# Министерство науки и высшего образования Российской Федерации

Пензенский Государственный университет Кафедра "Вычислительная техника"

### Пояснительная записка

К курсовому проектированию По курсу «Логика и основы алгоритмизации в инженерных задачах » На тему: «Реализация алгоритма Прима»

Выполнил студент группы 22ВВВЗ:

Кузьмин Д.В.

Принял:

Юрова О.В.

Акифьев И.В.

ЗАДАНИЕ  на курсовое проектирование по курсу  Мунка и основа дломитимующий винтенцика зоная Студенту Унульнику ванину выгодицьвый у Группа 22 вряз (игру  Тема_проекта Реализиция осноритьма Пунка  Исходные данные (технические требования) на проектирование возразотка амаритыва и гром решинию общения	Кафелра 5	"Вычислительн	ая техника"		
3ADAHUE  HA KYPCOBOR PROCESTUPOBAHUE NO KYPCY  LIMING U OCH OBST GLOMMANIAMINE BUNNEMUM ZOGOMA  CTYZEHTY LIMINAMY EAMINY BINIGHTIOGOMY PPYNNA 22 BBBB (2288)  TEMA_ ПРОЕКТА RAMINAMINE OMNOFILITMA PRIMICA  ИСХОДНЫЕ ДАННЫЕ (ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ) НА ПРОЕКТИРОВАНИЕ  КОМПЬТКИ ОМПОТИТЬНОЕ И РОМ НЕМИНИТЕ ОБЛИСИНИЯ  СООТВЕТСТВИИ С ОМПИТЬНОЕ И РОМ НЕМИНИТЬНОГО ТОВЛЕНИЯ  СООТВЕТСТВИИ С ОМПОТИТЬНОЕ ВЫ ПОКА (ОЗЕДИТЕЛЬНОГО  Т. ПОСТОИЛОВКУ ЗАЗАНИЯ;  2. ТЕОРЕТИЧЕТИЗНО РОМОТИТЬНОГО ЗАЗАНИЯ;  МИТЕМИТЬНОГО ПОКЛОТИТЬНОГО  В ПОТИКИ ИПЕДАТИРИЯ;  8. ПИТИКИ ИПЕДАТИРИЯ;  8. ПИТИКИ ИПЕДАТИРИЯ;  8. ПИТИКИ ИПЕДАТИРИЯ;  9. ПИТИКИ ИПЕДАТИРИЯ;  9. ПИТИКИ ИПЕДАТИРИЯ;  9. ПИТИКИ ИПЕДАТИРИЯ;	Кифедри	BBI INC.III COIDI			рж па
«		200	wahawa ŏ D	/	-
Ha Kypcoboe проектирование по курсу  Момка и основое умученирование по курсу  Прина 22 вряз (2288  Тема проекта вализация от прина Тупиа  Исходные данные (технические требования) на проектирование  воргаботка амуритмов и траничения обтечний собъетствии с одиним задоснием тупового наекта  Тохниченных записка уминия;  Тостосновки задосний;  Тостосновки задосний;  Тостосновки задосний;  Торетическую часть задосний;  Торетический насть задосний насты задос		Зав.	кафедрои В		
на курсовое проектирование по курсу  Домка и основа длориту вину вини вини вини вини видина зада в в в в в в в в в в в в в в в в в в			<u> </u>	<u> </u>	20
Походные данные (технические требования) на проектирование  воднаютья америка замину выстроина обетирование  воднаютья америка заминия заминия тупина  Остовенты с данный заминий тупина обетичений собъетьым с данный заминый тупового учекта Похомиченьных замина домпена содержать;  1. Постоиновку задамий;  2. Порежинескую часть дадамий;  3. Описание америка пожавенный задамий;  4. Пример учестве задамий;  5. Описание самы наружный;  5. Описание самы наружный;  6. Тести;  7. Стом меранурамий;  8. Миник митературы;  8. Миник митературы;  8. Миник митературы;  8. Миник митературы;  9. Везумьтать наботи поменамия;		ЗАДАНИЕ			
Тема_проекта рамину вы выросия вы проектирование  Исходные данные (технические требования) на проектирование  водразотка амаритывый и трания проектирование  водразотка амаритывый заданивые турового проекта  Тохнительных записка облиска содержать:  1. Постоповку задания;  2. Теоретическую часть дадания;  3. Отисание амариты поставленный задани;  4. Причен ручного рассета задани и вычисиений (та небымини участи участи амариты»;  5. Отисание саный программик;  6. Тесто;  9. Реушьтата паботи тиотамина;	на курсово	е проектирова	ние по курс	ey	
Исходные данные (технические требования) на проектирование  водработка амаритмов и тром ребомного обетечний соответствии с данным заданием турноваго троекта  Похинтемьнах записка домнога содержать:  1. Постоповку задания;  2. Коретическую часть даданий;  3. Описание амаритма поставленной задани;  4. Пример учного рассета задани вычислений (та небымим частке работи амаритма);  5. Описание саной программия;  6. Тесто;  4. (ток меранута;  9. Ганных программи;  9. Ганных паботи постания:	CTVIEHTV MASS MAINE SOLVILLE	A low our INO	KINNE T	и <i>менерии</i> руппа 22 в	x 3019010 383 (2288
LOPRESTRO ANDJUTHLOS U PROPREMIENTO OSENETERUS.  CONSETERSUM C COMMUNICA SOLVENIA TOJUSTO PROCESTO  TORNINGENSHOUX ZAMURIE OSINCHA COJEJIMONTO:  1. TLOCTOMOSKY ZAGANUR;  2. TEODETURERAJAS VACTO JAGANUR;  3. IMILAN MANONITHA PORTASIENHOÑ ZAGARU;  Y. TIMILA MANONITHA PARTASIENHOÑ ZAGARU;  Y. TIMILA MANONITHA PROPREMIEN;  5. OMICANIE CANON HADYNAMON;  6. TECTO;  9. PAYMOTAN MASOTA MONTANIA, RIP;	Tema_npoekta_tea_tugaty_0.90	unojuuma	: Huma		
LOPRESOTRO anoputhlos u propresentamento ocenerenas.  CONSETETSUM C COMMUNICA SOLVENIA TOJUSTO PROCESTO  TORNINGENSHOUX ZAMURIE OSINCHA COJEJIMONTO:  1. TLOCTOMOSKY ZAGANURI;  2. TEODETWEEKYJE TACTE JAGANURI;  3. IMILAN MYNONO PRECETO ZAGARU U SURVELLENIA (TE  METALINAM TRANSPORTA MASOTIN ANOPUTHA);  5. OMICANIE CANON HAMPIANOWI;  6. TECTOS;  4. IMILAN MEDATYMA;  8. MUTUM, RADINAMURI;  9. PAYMOTAN MASOTIL MONTANIA.					
CONSETERSUM C CAMMON ZAGAMULLY TYMOSOND TYMERTA TLOCKMITENSHAX ZAGAMUZ; 1. TLOCKMOSKY ZAGAMUZ; 2. KOPETWEEKYSE VACTE GAGAMUZ; 3. MILAMUE ANOMITWA MOTAGUENNOW ZAGAMU; 4. TYMEN MURIONO MACKETA ZAGAMU W BONTUCHENWE(K KESONDUM YAGUTKE MADOTIK ANOMITWA); 5. PRICAMUS CANON HOMPAMON; 6. TECTOS; 4. (MICOK MIERATYMI); 8. MITUMI, ROPOPAMUN; 9. PAYMOTAN MADOTIC MONTAGUN;	Исходные данные (техн	ические требов	ания) на про	ектирован	ие
Coorbete Tour ( adknish zagarille Typicoloro typick To Tlose kuresh kase zagarilis; 1. Tocromobky zagarilis; 2. Tropeturel kyst racto zagarilis; 3. Indianile alromatica hat aliennoù zagaril; 4. Tymilen yurioro parceta zagaril y borrucherini (he hesostaniar grante pasotre adropativa); 5. Ormanile caroù reporpanour; 6. Tectos; 4. (mucok surepanyur; 3. surun, reporpanur; 3. surun; 3. surun, reporpanur; 3. surun, repor	Commer Cotto or MIND MITTELL	0			
