

ОТЧЕТ

о выполнении практической (лабораторной) работы

Наследование классов

Выполнили:

2ИСП 11-24Вб

Дубцова С.А.

Муратов С.С

Проверил: преподаватель

Карпов Р.В.

2022

Оглавление

[Код программы версия 1 3](#_Toc119571857)

[Диаграмма классов для версии 1 7](#_Toc119571858)

[Вывод для версии 1 7](#_Toc119571859)

[Код программы версия 2 8](#_Toc119571860)

[Диаграмма классов для версии 2 14](#_Toc119571861)

[Вывод программы для версии 2 15](#_Toc119571862)

[Вывод 27](#_Toc119571863)

# Код программы версия 1

#include <iostream>

#include <string>

using namespace std;

class Human {

public:

int weight, height, age;

string name, education;

void setWeigth(int weight) {

if (weight > 0) {

this->weight = weight;

}

else {

cout << "Error" << endl;

}

}

void setHeigth(int height) {

if (height > 0) {

this->height = height;

}

else {

cout << "Error" << endl;

}

}

void setAge(int age) {

if (age >= 18) {

this->age = age;

}

else {

cout << "Error" << endl;

}

}

void setName(string name) {

if (name == "") {

cout << "Error" << endl;

}

else {

this->name = name;

}

}

void setEducation(string education) {

if (education == "") {

cout << "Error" << endl;

}

else {

this->education = education;

}

}

int getWeight() {

return weight;

}

int getHeight() {

return height;

}

int getAge() {

return age;

}

string getName() {

return name;

}

string getEducation() {

return education;

}

void getInfo() {

cout << "Name: " << getName() << endl

<< "Age: " << getAge() << endl

<< "Weight: " << getWeight() << endl

<< "Height: " << getHeight() << endl

<< "Education: " << getEducation() << endl;

}

};

class Person : public Human {

public:

string resume;

int skills;

void setResume(string resume) {

this->resume = resume;

}

void setSkills(int skills) {

if (skills > 0) {

this->skills = skills;

}

else {

cout << "Error" << endl;

}

}

string getResume() {

return resume;

}

int getSkills() {

return skills;

}

void getInfo() {

Human::getInfo();

cout << "Resume: " << getResume() << endl

<< "Skills: " << getSkills() << endl;

}

};

class Employ : public Human {

public:

string workExperience;

int seniority;

void setWorkExperience(string workExperience) {

this->workExperience = workExperience;

}

void setSeniority(int seniority) {

if (this->workExperience == "") {

this->seniority = 0;

}

else if (seniority > 0) {

this->seniority = seniority;

}

else {

cout << "Error" << endl;

}

}

string getWorkExperience() {

return workExperience;

}

int getSeniority() {

return seniority;

}

void getInfo() {

Human::getInfo();

cout << "Work expirience: " << getWorkExperience() << endl

<< "Seniority: " << getSeniority() << endl;

}

};

int main()

{

Employ Tom;

Tom.setAge(20);

Tom.setEducation("High");

Tom.setHeigth(183);

Tom.setName("Thomas");

Tom.setWeigth(80);

Tom.setWorkExperience("Junior C++");

