Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Функции с переменным числом параметров»

Выполнила:

Студент(ка) 1 курса 7 группы

Подшиваленко Диана Игоревна

Проверил:

Белодед Николай Иванович

2024, Минск

**Содержание**

1. Предисловие … 3
2. Понятие функции с переменным числом параметров … 4
3. Примеры реализации функции с переменным числом параметров разными способами … 5

3.1 Реализация функции без использования специальных библиотек … 5

3.1.1 Функция нахождения суммы чисел … 5

3.1.2 Функция подсчета количества слов на букву «а» … 5

3.1.3 Функция нахождения минимального числа из переданных массивов … 6

3.2 Реализация функции с использованием библиотеки cstdarg… 6

3.2.1 Функция нахождения суммы чисел … 6

3.2.2 Функция подсчета количества слов на букву «а» … 7

3.2.3 Функция нахождения минимального числа из переданных массивов … 8

3.3 Реализация функции с использованием библиотеки initializer\_list …8

3.3.1 Функция нахождения суммы чисел … 8

3.3.2 Функция подсчета количества слов на букву «а» … 9

3.3.3 Функция нахождения минимального числа из переданных массивов … 9

3.4 Реализация функции с использованием шаблона … 10

3.4.1 Функция нахождения суммы чисел … 10

3.4.2 Функция подсчета количества слов на букву «а» … 11

3.4.3 Функция нахождения минимального числа из переданных массивов … 11

1. Вывод … 12

**Предисловие**

В данном реферате будет рассмотрено использование функции с переменным числом параметров, реализация различными способами.

**1. Понятие функции с переменным числом параметров**

Язык программирования С++ допускает использование функций, которые имеют нефиксированное количество параметров. Более того может быть неизвестным не только количество, но и типы параметров. То есть точное определение параметров становится известным только во время вызова функции. Функции с переменным числом параметров объявляются как обычные функции, но вместо недостающих аргументов ставится многоточие.

Синтаксис объявления такой функции выглядит так:

тип имя\_функции(обязательные параметры, ...)

При этом надо учитывать, что функция должна иметь как минимум один обязательный параметр.

Такие функции могут быть полезны в следующих случаях:

1) Функции с переменным числом параметров позволяют передавать произвольное число аргументов, что может быть удобно, когда нужно обрабатывать разные варианты входных данных. Например, функция printf использует переменные аргументы для форматированного вывода в консоль.

2) Функции с переменным числом параметров могут быть использованы для создания обобщенных алгоритмов или контейнеров, которые могут работать с различными типами данных. Можно, например, создать функцию, которая будет выводить в консоль элементы любого контейнера, переданного в качестве аргумента.

3) Если требования к функции изменяются в процессе разработки, использование функций с переменным числом параметров позволяет гибко адаптировать функцию к новым условиям без необходимости изменения ее интерфейса.

4) Функции с переменным числом параметров могут упростить код, особенно когда нужно передавать произвольное число аргументов. Они позволяют сократить объем кода и сделать его более читаемым и модульным.

Но несмотря на это, стоит помнить, что функции с переменным числом параметров могут быть сложными для отладки и подвержены ошибкам типизации. Поэтому они должны использоваться с осторожностью и внимательно проверяться на правильность типов аргументов во время выполнения программы. Иначе, это может привести к непредсказуемому поведению и ошибкам, которые могут быть сложными для обнаружения и исправления.

**2. Примеры реализации функции с переменным числом параметров разными способами**

Для лучшего понимания будут использованы несложные функции: вычисления суммы чисел, подсчет количества слов, которые начинаются на латинскую букву «a», нахождение минимального числа из переданных массивов.