1. Thochambery zaganus;  1. Thochambery zaganus;  2. Kopetwelky be valle gaganus;  3. Ameanul adromatica hostaliennoù zagaru;  4. Tymilen munos marceta zagaru u borunteruni(ta hebantuur franke pravotro antoputua);  5. Ameanule Caron hyporramour;  6. Terro;  7. Comok sutenatyma;  8. surum, aporuansur;  9. leyestata maronasure;		of u homan	d ullum	5. male	011119
3. Intrapul adoputiva hottaliennoù zagarn; 4. Aprille pydroro parreta zagarn y bornelenni(h hedonouen grante poppanour; 5. Armanie caroù rporpanour; 6. Tector; 4. (mucok surepoppanour; 8. Suren, proppanour; 9. Pegulotata madota mormaneze;	CONSETUTION C COMMUNICATION OF THE CONTRACTION OF T	of u prom	Here Energy	odener	exun,
3. Intrapul adoputiva hattaliennoù zagarn; 4. Aprille pydroro parceta zagarn y bornelenni (ri resonouest grante pasotro adroputiva); 5. Orneanne caroù reporpanous; 6. Tectos; 4. (mucok surepanyer; 8. surun reporpansur; 9. legulotata masorie mornanez.	Corbeterbun C galler	of u rpoon	ulve try	wolono ty	exul,
3. Intrapul adoputiva hattaliennoù zagarn; 4. Aprille pydroro parceta zagarn y borruchereun (ra redontumer grante parotro adroputiva); 5. Ormanie caroù reporpanour; 6. Tector; 4. (mucok surepanymo; 8. surtuen aporpansur; 9. legulotata madore morragune;	Coorbeterbun C garris Torrumenbrux zami 1. Tocromobry znogra	of u rparp we zagoines us:	leve try	odener noboro ty mart:	exuit.
4. America proportion sugare y borructerini (resolution grante proportion); 5. Orneasure caron hopogranous; 6. Textos; 7. Smuok sutenatyms; 8. Surrum proportions; 9. Peyulotato madore morranges;	CONSETUTEMENT ZAMENTE ZAMENTENSKY ZAMENTENSKY ZAGANI 1. TOCTOMOSKY ZAGANI 2. TEONETWIECKY 18 YOUT	ur zagoni ur golnes ur;	Mery Try	osener wolon ty marti:	exul.
Hedonbulst grante proportion;  5. Ormanic Caron reportation;  6. Tector;  4. (mucor sureparyme;  8. surrum aporparism;  9. legylotata madota mormanica;	COOTSETCTSUU C GAMV. TOCKMITENSKY ZOGANO 2. TODETWECKYSE YOCT 3. ONWERNEL SINOWENCO	ur zagoni ur golnes ur;	ca cogep	month:	etuit, wekta
8. Worum promaneur; 9. Regulatation madatre mornageneri	COOTSETCTSUU C GAMU TORMUTENSHUX ZAGAMU 1. TOCTOMOSKY ZAGAMU 2. KODETWEEKYNS YACT 3. MILANNE ANOMITMA 4. TOMILLON MANONO M	we zalgali whi; b gaganus hotabien	ca cogen i; non zag	matt:	
8. Lucture propriedur; 9. Regulotato madore mornadures.	COOTSETCTSUU C GONNU TOANUTENSHUX ZOGANI 1. TOCTOMOSKY ZOGANI 2. TROMETURESKYSS YOUT 3. PRINTERNE ON OPUTICA 4. THUSLEN MUNICOM HEGOLMUNH ON OUTRE	we zalgali whi; b gaganus hotabien	ca cogen i; non zag	matt:	
8. Worner properties; 3. Regulotato madore mornagenes;	COOTSETCTSUU C QUEEN TOUR MUTENSHUX ZUNNI 1. TOCTOMOSKY ZUGENI 2. TEODETWEEKYJE VACT 3. AMCANUE STROMOTOM Y. TIMBER MYCHOROM HELDINGER PROTECTION OF	we zalgali whi; b gaganus hotabien	ca cogen i; non zag	matt:	
8. Lucture propriedur; 3. Regulotato madore mornadures;	COOTSETCTSUU C GARRY TORRHUTENSKY ZAGARY 1. TOCTOMOSKY ZAGARY 2. TEORETURECKYSE RACT 3. PROCESURE ANOMETICA Y. TYMPLEN MYCHORO M RESILVUM GRANDE CARON RACT 6. TENTO:	we zalgali whi; b gaganus hotabien	ca cogen i; non zag	matt:	
	Cobrbeterbuu C galling. Toxemirenbuux zami 1. Toxemirenbuux zagana 2. Teoperweekkyrb ract 3. Amerika diromitika 4. Tymben myenoro m ketorbuur gractike m 5. Ormanne caron hyor 6. Terro;	we zalgali whi; b gaganus hotabien	ca cogen i; non zag	matt:	
	Cobrbeterbuu C galling. Toxemirenbuux zami 1. Toxemirenbuux zagana 2. Teoperweekkyrb ract 3. Amerika diromitika 4. Tymben myenoro m ketorbuur gractike m 5. Ormanne caron hyor 6. Terro;	we zalgali whi; b gaganus hotablen	ca cogen i; non zag	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Koperwelky brack 3. Antranul alromativa 4. Tymbel ynghoro yn ketonburg gracke yn 5. Ormanue caron hyno 6. Terro; 4. Comok Miteranyne; 8. Jurun prophymus	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteratypus; 8. Jurtum prophanium 9. Peyulotata nabotu m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteratypus; 8. Jurtum prophanium 9. Peyulotata nabotu m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteratypus; 8. Jurtum prophanium 9. Peyulotata nabotu m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteratypus; 8. Jurtum prophanium 9. Peyulotata nabotu m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteratypus; 8. Jurtum prophanium 9. Peyulotata nabotu m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteratypus; 8. Jurtum prophanium 9. Peyulotata nabotu m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteratypus; 8. Jurtum prophanium 9. Peyulotata nabotu m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteratypus; 8. Jurtum prophanium 9. Peyulotata nabotu m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteratypus; 8. Jurtum prophanium 9. Peyulotata nabotu m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun C galling. Thoromoby zagana 1. Thoromoby zagana 2. Roperwelky by ract 3. Indianul alromativa 4. Tymbler yulnoro yu kelonbur yulnoro yu kelonbur yulnoro yu 5. Onucasule caron hoo 6. Teero; 7. Comoon unepanyme; 8. Jurum propanyme; 9. Peyydorana naboru m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun C galling. Thoromoby zagana 1. Thoromoby zagana 2. Roperwelky by ract 3. Indianul alromativa 4. Tymbler yulnoro yu kelonbur yulnoro yu kelonbur yulnoro yu 5. Onucasule caron hoo 6. Teero; 7. Comoon unepanyme; 8. Jurum propanyme; 9. Peyydorana naboru m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	
	Coorbeterbun ( galling. Tlockwisenbung zagana 1. Tockwisenbung zagana 2. Ropetwelky by ract 3. Andranul alromativa 4. Tymbler yulkoro yu kelonbung yulkoro yu kelonbung yulkoro yu 5. Ormanul caron haor 6. Tector; 7. Common unteranyor; 8. Jurum propaganun 9. Pegulotaka nasoru m	us zolganis us ; b zaganus hotrabien aceta za asorra dus	ca coglyn i ; noù z ag garli u b opurva)	matt:	