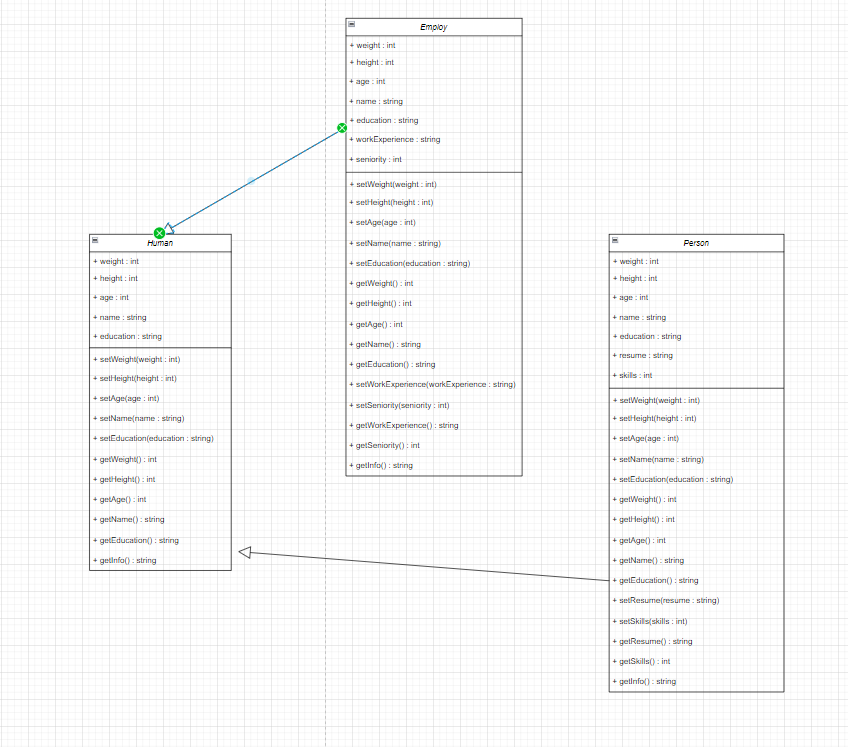
Tom.setSeniority(2);

Tom.getInfo();

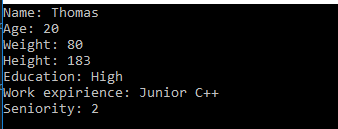
return 0;

}

# Диаграмма классов для версии 1



# Вывод для версии 1



# Код программы версия 2

#include <iostream>

#include <string>

#include <list>

using namespace std;

class Human {

public:

int weight, height, age;

string name, education;

void setWeight(int weight) {

if (weight > 0) {

this->weight = weight;

}

else {

cout << "Error" << endl;

}

}

void setHeight(int height) {

if (height > 0) {

this->height = height;

}

else {

cout << "Error" << endl;

}

}

void setAge(int age) {

if (age >= 18) {

this->age = age;

}

else {

cout << "Error" << endl;

}

}

void setName(string name) {

if (name == "") {

cout << "Error" << endl;

}

else {

this->name = name;

}

}

void setEducation(string education) {

if (education == "") {

cout << "Error" << endl;

}

else {

this->education = education;

}

}

int getWeight() {

return weight;

}

int getHeight() {

return height;

}

int getAge() {

return age;

}

string getName() {

return name;

}

string getEducation() {

return education;

}

void getInfo() {

cout << "Name: " << getName() << endl

<< "Age: " << getAge() << endl

<< "Weight: " << getWeight() << endl

<< "Height: " << getHeight() << endl

<< "Education: " << getEducation() << endl;

}

};

class Person : public Human {

public:

string resume;

int skills;

void setResume(string resume) {

this->resume = resume;

}

void setSkills(int skills) {

if (skills > 0) {

this->skills = skills;

}

else {

cout << "Error" << endl;

}

}

string getResume() {

return resume;

}

int getSkills() {

return skills;

}

void getInfo() {

Human::getInfo();

cout << "Resume: " << getResume() << endl

<< "Skills: " << getSkills() << endl;

}

};

class Employ : public Human {

public:

string workExperience;

int seniority;

Employ(Person person, string workExperience, int seniority)

{

this->age = person.getAge();

this->education = person.getEducation();

this->height = person.getHeight();

this->name = person.getName();

this->seniority = seniority;

this->weight = person.getWeight();

this->workExperience = workExperience;

}

void setWorkExperience(string workExperience) {

this->workExperience = workExperience;

}

void setSeniority(int seniority) {

if (this->workExperience == "") {

this->seniority = 0;

}

else if (seniority > 0) {

this->seniority = seniority;

}

else {

cout << "Error" << endl;

}

}

string getWorkExperience() {

return workExperience;

}

int getSeniority() {

return seniority;

}

void getInfo() {

Human::getInfo();

cout << "Work expirience: " << getWorkExperience() << endl

<< "Seniority: " << getSeniority() << endl;