**2.1 Реализация функции без использования специальных библиотек**

**2.1.1 Функция нахождения суммы чисел**

#include <iostream>

#include<Windows.h>

#include <conio.h>

using namespace std;

void sum(int kol...) {

int\* p = &kol;

p++;

int sumn = 0;

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

sumn += \*p++;

}

cout << sumn << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

sum(10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

sum(5, 10, 20, 30, 40, 50);

sum(3, -1, -2, -3);

sum(13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13);

sum(2, 100, -100);

}

Результат работы:

**2.1.2 Функция подсчета количества слов на букву «а»**

#include <iostream>

#include<Windows.h>

#include <conio.h>

using namespace std;

void kol\_world(int kol, const char a[]...) {

const char\*\* p = &a;

int kolvo = 0;

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

if ((\*p)[0] == 'a') {

kolvo += 1;

}

p++;

}

cout << kolvo << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

kol\_world(7, "Masha", "Sasha", "Anna", "apple", "sort", "alone", "string");

kol\_world(5, "C++", "Java", "HTML", "Python", "C#");

kol\_world(3, "argument", "array", "algorithm");

}

Результат выполнения:

**2.1.3 Функция нахождения минимального числа из переданных массивов.**

#include <iostream>

#include<Windows.h>

#include <conio.h>

using namespace std;

void matrix\_sum(int kol, int size, int a[] ...) {

int\*\* p = &a;

int min = \*\*p;

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

for (int j = 0; j < size; j++) {

if ((\*p)[j] < min) {

min = (\*p)[j];

}

}

p++;

}

cout << min << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int a1[3] = {1, 2, 3};

int a2[3] = {12, 22, 32};

int b1[2] = {13, 26};

int b2[2] = {-11, 21};

int b3[2] = {10, -20};

int c[4] = {1, 2, -3, 10};

matrix\_sum(2, 3, a1, a2);

matrix\_sum(3, 2, b1, b2, b3);

matrix\_sum(1, 4, c);

}

Результат выполнения:

**2.2 Реализация функции с использованием библиотеки cstdarg**

**2.2.1 Функция нахождения суммы чисел**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <cstdarg>

using namespace std;

void sum(int kol...) {

va\_list arg;

va\_start(arg, kol);

int sumn = 0;

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

sumn += va\_arg(arg, int);

}

va\_end(arg);

cout << sumn << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

sum(10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

sum(5, 10, 20, 30, 40, 50);

sum(3, -1, -2, -3);

sum(13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13);

sum(2, 100, -100);

}

Результат выполнения:

**2.2.2 Функция подсчета количества слов на букву «а»**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <cstdarg>

using namespace std;

void kol\_world(int kol ...) {

va\_list arg;

va\_start(arg, kol);

int kolvo = 0;

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

if (va\_arg(arg, const char\*)[0] == 'a') {

kolvo += 1;

}

}

va\_end(arg);

cout << kolvo << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

kol\_world(7, "Masha", "Sasha", "Anna", "apple", "sort", "alone", "string");

kol\_world(5, "C++", "Java", "HTML", "Python", "C#");

kol\_world(3, "argument", "array", "algorithm");

}

Результат выполнения:

**2.2.3 Функция нахождения минимального числа из переданных массивов.**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <cstdarg>

using namespace std;

void matrix\_sum(int kol, int size, ...) {

va\_list arg;

va\_start(arg, size);

int min = 10000000;

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

int\* b = va\_arg(arg, int\*);

for (int j = 0; j < size; j++) {

if (b[j] < min) {

min = b[j];

}

}

}

va\_end(arg);

cout << min << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int a1[3] = { 1, 2, 3 };

int a2[3] = { 12, 22, 32 };

int b1[2] = { 13, 26 };

int b2[2] = { -11, 21 };

int b3[2] = { 10, -20 };

int c[4] = { 1, 2, -3, 10 };

matrix\_sum(2, 3, a1, a2);

matrix\_sum(3, 2, b1, b2, b3);

matrix\_sum(1, 4, c);

}

Результат выполнения:

**2.3 Реализация функции с использованием библиотеки initializer\_list**

**2.3.1 Функция нахождения суммы чисел**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <initializer\_list>

using namespace std;

void sum(initializer\_list<int> values) {

int sumn = 0;

for (int i : values) {

sumn += i;

}

cout << sumn << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

sum({ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 });

sum({ 10, 20, 30, 40, 50 });

sum({ -1, -2, -3 });

sum({ 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13 });

sum({ 100, -100 });

}

Результат выполнения:

**2.3.2 Функция подсчета количества слов на букву «а»**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <initializer\_list>

using namespace std;

void kol\_world(initializer\_list<const char\*> strings) {

int kolvo = 0;

for (const char\* i : strings) {

if (i[0] == 'a') {

kolvo += 1;

