14.11.2 перегрузка

одна из важнейших особенностей ф-ции си явл перегузка – в момент выполнения кода компилятором решается какая ф-ция будет выполняться

перегрузка явл одним из важнейшим факторов ООП(объекты по разному могут реагировать на одно и то же сообщение и для этих целей исп перегрузка)

для того чтобы ф-ция была перегружаема должны выполняться правила

1)имена ф-ции должны быть одинаковы

2)к-во арг перегружаемой ф-ции должно отличаться

3)если к-во арг совпадает то хотя бы 1 тип арг должен отличаться

Замечание: тип возвращ значения не имеет

АЛГОРИТМЫ ФУНКЦИОНИРОВАНИЯ

Замечание: компилятора при перегрузке явл одинаковый тип для ссылки указателя и значения

int adder(int iarray[]);

float adder(float farray[]);

main() {

int iarray[7]={5,1,6,20,15,0,12};

float farray[7]={3.3,5.2,0.05,1.49,3.12345,31.0,2.007};

int isum;

float fsum;

isum = adder(iarray);

fsum = adder(farray);

…

}

int adder(int iarray[]);

{

int I;

int ipartial;

ipartial=iarray[0];

for(i=1;i<7;i++)

ipatrial+=iarray[i];

return (ipartial);

}

float adder(float farray[])

{

int I;

float fpartial;

fpartial=farray[0];

for(i=1;i<7;i++)

fpartial+=farray[i];

return (fpartial);

}

14.11.3 Функции с переменным числом параметров. Многоточие(…)

В языке си аргументы ф-ции передаются справа налево, это позволят реализовать – возможность передачи неопределенного к-ва аргументов

определение ф-ции которое имеет неопределенное к-во аргументов обозначается символом … после последнего арг

Замечание: вообще-то си знает к-во передаваемых аргументов, чаще всего их к-во опред несколькими способами

в общем случае ф-ция прин. произ к-во арг опис

тип имя (тип арг имя арг)

тело ф-ции

ЛУЧШЕ НЕ ИСП ТАКОЙ СПОСОБ ПЕРЕДАЧИ ДАННЫХ, Т.К. КОМПИЛЯТОР ДО ПОСЛЕДНЕГО НЕ ЗНАЕТ НЕ ТИПЫ АРГУМЕНТОВ, НЕ ИХ К-ВО

лучших способ исп ф-ции с неопред к-вом параметров …

14.11.3.1 Задание числа доп параметров с помощбю первого параметра

пример

void main() {

int sum(int n,…);

cout << “\n 4+6=” << sum(2,4,6);

cout << “\n 1

}

int sum(int n,…)

{

int s=0;

int \*p=&n;

for(int I = 1;i<=n;i++)

s+=\*++p;

return s;

}

Для исп ф-ции в памяти поялвяется стек

Если известен адрес первого аргумента, то мы может вычислить адреса последующим, расположенные последовательно друг за другом

Замечание: возникают проблемы даже в этом случае у компилятора, т.к. к-во арг целого значения

сами данные могут быть более сложные(тип дабл)

в этом случае прим…

пример

double f(int n, …) //количество

{

int \*p = &n;

double sum = 0, count = n;

for(; n--;(double\*)p++)

sum+=(\*p);

return ((sum)? sum / count : 0);

}

void main()

{

double y = f(4, 1.0, 2.0, 3.0, 4.0, 0.0); //not 2.5

cout << y << endl;

system(“pause”);

}

решение обозначенной проблемы существует

14.11.3.2 Определение конца списка параметров с помощью параметра индикатора

void MyFunc(int a,…)

{

int \*P=&a;

while(\*P)

{

cout << \*P << “ “;

P++;

}

}

int main() {

MyFunc(100,200,300,400,0);

cout << “\n”;

}

14.11.3.3 Использование спец набора макроопределений

void va\_start(va\_list param, <последний явный параметр>);

type va\_arg(va\_list param, type);

void va\_end(va\_list param);

для исп макросов нужно исп заголовок <stdarg.h>

int sum(int count,…)

{

va\_list ap;

int s = 0;

va\_start(ap,count);

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

s+=va\_arg(ap,int);

va\_end(ap);

return s;

}

int main()

{

cout << sum(3,20,30,40) << endl;

return 0;

}

14.11.3.4