}

};

list <Person> persons;

list <Person>::iterator itPer;

list <Employ> employs;

list <Employ>::iterator itEmp;

void start() {

int weight, height, age, count = 3, skills;

string name, education, resume;

for (int i = 0; i < count; i++) {

persons.push\_back(Person());

}

for (itPer = persons.begin(); itPer != persons.end(); itPer++) {

cout << "Enter your name: "; cin >> name; itPer->setName(name);

cout << "Enter your weight: "; cin >> weight; itPer->setWeight(weight);

cout << "Enter your height: "; cin >> height; itPer->setHeight(height);

cout << "Enter your age: "; cin >> age; itPer->setAge(age);

cout << "Enter your education: "; getline(cin >> ws, education); itPer->setEducation(education);

cout << "Enter your resume: "; getline(cin >> ws, resume); itPer->setResume(resume);

cout << "Enter count of your skills: "; cin >> skills; itPer->setSkills(skills);

cout << endl;

}

}

void showAllPersons() {

for (itPer = persons.begin(); itPer != persons.end(); itPer++) {

itPer->getInfo();

cout << endl;

}

}

void regist() {

string workExperience, name;;

int seniority;

cout << "Enter name, which you want to move to Employ: "; getline(cin >> ws, name);

for (itPer = persons.begin(); itPer != persons.end(); itPer++) {

if (name == itPer->getName()) {

cout << "Enter the work experience: "; getline(cin >> ws, workExperience);

cout << "Enter the seniority: "; cin >> seniority;

employs.push\_back(Employ((\*itPer), workExperience, seniority));

}

}

}

void showAllEmploys() {

for (itEmp = employs.begin(); itEmp != employs.end(); itEmp++) {

itEmp->getInfo();

cout << endl;

}

}

int main()

{

start();

showAllPersons();

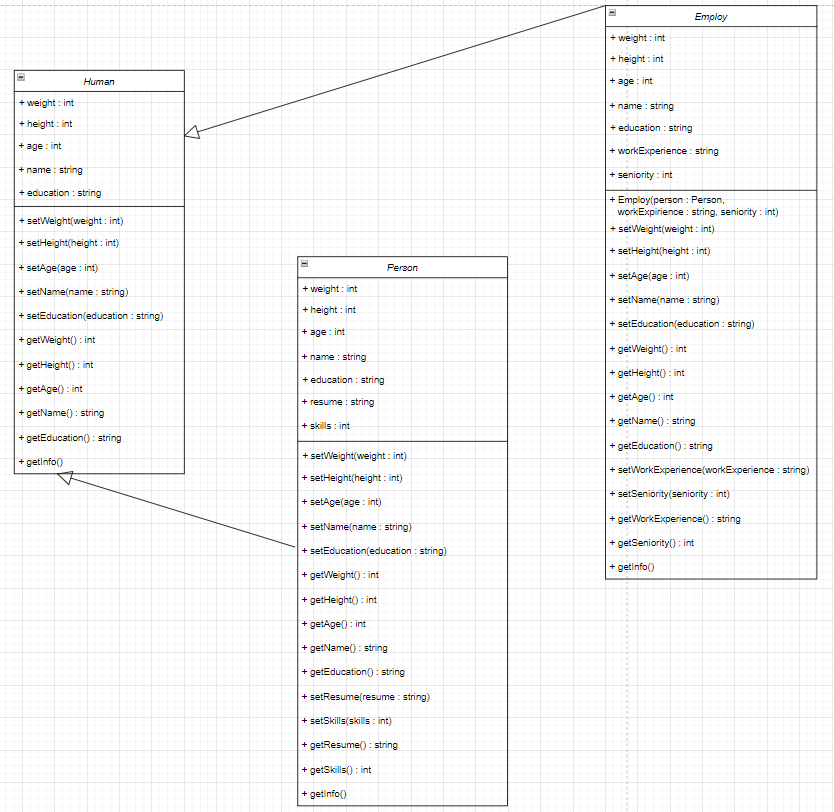
regist();

showAllEmploys();

