**БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Международный Институт Дистанционного Образования

Кафедра “Информационные системы и технологии”

Контрольная работа по дисциплине «Системное программирование»

Тема работы:

**«Программирование на Си»**

Выполнил:   
студент 3 курса, гр. 41703120  
Реут Владислав Леонидович

Проверил: Бумай А.Ю.

Минск 2022

**ВВЕДЕНИЕ**

Язык программирования С (си) является одним из самых популярных и распространенных языков. Он представляет компилируемый язык программирования общего назначения со статической типизацией, разработанный в 1969—1973 годах в компании Bell Labs программистом Деннисом Ритчи (Dennis Ritchie).

Язык С нередко называют языком программирования "среднего уровня" или даже "низкого уровня", так как он сочетает элементы языков высокого уровня с функциональностью и производительностью ассемблера и работает близко к аппаратной части компьютера. В итоге мы можем манипулировать данными на низком уровне и при этом использовать высокоуровневые конструкции для управления работы программы.

Первоначально язык С предназначался для написания операционной системы Unix. Впоследствии Си стал одним из популярных языков, а его основной сферой применения стало системное программирование, в частности, создание операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, Linux большей частью написан на Си. Однако только системным программированием применение данного языка не ограничивается. Данный язык можно использовать в программах любого уровня, где важны скорость работы и производительность. Так, мы можем писать с помощью Си и прикладные приложения, и даже веб-сайты (используя технологию CGI - Common Gateway Interface). Но, конечно, для создания графического интерфейса и веб-приложений, как правило, выбираются более подходящие инструменты и технологии, но тем не менее круг использования Си довольно широк. Это в немалой степени определило популярность языка. Например, в известном рейтинге языков программирования TIOBE язык С долгое время уверенно удерживает второе место.

Несмотря на большие возможности язык Си одновременно довольно прост. Он не содержит много конструкций, библиотек, его легко осваивать и изучать. Поэтому нередко его выбирают в качестве языка для изучения в целом программированию.

Си является компилируемым языком, а это значит, что компилятор транслирует исходный код на Си в исполняемый файл, который содержит набор машинных инструкций. Но разные платформы имеют свои особенности, поэтому скомпилированные программы нельзя просто перенести с одной платформы на другую и там уже запустить. Однако на уровне исходного кода программы на Си обладают переносимостью, а наличие компиляторов, библиотек и инструментов разработки почти под все распространенные платформы позволяет компилировать один и тот же исходный код на Си в приложения под эти платформы.

Развитие Си оказало большое влияние в целом на развитие языков программирования. В частности, его синтаксис стал основой для таких языков как С++, С#, Java, PHP, JavaScript. Особо следует сказать про связь с C++. C++ напрямую произошёл от Си. Но впоследствии их развитие происходило отдельно друг от друга, и даже появилась несовместимость между ними. Стандарт C99 добавил в язык Си ряд конфликтующих с C++ особенностей. В итоге в настоящее время оба языка являются фактически самодостаточными и развиваются независимо.

**Решение задач:**

**Задача 1**

**Вариант №5**

В написанном выражении ((((1?2)?3)?4)?5)?6 вместо каждого знака «?» вставить знак одной из четырех арифметических операций +, –, ?, / так, чтобы результат вычислений равнялся 35 (при делении дробная часть в частном отбрасывается). Достаточно найти одно решение.

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define N 6

int main()

{

int a[N] = { 1, 2, 3, 4, 5, 6, };

int result = 35;

char o[N - 1];

int r;

for (int i = 0; i < pow(4, N - 1); ++i) {

r = a[0];

int p = 1;//4^(j - 1)

for (int j = 1; j < N; ++j) {

switch ((i / p) % 4) {

case 0:

r += a[j];

o[j - 1] = '+';

break;

case 1:

r -= a[j];

o[j - 1] = '-';

break;

case 2:

r \*= a[j];

o[j - 1] = '\*';

break;

case 3:

r /= a[j];

o[j - 1] = '/';

break;

}

p \*= 4;

}

if (r == result) {

printf("((((%d ", a[0]);

for (int k = 1; k < N; ++k)

printf("%c %d ) ", o[k - 1], a[k]);

printf("= %d\n", result);

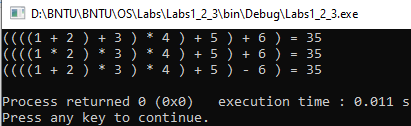
}

}

return 0;

}

Результат выполнения программы:



**Задача 2**

В написанном выражении ((((1?2)?3)?4)?5)?6 вместо каждого знака «?» вставить знак одной из четырех арифметических операций +, –, ?, / так, чтобы результат вычислений равнялся 35 (при делении дробная часть в частном отбрасывается). Достаточно найти одно решение.  
Решение задачи оформит в виде функции, которая получает в качестве параметров указатель на массив с числами выражения, количество чисел в выражении и значение выражения.

