Министерство образования Республики Беларусь

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №4**

**По дисциплине «Компиляторные технологии»**

**Тема: «Генерация и оптимизация объектного кода»**

Выполнил: Купаленко А.Д.

Группа 21ИТ1

Проверила: Сыцевич Д.Н.

Преподаватель кафедры ТП

Полоцк 2022

Тема: Генерация и оптимизация объектного кода  
Цель работы: изучение основных принципов генерации компилятором объектного  
кода, ознакомление с методами оптимизации результирующего объектного кода для  
линейного участка программы с помощью свертки и исключения лишних операций.  
ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ (ответы на контрольные вопросы):  
Генерация объектного кода – это перевод компилятором внутреннего  
представления исходной программы в цепочку символов выходного языка. Поскольку  
выходным языком компилятора (в отличие от транслятора) может быть только либо  
язык ассемблера, либо язык машинных кодов, то генерация кода порождает  
результирующую объектную программу на языке ассемблера или непосредственно на  
машинном языке (в машинных кодах).  
Генерация объектного кода выполняется после того, как выполнены лексический и  
синтаксический анализ программы и все необходимые действия по подготовке к  
генерации кода: проверены семантические соглашения входного языка (семантический  
анализ), выполнена идентификация имен переменных и функций, распределено  
адресное пространство под функции и переменные и т. д.  
Вариант 11  
Листинг программы  
#include <windows.h>  
#include <iostream>  
#include <fstream>  
#include <iostream>  
#include <stdlib.h>  
#include <string.h>  
#include <string>  
#include <ctype.h>  
#include <cstdio>  
#include <cstring>  
#include <windows.h>  
using namespace std;  
bool isPunctuator(char ch)  
{

if (ch == ' ' || ch == '+' || ch == '-' || ch == '\*' ||  
ch == '/' || ch == ',' || ch == '>' ||  
ch == '<' || ch == '(' || ch == ')' ||  
ch == '[' || ch == ']' || ch == '{' || ch == '}' ||  
ch == '&' || ch == '|')  
{  
return true;  
}  
return false;  
}  
bool validIdentifier(char\* str)  
{  
if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||  
str[0] == '3' || str[0] == '4' || str[0] == '5' ||  
str[0] == '6' || str[0] == '7' || str[0] == '8' ||  
str[0] == '9' || isPunctuator(str[0]) == true)  
{  
return false;  
}  
int i, len = strlen(str);  
if (len == 1)  
{  
return true;  
}  
else  
{  
for (i = 1; i < len; i++)  
{  
if (isPunctuator(str[i]) == true)  
{  
return false;  
}  
}  
}  
return true;  
}  
bool isRavno(char str)  
{  
if (str == '=') return true;  
else return false;  
}  
bool isZnak(char str)  
{

if (str == ';') return true;  
else return false;  
}  
bool isDelimetr(char str)  
{  
if (str == '(' || str == ')' || str == '{' || str == '}')  
return true;  
else return false;  
}  
bool isOperator(char ch)  
{  
if (ch == '+' || ch == '-' || ch == '\*' ||  
ch == '/' || ch == '>' || ch == '<' || ch == '|' || ch  
== '&')  
{  
return true;  
}  
return false;  
}  
bool isKeyword(char\* str)  
{  
if (!strcmp(str, "if") || !strcmp(str, "else") ||  
!strcmp(str, "while") || !strcmp(str, "do") ||  
!strcmp(str, "break") || !strcmp(str, "continue")  
|| !strcmp(str, "int") || !strcmp(str, "double")  
|| !strcmp(str, "float") || !strcmp(str, "return")  
|| !strcmp(str, "char") || !strcmp(str, "case")  
|| !strcmp(str, "long") || !strcmp(str, "short")  
|| !strcmp(str, "typedef") || !strcmp(str, "switch")  
|| !strcmp(str, "unsigned") || !strcmp(str, "void")  
|| !strcmp(str, "static") || !strcmp(str, "struct")  
|| !strcmp(str, "sizeof") || !strcmp(str, "long")  
|| !strcmp(str, "volatile") || !strcmp(str, "typedef")  
|| !strcmp(str, "enum") || !strcmp(str, "const")  
|| !strcmp(str, "union") || !strcmp(str, "extern")  
|| !strcmp(str, "bool") || !strcmp(str, "cout")  
|| !strcmp(str, "or") || !strcmp(str, "and")  
|| !strcmp(str, "then")) return true;  
else return false;  
}  
bool isIf(char\* str)  
{