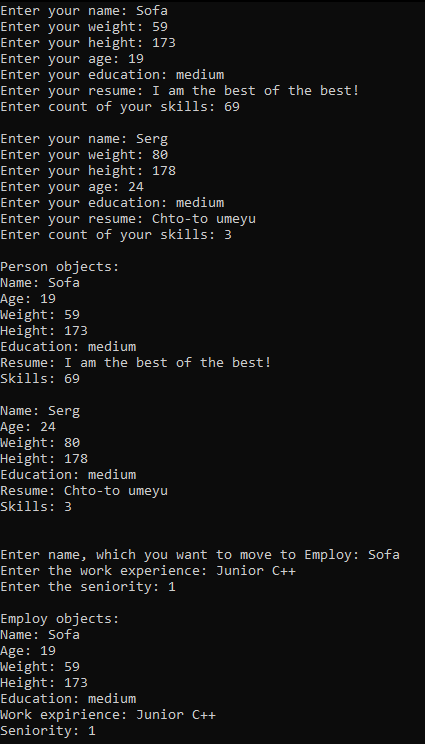
return 0;

}

# Диаграмма классов для версии 2



# Вывод программы для версии 2



Код программы для версии 3

#include <iostream>

#include <string>

#include <list>

using namespace std;

class Human {

public:

int weight, height, age;

string name, education;

void setWeight(int weight) {

if (weight > 0) {

this->weight = weight;

}

else {

cout << "Error" << endl;

}

}

void setHeight(int height) {

if (height > 0) {

this->height = height;

}

else {

cout << "Error" << endl;

}

}

void setAge(int age) {

if (age >= 18) {

this->age = age;

}

else {

cout << "Error" << endl;

}

}

void setName(string name) {

if (name == "") {

cout << "Error" << endl;

}

else {

this->name = name;

}

}

void setEducation(string education) {

if (education == "") {

cout << "Error" << endl;

}

else {

this->education = education;

}

}

int getWeight() {

return weight;

}

int getHeight() {

return height;

}

int getAge() {

return age;

}

string getName() {

return name;

}

string getEducation() {

return education;

}

void getInfo() {

cout << "1. Name: " << getName() << endl

<< "2. Age: " << getAge() << endl

<< "3. Weight: " << getWeight() << endl

<< "4. Height: " << getHeight() << endl

<< "5. Education: " << getEducation() << endl;

}

};

class Person : public Human {

public:

string resume;

int skills;

void setResume(string resume) {

this->resume = resume;

}

void setSkills(int skills) {

if (skills > 0) {

this->skills = skills;

}

else {

cout << "Error" << endl;

}

}

string getResume() {

return resume;

}

int getSkills() {

return skills;

}

void getInfo() {

Human::getInfo();

cout << "6. Resume: " << getResume() << endl

<< "7. Skills: " << getSkills() << endl;

}

};

class Employ : public Human {

public:

string workExperience;

int seniority;

Employ(Person person, string workExperience, int seniority)

{

this->age = person.getAge();

this->education = person.getEducation();

this->height = person.getHeight();

this->name = person.getName();

this->seniority = seniority;

this->weight = person.getWeight();

this->workExperience = workExperience;

}

void setWorkExperience(string workExperience) {

this->workExperience = workExperience;

}

void setSeniority(int seniority) {

if (this->workExperience == "") {

this->seniority = 0;

}

else if (seniority > 0) {

this->seniority = seniority;

}

else {

cout << "Error" << endl;

}

}

string getWorkExperience() {

return workExperience;

}

int getSeniority() {

return seniority;

}

void getInfo() {

Human::getInfo();

cout << "6. Work expirience: " << getWorkExperience() << endl

<< "7. Seniority: " << getSeniority() << endl;

}

};

list <Person> persons;

list <Person>::iterator itPer;

list <Employ> employs;

list <Employ>::iterator itEmp;

void start() {

int weight, height, age, count = 2, skills;

string name, education, resume;

for (int i = 0; i < count; i++) {

persons.push\_back(Person());

