDerivative: public FourthDerivative, class SixthDerivative: public FourthDerivative

конструкторы классов base, FourthDerivative, ThirdDerivative, SixthDerivative определяют свойства.

Вывод

root

root object\_1 3 1

root object\_2 2 1

object\_2 object\_4 3 -1

object\_2 object\_5 3 1

root object\_3 3 1

object\_2 object\_6 2 1

object\_1 object\_7 2 1

endtree

1 object\_3 object\_4

2 object\_2 object\_6

3 object\_3 object\_5

0

object\_3 "Hi!"

object\_2 "Hello!"

endsignals

Object tree

root

object\_1

object\_7

object\_2

object\_4

object\_5

object\_6

object\_3

Set connects

1 object\_3 object\_4

2 object\_2 object\_6

3 object\_3 object\_5

Emit signals

Signal to object\_4 Text: object\_3 -> "Hi!"

Signal to object\_5 Text: object\_3 -> "Hi!"

Signal to object\_6 Text: object\_2 -> "Hello!"