Объем работы по курсу 1. Расчетная часть	
Pyrnon pacret pasor	to alloputila
2. Enghymagrag magri	
2. Графическая часть	Producto Slaverony
Dara amojnema.	bogneste flok-crese.
3. Экспериментальная ч	асть
3. Экспериментальная ч Теххиповолния тром	ANGHAMAR:
Terxunohamue, nnon	
Terxunohamue, nnon	ANGHAR:
Terripolarne morpi Egyddiarri grasorie	Mogradiur ka Tectober gammax
Terrupobarrie ryorp Egyntrarri grasorie Cpok Bb 1 Ucclerobarnie Teonez	муник ва Тестових данних  муник ва Тестових данних  мполнения проекта по разделам  тих ва части ким овто
Textupobarne ryorp Egyptotatre grasore Cpok Bb 1 Ucalegobarne Teonez 2 Eagn ordornoe au no	мунуник катильних данних  мининия проекта по разделам  миникой части кунсового  митиве проможения
Cpok Bb  1 Ucalenobarne Report  2 Raynordothor auro  3 Laynordothor mary	муник ва Тестових данних минирамин ка Тестових данних мполнения проекта по разделам миской части курсового учити в програминя
Cpok Bb  1 Ucalenobarrue Te one:  2 Eagnord That also 4 Telrupobarrue U 3	муничит катильних данних  минина проекта по разделам  минина проекта по разделам  минина проекта по разделам
Cpok Bb  1 Ucalendamie Reper 2 Raynordotha auro 3 Laynordotha auro 4 Telrupobamie u z 5 Dognine naic 6 7	мунунамин ка Титових данних  молнения проекта по разделам  тической части курсового  уштиов краннамина  авермение мографотки пранамина
Cpok Bb  1 Ucalenobarrue Te one:  2 Eagnord That also 4 Telrupobarrue U 3	мунунамин ка Титових данних  молнения проекта по разделам  тической части курсового  уштиов краннамина  авермение мографотки пранамина
Cpok Bb  1 Ucalendamie Reper 2 Raynordotha auro 3 Laynordotha auro 4 Telrupobamie u z 5 Dognine naic 6 7	мунунамин ка Титових данних  молнения проекта по разделам  тической части курсового  уштиов краннамина  авермение мографотки пранамина
Cpok Bb  1 Ucalendamie Reper 2 Raynordotha auro 3 Laynordotha auro 4 Telrupobamie u z 5 Dognine naic 6 7	программи катильних данних  полнения проекта по разделам  пиской части купсового  питнов программич  авения  питемной записки
Cpok Bb  1 Ucalendamie Reper 2 Raynordotha auro 3 Laynordotha auro 4 Telrupobamie u z 5 Dognine naic 6 7	пропрамент ка техових данних  мистов части курсового  питиов пропрамения  аверине погработки прираментической записки  дата выдачи задания "6" сентября