}

for (itPer = persons.begin(); itPer != persons.end(); itPer++) {

cout << "Enter your name: "; cin >> name; itPer->setName(name);

cout << "Enter your weight: "; cin >> weight; itPer->setWeight(weight);

cout << "Enter your height: "; cin >> height; itPer->setHeight(height);

cout << "Enter your age: "; cin >> age; itPer->setAge(age);

cout << "Enter your education: "; getline(cin >> ws, education); itPer->setEducation(education);

cout << "Enter your resume: "; getline(cin >> ws, resume); itPer->setResume(resume);

cout << "Enter count of your skills: "; cin >> skills; itPer->setSkills(skills);

cout << endl;

}

}

void showAllPersons() {

cout << "Person objects:" << endl;

for (itPer = persons.begin(); itPer != persons.end(); itPer++) {

itPer->getInfo();

cout << endl;

}

}

void regist() {

string workExperience, name;;

int seniority;

cout << "\nEnter name, which you want to move to Employ: "; getline(cin >> ws, name);

for (itPer = persons.begin(); itPer != persons.end(); itPer++) {

if (name == itPer->getName()) {

cout << "Enter the work experience: "; getline(cin >> ws, workExperience);

cout << "Enter the seniority: "; cin >> seniority;

employs.push\_back(Employ((\*itPer), workExperience, seniority));

cout << endl;

}

}

}

void showAllEmploys() {

cout << "Employ objects:" << endl;

for (itEmp = employs.begin(); itEmp != employs.end(); itEmp++) {

itEmp->getInfo();

cout << endl;

}

}

void editEmploy() {

bool flag = true;

int check, age, weight, height, seniority;

string name, workExperience, education;

cout << "Do you want to edit imformation about Employ?" << endl

<< "1 - YES, 2 - NO" << endl; cin >> check;

if (check == 1) {

cout << "Enter the name of employ: "; getline(cin >> ws, name);

for (itEmp = employs.begin(); itEmp != employs.end(); itEmp++) {

if (name == itEmp->getName()) {

itEmp->getInfo();

cout << "What parameter do you want to edit?" << endl; cin >> check;

switch (check)

{

case 1:

cout << "Current value: " << itEmp->getName() << endl;

while (flag) {

cout << "Enter new value: "; getline(cin >> ws, name);

if (name == itEmp->getName()) {

cout << "It is the previous value! Try again!" << endl;

}

else {

itEmp->setName(name);

cout << "Succesfully!" << endl;

flag = false;

}

}

break;

case 2:

cout << "Current value: " << itEmp->getAge() << endl;

while (flag) {

cout << "Enter new value: "; cin >> age;

if (age == itEmp->getAge()) {

cout << "It is the previous value! Try again!" << endl;

}

else {

itEmp->setAge(age);

cout << "Succesfully!" << endl;

flag = false;

}

}

break;

case 3:

cout << "Current value: " << itEmp->getWeight() << endl;

while (flag) {

cout << "Enter new value: "; cin >> weight;

if (weight == itEmp->getWeight()) {

cout << "It is the previous value! Try again!" << endl;

}

else {

itEmp->setWeight(weight);

cout << "Succesfully!" << endl;

flag = false;

}

}

break;

case 4:

cout << "Current value: " << itEmp->getHeight() << endl;

while (flag) {

cout << "Enter new value: "; cin >> height;

if (height == itEmp->getHeight()) {

cout << "It is the previous value! Try again!" << endl;

}

else {

itEmp->setHeight(height);

cout << "Succesfully!" << endl;

flag = false;

}

}

break;

case 5:

cout << "Current value: " << itEmp->getEducation() << endl;

while (flag) {

cout << "Enter new value: "; getline(cin >> ws, education);

if (education == itEmp->getEducation()) {

cout << "It is the previous value! Try again!" << endl;

}

else {

itEmp->setEducation(education);

cout << "Succesfully!" << endl;

flag = false;