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define N 6

void expression(int\* a, int S, int result)

{

char o[S - 1];

int r;

for (int i = 0; i < pow(4, S - 1); ++i) {

r = a[0];

int p = 1;//4^(j - 1)

for (int j = 1; j < S; ++j) {

switch ((i / p) % 4) {

case 0:

r += a[j];

o[j - 1] = '+';

break;

case 1:

r -= a[j];

o[j - 1] = '-';

break;

case 2:

r \*= a[j];

o[j - 1] = '\*';

break;

case 3:

r /= a[j];

o[j - 1] = '/';

break;

}

p \*= 4;

}

if (r == result) {

printf("((((%d ", a[0]);

for (int k = 1; k < S; ++k)

printf("%c %d ) ", o[k - 1], a[k]);

printf("= %d\n", result);

}

}

}

int main()

{

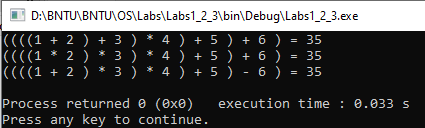
int a[N] = { 1, 2, 3, 4, 5, 6, };

expression(a, N, 35);

return 0;

}

Результат выполнения программы:



**Задача 3**

**Вариант №5**  
Массив размерностью MxN. Необходимо дополнить его (M+1)-й строкой и (N+1)-м столбцом, в которых записать суммы элементов соответствующих строк и столбцов. В элементе aM+1,N+1 должна храниться сумма всех элементов массива.

#include <stdio.h>

#include <stdlib.h>

#define M 5

#define N 5

// Функция выделения динамической памяти для двумерного массива

int\*\* malloc2DArray( int m, int n)

{

int\*\* arr = (int\*\*) malloc(m\*sizeof(int\*));

for (int i = 0; i < m; ++i)

arr[i] = (int\*) malloc(n\*sizeof(int));

return arr;

}

// Функция освобождения памяти

void free2DArray(int\*\* arr, const int m)

{

for (int i = 0; i < m; ++i)

free(arr[i]);

free(arr);

}

// Печать двумерного массива

void print2DArray(int\*\* arr, int m , int n)

{

for (int i = 0; i < m; ++i)

{

for (int j = 0; j < n; ++j)

printf("%d ", arr[i][j]);

printf("\n");

}

}

// Функция суммирования строк

int sumRow(int \*\*arr, int row, int cols)

{

int sum = 0;

for (int j = 0; j < cols; ++j)

sum += arr[row][j];

return sum;

}

// Функция суммирования столбцов

int sumCol(int \*\*arr, int col, int rows)

{

int sum = 0;

for (int i = 0; i < rows; ++i)

sum += arr[i][col];

return sum;

}

int main()

{

// создаём новую матрицу MxN

int\*\* mtrx = malloc2DArray(M+1, N+1);

// заполняем

for (int i = 0; i < M; ++i)

for (int j = 0; j < N; ++j)

mtrx[i][j] = j;

// выводим

printf("Matrix\n");

print2DArray(mtrx, M, N);

int rowsSum = 0;

for (int i = 0; i < M; ++i)

{

mtrx[i][N] = sumRow(mtrx, i, N);

rowsSum += mtrx[i][N];

}

int colsSum = 0;

for (int j = 0; j < N; ++j)

{

mtrx[M][j] = sumCol(mtrx, j, M);

colsSum += mtrx[M][j];

}

mtrx[M][N] = rowsSum + colsSum;

// выводим

printf("Matrix\n");

print2DArray(mtrx, M+1, N+1);

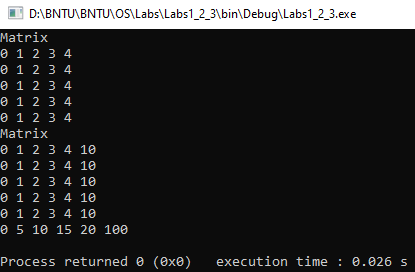
// освобождаем выделенную память

free2DArray(mtrx, M);

return 0;

}

Результат работы функции:



**Задача 4**

**Вариант №5**  
Имеется элемент стека (дисциплина обслуживания LIFO):

struct Stack

{

char \* data; //указатель на данные

Stack \* prev; //указатель на предыдущий элемент

}

\* top; //указатель на вершину стека

Определите следующие функции:

void Push(Stack\*\* stack, char\* data); // поместить данные в стек;

char\* Pop(Stack\*\* stack); // извлечь данные из стека (при этом элемент удаляется из стека);

PrintStack(Stack\* stack); // вывод на экран содержимого стека.

Листинг:

typedef struct Stack

{

char \* data;//указатель на данные

struct Stack \* prev;//указатель на предыдущий элемент

}Stack;

void Push(Stack\*\* stack, char\* data) // поместить данные в стек;

{

Stack \*tmp = (Stack\*)malloc(sizeof(Stack));

tmp->data = data;

tmp->prev = (\*stack) ? \*stack : NULL;

\*stack = tmp;

}

void Pop(Stack\*\* stack)

{

if (!(\*stack))//если нечего удалять

return;

Stack \*tmp = \*stack;//создаем указатель на вершину

\*stack = (\*stack)->prev;//вершину сдвигаем вниз

free(tmp);//очищаем память (удаляем старую вершину)

}

void PrintStack(Stack \*stack) /\* Печать списка \*/

{

Stack \*tmp = stack;

while (tmp)

{

printf("%s\n", tmp->data);

tmp = tmp->prev;

}

}

int main()

{

Stack \*top = NULL;

Push(&top, "abc");

Push(&top, "efg");

Push(&top, "mmm");

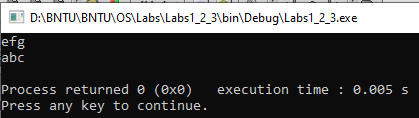
Pop(&top);

PrintStack(top);

return 0;

}

Результат работы функции:



**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО КУРСУ СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ
2. Сайт metanit.com