## ОГЛАВЛЕНИЕ

Реферат	5
Введение	6
1.Постановка задачи	7
2.Теоретическая часть	8
3.Описание алгоритма программы	9
4.Описание программы	11
5.Тестирование	15
6.Ручной расчёт программы	18
Заключение	19
Список литературы	20
7.Приложение А	21
Листинг программы	21

# Реферат

Отчет 23 стр., 12 рисунков, 1 таблица, 1 приложение. ГРАФ, ТЕОРИЯ ГРАФОВ, АЛГОРИТМ ПРИМА

Цель исследования – реализация алгоритма Прима для нахождения минимального остовного дерева.

В работе рассмотрены основные шаги алгоритма Прима для построения минимального остовного дерева взвешенного графа. Установлено, что с помощью данного алгоритма можно выделить минимальное остовное дерево независимо от степени связности и весов рёбер исходного графа.

#### Введение

Алгоритм Прима - это алгоритм минимального остовного дерева, что принимает граф в качестве входных данных и находит подмножество ребер этого графа, который формирует дерево, включающее в себя каждую вершину, а также имеет минимальную сумму весов среди всех деревьев, которые могут быть сформированы из графа.

Алгоритм Прима строит минимальное остовное дерево, добавляя на каждом шаге к строящемуся остову безопасное ребро минимальной длинны.

Ребро называется безопасным, если при добавлении его к строящемуся остову не нарушается свойство ацикличности.

В качестве среды разработки мною была выбрана MicrosoftVisualStudio2019, язык программирования – Си.

Целью данной курсовой работы является разработка программы на языке Си, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется алгоритм Прима, осуществляющий нахождение минимального остовного дерева.

#### 1. Постановка задачи

Требуется разработать программу, которая выделит минимальное остовное дерево для неориентированного взвешенного графа, используя алгоритм Прима.

Исходный граф в программе должен задаваться матрицей весов рёбер, причем при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы взвешенного графа. После обработки этих данных на экран должна выводиться матрица взвешенного графа и минимальное остовное дерево, полученное алгоритмом Прима. Необходимо предусмотреть различные исходы, такие как случаи отсутствия связности в графе или некорректного ввода данных, чтобы программа не выдавала ошибок и работала правильно. Устройство ввода — клавиатура и мышь.

#### 2. Теоретическая часть задания

Граф G (рисунок 1) задается множеством вершин  $X_1, X_2, ..., X_n$ и множеством ребер, соединяющих между собой определенные вершины. Ребра в этом графе имеют веса, показывающие стоимость прохождения между вершинами. Граф с такими ребрами называется взвешенным.

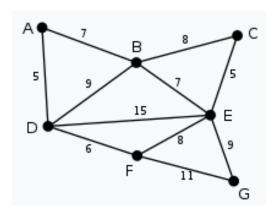


Рисунок 1 – Пример взвешенного графа

При представлении графа матрицей смежности информация о ребрах графа хранится в квадратной матрице, где пути из одной вершины в другую и обратно обозначаются единицей, иначе нулем.

На вход алгоритма подаётся связный неориентированный граф. Для каждого ребра задаётся его стоимость.

Сначала берётся произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются рёбра графа, один конец которых — уже принадлежащая дереву вершина, а другой — нет; из этих рёбер выбирается ребро наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа.