}

}

break;

case 6:

cout << "Current value: " << itEmp->getWorkExperience() << endl;

while (flag) {

cout << "Enter new value: "; getline(cin >> ws, workExperience);

if (workExperience == itEmp->getWorkExperience()) {

cout << "It is the previous value! Try again!" << endl;

}

else {

itEmp->setWorkExperience(workExperience);

cout << "Succesfully!" << endl;

flag = false;

}

}

break;

case 7:

cout << "Current value: " << itEmp->getSeniority() << endl;

while (flag) {

cout << "Enter new value: "; cin >> seniority;

if (seniority == itEmp->getSeniority()) {

cout << "It is the previous value! Try again!" << endl;

}

else {

itEmp->setSeniority(seniority);

cout << "Succesfully!" << endl;

flag = false;

}

}

break;

default:

break;

}

}

cout << endl;

itEmp->getInfo();

}

}

else {

cout << "Goodbye! :)";

}

}

int main()

{

start();

showAllPersons();

regist();

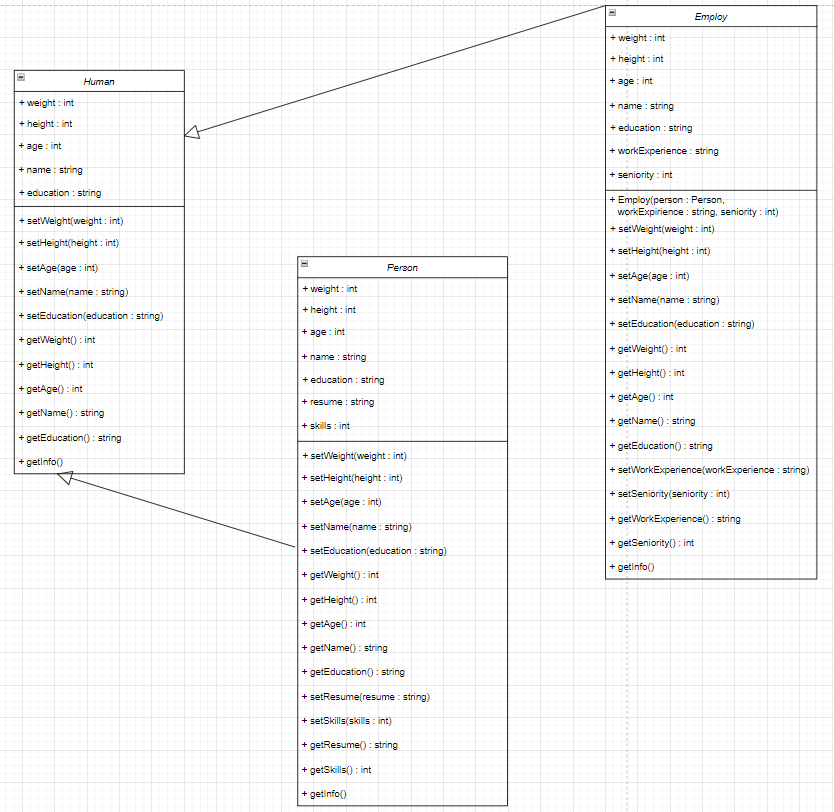
showAllEmploys();

editEmploy();

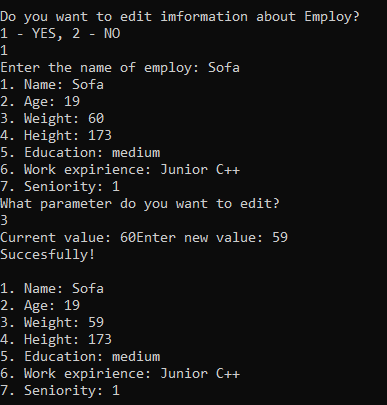
return 0;

}

Диаграмма классов для версии 3



Вывод для версии 3



# Вывод

На данной практической работе мы создали класс Human, а также два дочерних класса Person и Employ. Реализовали регистрацию человека для поиска работы и передачу его данных и их дополнения при устройстве на работу. В отдельных методах предоставили возможность пользователю изменить какие-либо данные.