}

}

cout << kolvo << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

kol\_world({ "Masha", "Sasha", "Anna", "apple", "sort", "alone", "string" });

kol\_world({ "C++", "Java", "HTML", "Python", "C#" });

kol\_world({ "argument", "array", "algorithm" });

}

Результат выполнения:

**2.3.3 Функция нахождения минимального числа из переданных массивов.**

#include <iostream>

#include<Windows.h>

#include <conio.h>

#include <initializer\_list>

using namespace std;

void matrix\_sum(int size, initializer\_list<const int\*> mas) {

int min = 10000000;

for (const int\* i : mas) {

for (int j = 0; j < size; j++) {

if(i[j] < min){

min = i[j];

}

}

}

cout << min << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int a1[3] = { 1, 2, 3 };

int a2[3] = { 12, 22, 32 };

int b1[2] = { 13, 26 };

int b2[2] = { -11, 21 };

int b3[2] = { 10, -20 };

int c[4] = { 1, 2, -3, 10 };

matrix\_sum(3, { a1, a2 });

matrix\_sum(2, { b1, b2, b3 });

matrix\_sum(4, { c });

}

Результат выполнения:

**2.4 Реализация функции с использованием шаблона**

Для использования этого способа обязательно должен быть стандарт языка с++ не ниже 17 версии.

**2.4.1 Функция нахождения суммы чисел**

#include <iostream>

#include<Windows.h>

#include <conio.h>

using namespace std;

template<typename ... Args>

void sum(Args... arg) {

int sumn = 0;

cout << (arg + ...) << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

sum(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

sum(10, 20, 30, 40, 50);

sum(-1, -2, -3);

sum(13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13);

sum(100, -100);

}

Результат выполнения:



**2.4.2 Функция подсчета количества слов на букву «а»**

#include <iostream>

#include<Windows.h>

#include <conio.h>

using namespace std;

template<typename ...Args>

void kol\_world(Args... arg) {

int kolvo = 0;

((arg[0] == 'a' ? kolvo += 1 : kolvo), ...);

cout << kolvo << '\n';

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

kol\_world("Masha", "Sasha", "Anna", "apple", "sort", "alone", "string");

kol\_world("C++", "Java", "HTML", "Python", "C#");

kol\_world("argument", "array", "algorithm");

}

Результат работы:



**2.4.3 Функция нахождения минимального числа из переданных массивов.**

#include <iostream>

#include<Windows.h>

#include <conio.h>

using namespace std;

template<typename T>

T search\_min(T\* array, int size) {

T min = array[0];

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

if (array[i] < min) {

min = array[i];

}

}

return min;

}

template<typename... Args>

void matrix\_sum(int kol, int size, Args... arrays) {

int min = 10000000;

int\* minimum = new int[kol];

int i = 0;

((minimum[i++] = search\_min(arrays, size)), ...);

for (int j = 0; j < kol; j++) {

if (minimum[j] < min) {

min = minimum[j];

}

}

delete[] minimum;

cout << min << '\n';

}

int main() {

int\* a1 = new int[3]{ 1, 2, 3 };

int\* a2 = new int[3]{ 12, 22, 32 };

int\* b1 = new int[2]{ 13, 26 };

int\* b2 = new int[2]{ -11, 21 };

int\* b3 = new int[2]{ 10, -20 };

int\* c = new int[4]{ 1, 2, -3, 10 };

matrix\_sum(2, 3, a1, a2);

matrix\_sum(3, 2, b1, b2, b3);

matrix\_sum(1, 4, c);

delete[] a1;

delete[] a2;

delete[] b1;

delete[] b2;

delete[] b3;

delete[] c;

}

Результат выполнения:



**Вывод**

Функции с переменным числом параметров могут быть очень полезным инструментом для работы с произвольным числом данных. Однако не стоит забывать, что функции с переменным числом параметров могут быть сложными для отладки и подвержены ошибкам типизации. Поэтому они должны использоваться с осторожностью и внимательно проверяться на правильность типов аргументов во время выполнения программы. Иначе, это может привести к непредсказуемому поведению и ошибкам, которые могут быть сложными для обнаружения и исправления.