Результатом работы алгоритма является остовное дерево минимальной стоимости.

### 3. Описание алгоритма программы

Цель алгоритма Прима графе В заключается в построении минимального остовного дерева путем пошагового добавления ребер с наименьшим весом. Начиная с начальной вершины, мы постепенно расширяем остовное дерево, выбирая ребро минимального веса, которое связывает уже выбранные вершины с вершиной из оставшихся. Этот процесс продолжается, пока все вершины не будут включены в остовное дерево или пока не будет достигнут критерий остановки. Ниже представлен псевдокод: функция Алгоритм Прима(граф[MAX\_SIZE][MAX\_SIZE], размер,

```
родитель[MAX_SIZE])
      минВес[MAX SIZE]
      посещено[MAX SIZE]
      для і = 0 пока і < размер
      минВес[i] = INT MAX
      посещено[i] = false
      родитель[і] = -1
      минBec[0] = 0
      для count = 0 пока count < размер - 1
      минВершина = -1
      для v = 0 пока v < размер
      если не посещено[v] и (минВершина == -1 или
минBec[v] < минBec[минBeршина])
      минВершина = v
      посещено[минВершина] = true
      для v = 0 пока v < размер
```

если граф[минВершина][v] != 0 и не посещено[v] и

```
граф[минВершина][v] < минВес[v]

родитель[v] = минВершина

минВес[v] = граф[минВершина][v]

вывод("\nМинимальное остовное дерево:\n")

для i = 1 пока i < размер

если родитель[i] != -1

вывод("Ребро: " + родитель[i] + " - " + i + ", Вес:
" + граф[i][родитель[i]])

иначе

вывод("Ребро: " + findUnconnectedNode(родитель, размер) + " - " + i + ", не существует")
```

### 4. Описание программы

Для написания данной программы использован язык программирования Си. Язык программирования Си – универсальный язык программирования, который завоевал особую популярность программистов, благодаря возможностей сочетанию языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения Win32(VisualC++).

Данная программа многомодульная, поскольку состоит из нескольких функций: saveGraphToFile, savePrimResultToFile, fileExists. loadGraphFromFile, fillUndirectedGraphRandom, printGraph, findUnconnectedNode, primAlgorithm, main(void), struct Node, void print\_tree.

Работа программы начинается с вывода общей информации о курсовой работе(рис.2).

```
printf("Курсовая работа\n");
printf("Выполнил: Кузьмин Данила 22ВВВЗ\n");
Курсовая работа
Выполнил: Кузьмин Данила 22ВВВЗ
```

Рисунок 2 – Меню.

Далее выводятся основные действия программы, которые пользователь может выбрать (рис. 3).

```
ргіптf("Меню:\n");
ргіптf("1. Загрузить неориентированный граф из файла\n");
ргіптf("2. Заполнить неориентированный граф автоматически\n");
ргіптf("3. Завершить и выйти из программы\n");
ргіптf("Выберите номер от 1-3: ");

Меню:

1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф автоматически
3. Завершить и выйти из программы
Выберите номер от 1-3: __
```

Рисунок 3 – Основные действия

Если пользователь выбрал первый пункт (рис. 4), то сначала программа просит ввести название файла. Затем она загружает этот файл (вызывая функцию loadGraphFromFile), выводит граф на экран (вызывая функцию printGraph), выполняет алгоритм Прима и выводит минимальное остовное дерево на экран (вызывая функцию primAlgorithm), сохраняет граф и

case 1:

```
printf("\nВведите имя файла для загрузки: ");
scanf("%s", filename);
if (!fileExists(filename)) {
        printf("\nВведён несуществующий файл\n\n");
loadGraphFromFile(graph, &size, filename);
printGraph(graph, size);
for (int i = 0; i < size; i++) {
         for (int j = i; j < size; j++) {
                  D = graph[i][j];
                  if (D == 0) continue;
                  root = CreateTree(root, D);
print_tree(root, 0);
root = NULL;
D = 0;
primAlgorithm(graph, size, parent);
saveGraphToFile(graph, size, "G.txt");
savePrimResultToFile(parent, graph, size, "result.txt");
break;
1. Загрузить неориентированный граф из файла
    Заполнить неориентированный граф автоматически
   Завершить и выйти из программы
Выберите номер от 1-3: 1
Введите имя файла для загрузки: G.txt
               97
           0
               15
```

Рисунок 4 – Загрузка графа из файла

Если пользователь выбрал второй пункт (рис. 5), то сначала программа просит ввести размер графа. Затем она заполняет этот граф случайными числами (вызывая функцию fillUndirectedGraphRandom), если вводите число. Далее выводит граф на экран (вызывая функцию printGraph), если вы вводите

Минимальное остовное дерево:

Граф успешно сохранен в файл G.txt

Результат успешно сохранен в файл result.txt

Ребро: 0 - 1, Вес: 39 Ребро: 1 - 2, Вес: 13 Ребро: 2 - 3, Вес: 15 буквы, то будет выдавать ошибку. После выполняет алгоритм Прима и выводит минимальное остовное дерево на экран (вызывая функцию primAlgorithm), сохраняет граф и результат в файлы (вызывая функции saveGraphToFile и savePrimResultToFile).

```
case 2:
         printf("\nВведите размер неориентированного графа: ");
         while (1) {
                  cin >> buffer;
                  if (atoi(buffer) == 0 && buffer[0] != '0') {
                           cout <<"Введите только целое число"<< endl;
                           continue;
                  else {
                           size = atoi(buffer);
                           break:
                           }
                           // scanf("%d", &size);
         if (size \ll 0) 
                  printf("\nВведён неверный размер графа\n\n");
         fillUndirectedGraphRandom(graph, size);
         printGraph(graph, size);
                  for (int i = 0; i < size; i++) {
                           for (int j = i; j < size; j++) {
                                     D = graph[i][j];
                                     if (D == 0) continue;
                                     root = CreateTree(root, D);
         print_tree(root, 0);
         root = NULL;
         D = 0;
         primAlgorithm(graph, size, parent);
         saveGraphToFile(graph, size, "G.txt");
         savePrimResultToFile(parent, graph, size, "result.txt");
         break;
```

```
Курсовая работа
Выполнил: Кузьмин Данила 228ВВЗ
Меню:

1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф автоматически
3. Завершить и выйти из программы
Выберите номер от 1-3: 2

Введите размер неориентированного графа: 4

Граф:

0 39 0 77

39 0 13 0 15

77 97 15 0

97

77

39

15

13

Минимальное остовное дерево:
Ребро: 0 - 1, Вес: 39
Ребро: 2 - 3, Вес: 15

Граф успешно сохранен в файл G.txt

Результат успешно сохранен в файл result.txt
```

Рисунок 5 – Заполнение графа автоматически

Если пользователь выбрал пункт «3», то программа завершит свою работу (рис. 6).

```
Курсовая работа
Выполнил: Кузьмин Данила 228883
Меню:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф автоматически
3. Завершить и выйти из программы
Выберите номер от 1-3: 3

E:\C++\cursovya\x64\Debug\cursovya.exe (процесс 16164) завершил работу с кодом 3.
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 6 – Проверка на завершение программы

#### 5. Тестирование

Среда разработки MicrosoftVisualStudio 2019представляет все средства, необходимые при разработке и отладки многомодульных программ.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество ошибок и проблем, связанных с вводом данных, изменением дизайна вводимых данных, алгоритмом программы, взаимодействием функций.

Ниже продемонстрирован результат тестирования программы при вводе пользователем различных количеств вершин.

```
Курсовая работа
Выполнил: Кузьмин Данила 22ВВВЗ
Меню:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф автоматически
3. Завершить и выйти из программы
Выберите номер от 1-3: _
```

Рисунок 7 – Проверка программы на запуск

```
Курсовая работа
Выполнил: Кузьмин Данила 22ВВВЗ
Меню:

1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф автоматически
3. Завершить и выйти из программы
Выберите номер от 1-3: 4

Сделан неверный выбор. Пожалуйста, выберите снова.
Меню:

1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф автоматически
3. Завершить и выйти из программы
Выберите номер от 1-3: __
```

Рисунок 8-Проверка на неправильный выбор номер

```
Курсовая работа
Выполнил: Кузьмин Данила 22ВВВЗ
Меню:

1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф автоматически
3. Завершить и выйти из программы
Выберите номер от 1-3: 1

Введите имя файла для загрузки: dada

Введён несуществующий файл

Меню:

1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф автоматически
3. Завершить и выйти из программы
Выберите номер от 1-3:
```

Рисунок 9-Проверка на несуществующий файл

```
Меню:

1. Загрузить неориентированный граф из файла

2. Заполнить неориентированный граф автоматически

3. Завершить и выйти из программы
Выберите номер от 1-3: 2

Введите размер неориентированного графа: -3

Введён неверный размер графа

Меню:

1. Загрузить неориентированный граф из файла

2. Заполнить неориентированный граф автоматически

3. Завершить и выйти из программы
Выберите номер от 1-3: ___
```

Рисунок 10-Проверка на введение правильного размера графа

```
Меню:
1. Загрузить неориентированный граф из файла
2. Заполнить неориентированный граф автоматически
3. Завершить и выйти из программы
Выберите номер от 1-3: 2
Введите размер неориентированного графа: караван
Введите только целое число
```

Рисунок 11-Проверка на введение только целых чисел

Описание теста	Ожидаемый результат	Полученный
		результат
Запуск программы	Вывод общей информации и основных действий	Верно
Проверка на	Программа должна оповестить	
неправильный выбор	пользователя об ошибке,	Верно
основного действия	предложить снова варианты	
Проверка на введение несуществующего файла	Программа должна оповестить пользователя об ошибке, предложить снова варианты	Верно
Проверка на введение неправильного размера графа	Программа должна оповестить пользователя об ошибке, предложить снова варианты	Верно
Вывод результата	Вывод правильного результата на разно-размерных графах, идентичность с ручным расчетом	Верно
Правильность работы	Совпадение ручных расчетов с	Верно
алгоритма	результатом работы алгоритма	1
Проверка на наличие	Должна выполняться проверка	
изолированных	на наличие изолированных	Верно
вершин	вершин	

Таблица 1 – Описание поведения программы при тестировании

В результате тестирования было выявлено, что программа успешно проверяет данные на соответствие необходимым требованиям.

## 6. Ручной расчёт программы

```
Граф:

0 30 32 97 61

30 0 66 89 53

32 66 0 0 20

97 89 0 0 97

61 53 20 97 0

Минимальное остовное дерево:
Ребро: 0 - 1, Вес: 30

Ребро: 1 - 3, Вес: 89

Ребро: 2 - 4, Вес: 20
```

Рисунок 12 – Граф и результат

Начинаем с вершины 0. Выбираем ребро с минимальным весом: ребро 0 - 1 с весом 30. Далее ищем следующие рёбра с минимальным весом по такому же принципу. Из вершины 0 можно пройти в вершину 2 и получить следующее ребро: ребро 0 - 2 с весом 32. Из вершины 2 можно перейти в вершину 4, получаем ребро: ребро 2 - 4 с весом 20. Из вершины 1 можно перейти в вершину 3, получаем ребро: ребро 1 - 3 с весом 89. В итоге мы получаем минимальное остовное дерево: 3 - 1 - 0 - 2 - 4. Таким образом, результат ручных расчетов совпадает с результатом работы алгоритма, таким образом можно сделать вывод, что программа работает верно.

#### Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Прима в MicrosoftVisualStudio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежности, а также работы с новыми алгоритмами. Углублены навыки знания языка программирования Си.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

# Список литературы

- 1. Уилсон Р. Введение в теорию графов. Пер.с анг. 1977. 208с.
- 2. Герберт Шилдт «полный справочник по C++» вильямс,2006
- 3. Оре О. Графы и их применение: Пер.сангл.1965.176с.
- 4. Брайан Керниган, Деннис Ритчи «Язык программирования Си»

# 7.ПриложениеА Листингпрограммы

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <stdbool.h>
#include <locale.h>
#include <time.h>
#include <vector>
#include <fstream>
//#include <sstream>
using namespace std;
#define MAX_SIZE 100
int findUnconnectedNode(int parent[MAX_SIZE], int size) {
         for (int i = 0; i < size; ++i) {
                           if (parent[i] == -1) {
                                    return i;
         return -1; // Если не найдено несвязанных вершин
void printTreeStructure(ofstream& test, const vector<vector<int>>& children, int vertex, int depth = 0) {
  for (int i = 0; i < depth; ++i){
    cout << " ";
                 test << " ";}
  cout << vertex << "\n";</pre>
        test << vertex << "\n";
         //рекурсивно выводятся все смежные вершины для текущей вершины
  for (int child : children[vertex]) {
    printTreeStructure(test, children, child, depth + 1);
//создание непосредсвенной структуры
void buildTreeStructure(const int parent[], int size, vector<vector<int>>& children) {
  for (int i = 0; i < size; ++i) {
    if (parent[i] != -1) {
       children[parent[i]].push_back(i);
//Вывод остовного дерева
void printPrimMST(ofstream& test, int graph[MAX_SIZE][MAX_SIZE], int size, int parent[MAX_SIZE]) {
  vector<vector<int>> children(size);
  buildTreeStructure(parent, size, children);
  int root = findUnconnectedNode(parent, size);
  if (root != -1) {
    cout << "Визуализаяция остовного дерева:\n";
    printTreeStructure(test, children, root);
  } else {
```

```
cout << "Граф не содержит ни одного ребра.\n":
// Функция сохранения графа в файл
void saveGraphToFile(int graph[MAX_SIZE][MAX_SIZE], int size, const char* filename) {
        FILE* file = fopen(filename, "w");
        if (file == NULL) {
                 printf("Ошибка при открытии файла\n");
                 return;
         fprintf(file, "%d\n", size);
        for (int i = 0; i < size; ++i) {
                 for (int j = 0; j < size; ++j) {
                          fprintf(file, "%d ", graph[i][j]);
                 fprintf(file, "\n");
        fclose(file);
        printf("\nГраф успешно сохранен в файл G.txt\n\n");
// Функция сохранения результата алгоритма Прима в файл
void savePrimResultToFile(int parent[MAX_SIZE], int graph[MAX_SIZE][MAX_SIZE], int size, ofstream& test)
        //FILE* file = fopen(filename, "w");
        if (!test.is_open()) {
                 printf("Ошибка при открытии файла для сохранения результата\n");
        test << "Минимальное остовное дерево:" << endl;
        //fprintf(file, "Минимальное остовное дерево:\n");
        for (int i = 1; i < size; ++i) {
                 //fprintf(file, "Peopo: %d - %d, Bec: %d\n", parent[i], i, graph[i][parent[i]]);
                 test << "Pe6po: " << parent[i] << " - " << i << ", Bec: " << graph[i][parent[i]] << endl;
        //fclose(file):
        printf("Результат успешно сохранен в файл result.txt\n\n");
// Функция для проверки существования файла
bool fileExists(const char* filename) {
        FILE* file = fopen(filename, "r");
        if (file != NULL) {
                 fclose(file):
                 return true; // Файл существует
        return false; // Файл не существует
// Загрузка графа из файла
void loadGraphFromFile(int graph[MAX SIZE][MAX SIZE], int* size, const char* filename) {
                 FILE * file = fopen(filename, "r");
        if (file == NULL) {
                 printf("\nОшибка при открытии файла\n");
                 return;
        fscanf(file, "%d", size);
        for (int i = 0; i < *size; ++i) {
                 for (int i = 0; i < *size; ++i) {
                          fscanf(file, "%d", &graph[i][j]);
        fclose(file);
// Функция для заполнения неориентированного графа случайными числами
void fillUndirectedGraphRandom(int graph[MAX_SIZE][MAX_SIZE], int size)
```

```
srand(time(NULL)):
        // Заполнение графа нулями и единицами на главной диагонали
        for (int i = 0; i < size; ++i) {
                 for (int i = 0; i < size; ++i) {
                          if (i == j) {
                                   graph[i][j] = 0; // Главная диагональ остается нулевой
                          else {
                                   graph[i][j] = rand() % 2; // Заполнение ребер 0 или 1
                                   graph[j][i] = graph[i][j]; // Отражение изменений для неориентированного графа
                          }
        // Замена единиц случайными числами
        for (int i = 0; i < size; ++i) {
                 for (int j = 0; j < size; ++j) {
                          if (i != j && graph[i][j] == 1) {
                                   graph[i][j] = rand() \% 100; // Заполнение случайным числом вместо единицы
                                   graph[j][i] = graph[i][j]; // Отражение изменений для неориентированного графа
// Вывод графа в консоль
void printGraph(int graph[MAX_SIZE][MAX_SIZE], int size)
        printf("\nΓpaφ:\n");
        for (int i = 0; i < size; ++i)
                 for (int i = 0; i < size; ++i)
                          printf("%3d ", graph[i][j]);
                 printf("\n");
// Функция для нахождения несвязных вершин
// Функция выполнения алгоритма Прима для графа
void primAlgorithm(ofstream& test, int graph[MAX SIZE][MAX SIZE], int size, int parent[MAX SIZE]) {
        int minEdge[MAX SIZE]; // Минимальные веса рёбер
        bool visited[MAX SIZE]; // Посещали ли вершины
        // Инициализация значений
        for (int i = 0; i < size; ++i) {
                 minEdge[i] = INT MAX; // Установка начальных значений весов рёбер как максимальное
значение
                 visited[i] = false; // Начально ни одна вершина не посещена
                 parent[i] = -1; // Начально нет родительской вершины
        minEdge[0] = 0; // Стартовая вершина, вес ребра равен 0
        for (int count = 0; count < size - 1; ++count) {
                 int minVertex = -1; // Минимальный индекс вершины
                 for (int v = 0; v < size; ++v) {
                          if (!visited[v] && (minVertex == -1 || minEdge[v] < minEdge[minVertex])) {
                                   minVertex = v; // Нахождение вершины с минимальным весом ребра
                 visited[minVertex] = true; // Включаем найденную вершину в остовное дерево
                 // Обновляем веса смежных вершин
                 for (int v = 0; v < size; ++v) {
                          if (graph[minVertex][v] != 0 && !visited[v] && graph[minVertex][v] <
                                   minEdge[v]) {
                                   parent[v] = minVertex;
                                   minEdge[v] = graph[minVertex][v];
```

```
// Вывод ребер минимального остовного дерева
        printf("\nМинимальное остовное дерево:\n");
        int D = 0;
        struct Node* root = NULL;
        for (int i = 1; i < size; ++i) {
                 if (parent[i] != -1) {
                          printf("Peopo: %d - %d, Bec: %d\n", parent[i], i, graph[i][parent[i]]);
                 else {
                          printf("Ребро: %d - %d, не существует\n", findUnconnectedNode(parent, size),i);
        printPrimMST(test, graph, size, parent);
int main(void) {
         setlocale(LC_ALL, "RUS");
        int graph[MAX SIZE][MAX SIZE];
        int size, choice;
        char filename[50];
        int parent[MAX_SIZE];
        bool b = \{\};
  char buffer[20];
        ofstream ttt;
         printf("Курсовая работа\n");
        printf("Выполнил: Кузьмин Данила 22ВВВЗ\n");
        do {
                 printf("Меню:\n");
                 printf("1. Загрузить неориентированный граф из файла\n");
                 printf("2. Заполнить неориентированный граф автоматически\n");
                 printf("3. Завершить и выйти из программы\n");
                 printf("Выберите номер от 1-3: ");
                          while (1) {
                                   cin >> buffer;
                                    if (atoi(buffer) == 0 && buffer[0] != '0') {
                                            cout << "Введите только целое число" << endl;
                                            continue:
         if (buffer[1]!=0) {
            cout << "only integers" << endl;
            continue;
                                   else {
                                            choice = atoi(buffer);
                                            break;
       };
                 switch (choice) {
                 case 1:
                          printf("\nВведите имя файла для загрузки: ");
                          scanf("%s", filename);
                          if (!fileExists(filename)) {
                                    printf("\nВведён несуществующий файл\n\n");
                                    break;
```

```
loadGraphFromFile(graph, &size, filename);
                  printGraph(graph, size);
                  ttt.open("result.txt");
                  primAlgorithm(ttt, graph, size, parent);
                  saveGraphToFile(graph, size, "G.txt");
                  savePrimResultToFile(parent, graph, size, ttt);
                 ttt.close():
                 break:
        case 2:
                 printf("\nВведите размер неориентированного графа: ");
                  while (1) {
                           cin >> buffer;
                           if (atoi(buffer) == 0 && buffer[0] != '0') {
                                    cout << "Введите только целое число" << endl;
                                    continue;
                           else {
                                    size = atoi(buffer);
                                    break;
                 if (size <= 0) {
                           printf("\nВведён неверный размер графа\n\n");
                           break;
                  fillUndirectedGraphRandom(graph, size);
                  printGraph(graph, size);
                  ttt.open("result.txt");
                  primAlgorithm(ttt, graph, size, parent);
                  saveGraphToFile(graph, size, "G.txt");
                  //cout << "jope" << endl;
                  savePrimResultToFile(parent, graph, size, ttt);
                  ttt.close();
                  break;
        case 3:
                  return 3; // Завершение программы при выборе "3"
        default:
                  printf("\nСделан неверный выбор. Пожалуйста, выберите снова.\n\n");
                  break;
} while (true);
```